

Goal-oriented design of value and process models from patterns

Zlatko Zlatev

PhD dissertation committee:

Promotor:

Prof. dr. R. J. Wieringa, Universiteit Twente, Enschede, The Netherlands

Members:

Dr. M. Reichert, Universiteit Twente, Enschede, The Netherlands

Prof. dr. J. van Hillegersberg, Universiteit Twente, Enschede, The Netherlands

Prof. dr. E. Dubois, Institute Henri Tudor, Luxembourg

Dr. J. Gordijn, Vrije Universiteit Amsterdam, The Netherlands

Dr. Andreas Wombacher, Universiteit Twente, Enschede, The Netherlands

Prof. dr. J.M. Akkermans, Vrije Universiteit Amsterdam, The Netherlands



CTIT Ph.D. thesis Series No. 07-102

Centre for Telematics and Information Technology (CTIT) P.O. Box 217 -
7500 AE Enschede - The Netherlands



SIKS Dissertation Series No. 2007-22

The research reported in this thesis has been carried out under the auspices
of SIKS, the Dutch Research School for Information and Knowledge
Systems.

ISBN: 978-90-365-2576-3

ISSN: 1381-3617

Copyright © 2007, Zlatko Zlatev, Enschede, The Netherlands

**GOAL-ORIENTED DESIGN OF VALUE AND
PROCESS MODELS FROM PATTERNS**

DISSERTATION

to obtain
the degree of doctor at University of Twente,
on the authority of the rector magnificus,
prof. dr. W.H.M. Zijm,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, 4th October 2007, at 16.45

by

Zlatko Vasilev Zlatev

born on 24th August, 1976
in Botevgrad, Bulgaria

This dissertation is approved by:

Prof. dr. R. J. Wieringa

Acknowledgements

Doing a PhD is a lonely project. However, it would not have been possible without a number of people to whom I would like to express my gratitude.

I want to thank Roel Wieringa for the endless patience and his belief that I would finish at all. In retrospect, I am pleased that he did not make it any easier for me. Now, I see and appreciate the value of reexamining the main problem at every meeting. Of everything I learned from him, what comes to my mind now is: “We are not looking for fantasy solutions to a fantasy problem”. Roel, for this and all your guidance, thank you!

If there is one person without whom this work would not have been possible, this is Andreas Wombacher. Andreas, thank you for giving me the confidence that I can write. Thank you for teaching me all the tricks of the trade. You are a true mentor! I express my utmost respect! And of course, thank you for co-authoring ‘The Masterpiece’.

Stanislav (Stanislav Pokraev), thank you for practically holding my hand in my first paper. Yes, the one that made it to the NASA research report. Who would have believed this ☺ Thank you for the never-ending optimism during the darkest periods. Thank you also for the sharp realism when it came to applicability of ideas.

I thank Jaap Gordijn for providing me with fresh perspectives on my research problem, with the insight into how to position better my research in the field of economics, and with the help in validating my framework.

I thank Maarten Fokkinga for the support in one of my chapters. I thank him for the genuine intention to help, for the effort to reach an in-depth understanding of the problem, and for the reviews of my text.

I thank Nikolay Diakov for the energy and determination he put in the work that unfortunately did not make it to the thesis.

I thank Ivan Kurtev for his readiness to discuss with me any (research) topic at anytime. For more or less the opposite, I thank Natasa Jovanovic for agreeing NOT to discuss my PhD research, at anytime.

I thank Marko Smiljanic for the unique sense of humor, but more importantly for his help surveying a branch of related work.

I thank Manfred Reichert for the positive distraction, for constantly reminding me that there is life outside the office as well as after the PhD.

I thank Niki (Nikolay Kavaldjiev) and Vancheto (Ivanka Pokraeva) for the moral support. Thank you for staying positive and for believing that one day I will finish.

And last but not least, I thank Damiano Bolzoni, Dulce Pumareja, Lianne Bodenstaff, Wouter Kuijper, and Xiaomeng Su for keeping my work/life balance just about right.

Contents

Chapter I Introduction	1
1 Background	1
2 Research Path	2
3 Research Scope	4
4 Research Questions and Approach	5
5 Research Results	7
6 Outline	8
7 Summary	9
Chapter II Design Knowledge in Existing e-Business Models.....	11
1 Introduction	12
2 What is a Pattern?.....	12
3 On Discovery of Patterns	13
4 Documenting Patterns	14
5 Annotating Patterns with Capability Models.....	19
6 Discovery of Value Patterns.....	20
7 Discovery of Process Patterns	24
8 Relation between Value and Process Patterns	26
9 Structure of the Library	26
10 Related work	26
11 Summary	27
Chapter III Design Framework for e-Business Models	29
1 Introduction	29
2 Motivation	30
3 Terminology	31
4 Review of Existing Frameworks	32
5 Objective and Available Resources	35
6 The Framework	36
7 Related work	41
8 Summary	42
Chapter IV Goal-modelling	43
1 Introduction	44
2 Goal Representation	45
3 Relations between Goals	50
4 Goal Model.....	58
5 Satisfaction Calculation.....	60
6 Related work	72
7 Summary	73
Chapter V Selection Procedure	75
1 Introduction	76
2 The Running Example.....	76
3 Selection Procedure.....	79
4 Generation of Alternative Goal Models	87

5 Matching Goal and Capability Models.....	102
6 Integrating Goal and Capability Models.....	112
7 Propagation and Evaluation Steps	115
8 Improved Selection Procedure: the Pre-evaluation Step.....	115
9 Related work.....	116
10 Summary.....	117
Chapter VI Synthesis of Value and Process Patterns.....	119
1 Introduction	119
2 Synthesis of Value Fragments	120
3 Synthesis of Process Fragments.....	129
4 Related work.....	141
5 Summary.....	142
Chapter VII Consistency between Value and Process Models	143
1 Introduction	144
2 Checking Consistency between an e^3 -value Model and an Activity Diagram.....	145
3 Example.....	146
4 Informal Consistency Definition	148
5 Reduced Model.....	149
6 Transformations to Reduced Models.....	150
7 Equivalent Reduced Models	155
8 Plausibility of the Consistency Check	157
9 Granularity of Models: Many-to-Many Relationships between Instances.....	158
10 Related Work	163
11 Summary.....	164
Chapter VIII Validation of the Design Framework	165
1 Introduction	165
2 Real-life Example: Clearing Rights to Make Music Content Public	168
3 Applying Our Method.....	185
4 Developing the Value Perspective	185
5 Developing the Process Perspective	207
6 Checking Consistency	231
7 Evaluating the Design Method	248
8 Summary.....	250
Chapter IX Conclusion	267
1 Contributions	267
2 Positioning	269
3 Limitations.....	271
4 Future Work.....	273
Appendix A e^3 -value Modelling Notation	275
Appendix B UML Activity Diagram Modelling Notation.....	278
Appendix C Goal Modelling Notation.....	279
Appendix D Value Models of Real-life Businesses.....	280
Appendix E Library of Value Patterns.....	300
Appendix F Process Models of Real-life Businesses.....	324
Appendix G Library of Process Patterns	352
References	377

Summary	381
Samenvatting	382

Chapter I

Introduction

This chapter provides an overview of the research published in the thesis. It presents the background of the investigated problems and the motivation of the presented solution approach. Furthermore, it draws the scope of the research and states the research questions. It concludes with an outline of the thesis.

1 Background

The research object of this thesis is the networked businesses and the information systems supporting their day-to-day operations. By networked businesses, we mean linked businesses which deliver value by jointly executing a common business transaction. An example of networked businesses is the constellation of a hotel, a tourist agency, and a bank: they jointly perform the booking and the delivery of a holiday trip.

Networked businesses have existed for a very long time; hence, our interest in their rationale or economic viability is limited. In fact, our research interests are in the information systems applied to improve the performance of businesses. Namely, we aim to explain, categorize, and improve the use of Information and Communication Technology (ICT) in networked businesses.

Our research is motivated by the opening of the Internet for commercial use in the past decade. This has brought many disruptive changes to the business world [8]. Businesses needed to rethink their business models to respond to the opportunities and challenges of a widely available and cheap ICT infrastructure. The considerable changes in value proposition and the need to utilize the Internet technology spawned a new field of research, namely e-business research.

E-business research is divided into two main streams [30]. The first stream aims at conceptualizing the principles that form the foundation of a business. The output of the research explains classical business-science concepts in the light of ICT. The scientific contribution is in models that describe an e-business in general [5, 7] and in terms of structure and governance of transactions [7], customer value, scope, price, revenue sources, connected activities, implementation, capabilities and sustainability [5].

The second stream aims at creating taxonomies of business models of existing businesses in a specific domain [48, 58, 59]. The outcome is a number of criteria based on which business models can be classified into groups with similar characteristics. Example taxonomy based on customer (i) relationship, (ii) data and (iii) transaction is the division to: direct customer, full-service provider, intermediary, whole of enterprise, shared infrastructure, virtual community, value net integrator, and content provider [59]. Another useful taxonomy is the grouping into: brokerage, advertising, infomediary, merchant, manufacturer (direct), affiliate, community, subscription, and utility [48]. With these two examples, we illustrate the focus of the research stream; namely, finding common properties in existing businesses.

We look at the first, conceptualization approach as a top-down one; in the sense that it aims at improving existing theories to account for the importance of information technology in business. Complementary, we regard the second, taxonomy approach as a bottom-up one, which aims at discovering of new abstract models to classify e-businesses. In comparison with these two, our approach is meet-in-the-middle. First, we abstract fragments from innovative models of existing e-businesses. This is a bottom-up approach to create a new library of building block for constructing new business models. Second, we model the requirements for a new business and match these with the available fragments. This is the top-down part of our approach to reuse design knowledge.

2 Research Path

To motivate the interest in constructing new e-business models from existing pieces, we give a brief trace of our quest for interesting research questions. Usually, the personal interests and the path taken determine the scope of one's research. Figure I-1, which is discussed in the next section, draws the boundaries of our research. Here, we use to position our activities in the big picture.

We had a general interest in the impact of information technology on the way business is conducted. We believed that the source of disruption, brought by the ICT, lies in the change of the cost structure for businesses. The availability and low cost of communication infrastructure allowed businesses to organize work in new ways, utilizing new coordination capabilities.

Businesses coordinate work by exchanging objects of value¹; i.e., to deliver a particular product to a client, businesses have to exchange a number of objects, being money, resources, information, etc. Each of these objects has a different value attached to it by the various participants. The overall chain of exchanges determines if there is profit for every business in the network, which is a necessary condition for a network to sustain.

We began modelling networks of businesses that jointly deliver a product to a client by exchanging a number of value objects. We identified a number of businesses (Appendix D Value Models of Real-life Businesses and Appendix F Process Models of Real-life Businesses) which were considered successful, famous, or fit a certain profile. (These appear with number 1 in our research framework on Figure I-1.) To restrict the population, we considered only businesses that offered intermediation services. More specifically, we considered businesses that offered negotiation services because these involve a lot of communication. By negotiation services, we mean: e.g., conflict resolution, wherein a business helps other businesses to reach an agreement over a broken contract; auctioning, wherein a business facilitates the price determination of a scarce resource, or mediation, wherein a business helps other businesses reach an optimal agreement over a contract with many parameters.

We observed that certain combinations of value objects were exchanged in similar transactions in several models. This phenomenon caught our attention and we needed to

¹ This statement is not generally true. Here, we mean a market with anonymous participants which are not related and do not have contractual obligations.

explain why this is happening and what the exchanges are. We concluded that a certain sequence of value transfers happens more often than others because this better resolved a particular business problem or better satisfied a particular need. (This refers to number 3 in Figure I-1.)

Up to this point, we were answering a knowledge question: namely, what are the reoccurring fragments in value models and what are their properties? Having identified a number of such fragments (number 5 in Figure I-1,) our interests shifted to how we can reuse the knowledge captured in each fragment to build new models. This is a design question. We approached the problem aiming at a design framework that would support business analysts in their creative job of developing new business models.

A design framework predefines a number of concepts and constructs to be used in a design process. This includes, but is not restricted to, the number of models to be developed, the relationships between models, and the modelling notations to be used. A framework provides also a method to develop a specification of the system-to-be from initial requirements.

Recalling our main interest in the impact of information and communication technology on businesses, our framework needed additional perspectives to model the communication and coordination aspects carried out mainly by information systems. This added extra knowledge questions, namely: (1) what communication and coordination fragments are and (2) what relationships with value fragments are.

We reverse-engineered the initially selected e-businesses again. This time, we modelled the business processes realizing the value exchanges. (These are depicted with number 2 in Figure I-1.) The models contained both the communication and coordination aspects as they captured the senders of messages, the recipients of messages, the messages them-selves, and the sequence of messages exchanged. We found a number of business process fragments (number 6 in Figure I-1) and we found that their relations with the value fragments were many-to-one. This fact made us unlink the value and process fragments and develop the perspectives independently.

The choice to treat the perspectives separately added a new knowledge question to answer. Following the method in the framework, the designers would develop independently two models of one business (numbers 14 and 16 in Figure I-1.) Hence the question, what consistency criteria the two models should comply with such that the system-to-be is possible to build. We answered providing consistency criteria and a procedure to check these (depicted as number 15 in Figure I-1.)

Going from the back to the front, we have ensured that the developed models will be consistent. The remaining two questions were how to link fragments into one model and, prior to that, how to identify the relevant fragments given the goals of networked businesses. These are two design questions which we approached similarly by providing a set of concepts and rules to apply. In particular, we outlined a sequence of steps for each modelling notation to synthesize a finished model from fragments. (This is shown in Figure I-1 with numbers 12 and 13.) Regarding the selection of relevant fragments, it required more work. To distinguish the fragments that best implement certain requirements, we needed a measure to rate the relevance of fragments. We developed a common representation for requirements and capabilities of fragments (number 8 in Figure I-1.) This allowed us to integrate fragments and requirements and, moreover, to feed and propagate values from fragment capabilities to top-level requirements. The

values assigned to top-requirements are the measure of relevance of a particular fragment for a particular e-business. (The rating of value and process fragments is shown in Figure I-1 with numbers 7 and 9, respectively.)

This section summarized the performed activities that led to this thesis. We proceed with presenting the above-mentioned research questions in a structured manner. But before that, we draw the boundaries of our research interests.

3 Research Scope

Figure I-1 illustrates the proposed approach to reuse design knowledge in the development process of e-business models. The research scope is determined (1) by the domain of the businesses from which the design knowledge is derived and (2) by the steps taken to reapply the design knowledge in new models.

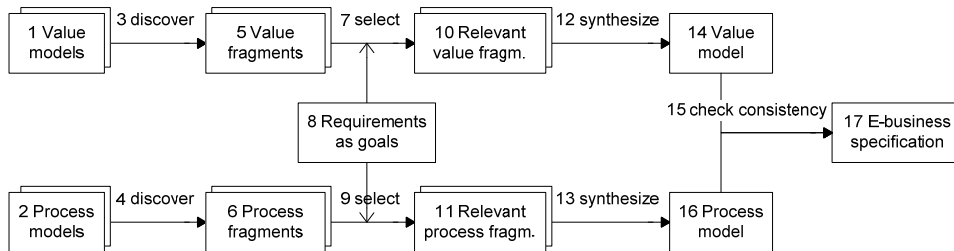


Figure I-1. Approach to reuse design knowledge

Our approach to reuse design knowledge has two phases: preparation and execution. The preparation phase covers the knowledge research to derive design fragments from existing models. This is shown to the left in Figure I-1, elements from 1 to 6; wherein value and process models are depicted with stacked rectangulars, (numbers 1 and 2, respectively.) The arrows coming out of these rectangulars represent the discovery of design fragments, (numbers 3 and 4.) They point to another group of stacked rectangulars depicting the two sets of design fragments, (numbers 5 and 6, respectively.)

The existing models belong to the domain of business intermediaries. More specifically, they belong to e-businesses that offer negotiation services. Naturally, the discovered design fragments belong to the same domain. This restricts our design knowledge to the domain of negotiating intermediaries.

The design knowledge in the value and process fragments is the basis for our design research, the result of which is a design framework. Applying this framework comprises the execution phase of our approach to reuse design knowledge. This is shown to the right in Figure I-1, elements from 7 to 17.

The execution phase starts with modelling the requirements for the e-business under development, (number 8 in Figure I-1.) The requirements are used to select (elements 7 and 9) the relevant fragments for the new design. The relevant fragments are shown in the third column of stacked rectangulars in Figure I-1, (elements 10 and 11.) The outgoing arrows (elements 12 and 13) represent the synthesis step in the development process that leads to one value (element 14) and one process (element 16) models. The last step in the

design framework is the check for consistency (element 15,) which assemble the two models into a specification of an e-business system (element 17) that is possible to build. (The design framework is discussed in detail in Chapter III Design Framework for e-Business Models.)

The usefulness of the design framework is validated with an intermediation business, (see Chapter VIII Validation of the Design Framework.) Nevertheless, we believe that the framework is generally applicable to any type of e-business for which there is a library of design fragments. Our belief is supported by the fact that none of the used modelling notations and methods is dependent on or specific to the domain of intermediation businesses.

4 Research Questions and Approach

We divide the research questions into two types. The first type is the *design* research question. It usually begins with ‘how to’ and asks for a way to achieve a desired output from a given input. It is difficult to give a method which if repeated will lead to the same answer because the answer requires creativity. The approach followed is individual for the research and is based on experience with analogical problems. An answer is valid if, when applied, leads to the desired output.

The second type of research questions is the *knowledge* question. It usually begins with ‘what are’ and asks for a certain output from given input and a method. A knowledge question requires a systematic approach that must lead to reproducible answers. The criteria for validity of the answer are usually part of the method.

We began our research with an intuition that there should be some sort of design knowledge in the models of already existing e-business. Our interest in this knowledge is reuse in future design, which will make the development process cheaper, faster and with better quality. In this context, we wanted to know what the design knowledge is and how we can reuse it. This motivated our two main research questions. The first one is:

- **Q1: What is the design knowledge in existing e-business models?**

This is a knowledge question. We know the input—a number of models of existing e-business. We need to select a method to extract that knowledge, which is a motivation for our approach. The different approaches in the literature lead to design knowledge represented in various forms: e.g., patterns, anti-patterns, lessons-learned, etc. To pick the right approach, we need the answer to our second research question, which in turn depends on the answer to the first research question. To break the cycle, we present the approach we took. In case another approach was taken, the research path, correspondingly results, would have been different.

We aimed at identification of design pattern in value, communication, and coordination models of existing e-business intermediaries that offer negotiation services. This determined the form of the design knowledge. Reformulating our first main research question, it would read: what are the design patterns in value and process models of existing e-businesses? (For the relation between value, communication, and coordination

perspectives and value and process models, look at the answers of research question **Q2.1** below.)

Opting for design patterns gives us an advantage due to pattern properties, namely a description that includes a pair of a problem and a solution.

Having found a number of design patterns, we want to know how to use them. Hence, our second main research question:

- **Q2: How to reuse design knowledge in the development process of an e-business specification?**

This is a design question. In the particular case, we need to find a transformation from design knowledge to a specification of an e-business. Both the input and the output are ambiguous and need further clarification. This motivates two sub-questions:

- **Q2.1: What makes an e-business specification?**

We do not answer this knowledge question in this thesis. In the particular case of e-businesses, we assume the answer is: a consistent set of models representing the value, communication, and coordination perspectives of an e-business. Further in the thesis (see Chapter III Design Framework for e-Business Models, Section 5.3 Relationships between Libraries and Perspectives,) we show that a business process model includes both the communication and coordination perspectives. Therefore, we assume the answer that an e-business specification is consistent value and process models. We define consistency between value and process models in Chapter VII; thus, the remaining unknown is:

- **Q2.1.1: How to check consistency between value and process models?**

This design question has two models as input and a Boolean output. We approach the question by transforming the two models to a third notation and defining a measure in this notation.

Having clarified what the output is for **Q2**, we need to elaborate on the input. We have identified a number of design patterns as part of the answer of research question **Q1** but they may not be applicable. Hence, the second sub-question:

- **Q2.2: What are the relevant design patterns for a particular design?**

This knowledge question has two inputs. The first is a set of design patterns that is delivered by the answer of question **Q1**. The second is the set of requirements for the particular e-business under development. The missing element is the method to apply which motivates our next sub-question:

- **Q2.2.1: How to identify the relevant design patterns given a set of requirements?**

This is a design question. The inputs are design patterns and requirements; the output is a sub-set of the patterns. We approach the question by defining a representation technique for the requirements which allows us to propagate satisfaction values from the

pattern capabilities to top-level requirements. Each combination of patterns is rated against the top-level requirements and the best satisfying combination determines the relevant patterns.

Having clarified the input and the output for design question **Q2**, we can reformulate this to:

- **Q2.3: How to synthesize value and process models from design patterns?**

We approach this design question by providing a procedure to link fragments into complete models.

5 Research Results

Answering the research questions, we compiled:

- **R1: two libraries of value design patterns and process design patterns, correspondingly.**

The patterns are a result of a survey of models of existing e-businesses from the domain of intermediaries offering negotiation services. Nevertheless, the patterns are not only from the negotiation domain: every identified pattern was included in the library. Each pattern is additionally annotated with a capability model which provides a measure of what the pattern can achieve.

The capability models are an application of the second result of this thesis, namely:

- **R2: a goal-modelling technique, including propagation of satisfaction values.**

The technique is used to model both business requirements and pattern capabilities. This property is unique to our approach. It allows us to integrate requirements and capabilities, and propagate satisfaction values throughout the joint model. The propagation technique is the basis for the next contribution:

- **R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements.**

The selection procedure is tightly coupled with the goal-modelling notation for representation of requirements and capabilities. It ensures an exhaustive search for the most relevant patterns is the design space of all possible combinations of available patterns.

The identified patterns need to be linked together into one model. The way to do this is our next contribution, namely:

- **R4: a synthesis approach to derive complete models from patterns.**

The approach outlines a procedure which automates the process of synthesis. It ensures the technical quality of the developed models. More specifically, it guarantees that the synthesised models are valid in the term of the particular modelling notation. With respect to the validity of the specification consisting of the two models, our last research result reads:

- **R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria.**

A consistency check procedure is specific to the notations used to represent the models. With respect to e^3 -value and Activity diagram models, the contribution is the first of its kind.

6 Outline

We structure the content in the following chapters:

Chapter II, Design Knowledge in Existing e-Business Models, defines the kind of design knowledge we are interested in, namely design patterns in economic value and business process models of e-business intermediaries. The chapter begins with our definition of a pattern and then outlines the method we use for pattern discovery. Further, the particular representation form for patterns is given. This includes the annotation with a capability model of each pattern. After that, the setup for the discovery of value and process patterns is described: i.e., details are given on the selected existing e-business models for value and process pattern discovery. At the end, the chapter provides the structure of the libraries that collect the patterns.

Chapter III, Design Framework for e-Business Models, defines our design framework and design method. It begins with the key properties we want to have and an overview of existing frameworks. Then, it gives the motivation behind our choice of perspectives. At the end, the chapter presents the structure of the framework which includes the relationships between concepts of the framework and the sequence of development activities comprising the design method.

Chapter IV, Goal-modelling, presents our goal-modelling technique for representation of business requirements. It provides a definition of goal and, consequently, describes a representation that allows to measure goal satisfaction. Further, the types of relationships among goals are defined with the respective motivation and use of each one of them. Based on the permitted relationships, the chapter gives the rules for constructing goal models. At the end, a mechanism for satisfaction calculation throughout the model is provided.

Chapter V, Selection Procedure, describes our selection procedure for value and process patterns. It begins with an example that is later used as an illustration of the selection procedure. Further, the chapter presents the sequence of steps that makes the procedure. Each step is explained in detail and demonstrated with the example.

Chapter VI, Synthesis of Value and Process Patterns, presents the synthesis procedures for value and process patterns. Both procedures are described in the following structure: first, an example set of patterns is given; second, the synthesis procedure is defined; and third, the procedure is applied on the example set of patterns.

Chapter VII, Consistency between Value and Process Models, discusses the consistency among the models in the perspectives of our design framework. It discusses consistency among the notations used to model the economic value and business process perspectives. Further, the chapter presents the approach to consistency checking; namely, a common notation to which all other models are transformed. In addition to the transformation procedures, the chapter defines consistency between value and process models and provides a consistency criterion in the common notation. The chapter

concludes giving plausibility arguments for the consistency approach. The transformation procedures and the consistency check are illustrated with an example.

Chapter VIII, Validation of the Design Framework, provides arguments in support of the usefulness of the proposed design framework. It starts with a list of desired properties that our design method should exhibit. For each property, a falsifiable claim is provided: if a claim holds then the property is present. The validation of the framework is demonstrated with a real-life example to which the method is applied. After a detailed description of the development steps, the chapter concludes with an analysis of the validation claims and the method properties.

Chapter IX, Conclusion, summarizes the main contributions and lists known limitations. It also gives directions for future research.

The thesis includes several appendixes. These contain details about the modelling notations used throughout the thesis, the collection of models of existing e-businesses used as a source for pattern discovery, and the two libraries of value and process patterns.

7 Summary

The table below relates the research questions, the research results, and thesis chapters. We use it in the beginning of each chapter to position the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	Chapter V Selection Procedure
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

Chapter II

Design Knowledge in Existing e-Business Models

Libraries of Value and Process Patterns

This chapter answers our research question **Q1**. It defines the kind of design knowledge we are interested in, namely design patterns in economic value and business process models of e-business intermediaries. The chapter begins with our definition of a pattern and then outlines the method we use for pattern discovery. Further, the particular representation form for patterns is given. This includes the annotation with a capability model of each pattern. After that, the setup for the discovery of value and process patterns is described: i.e., details are given on the selected existing e-business models for value and process pattern discovery. At the end, the chapter provides the structure of the libraries that collect the patterns¹.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling Chapter V Selection Procedure
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

¹ Part of this chapter is based on Technical Report TR-CTIT-04-26 by Zlatev, Z., Eck van, P. and Wieringa, R. [75]

1 Introduction

We arrived at the idea of reusing fragments of business models after observing¹ repetitive combinations of services on the offer in a number of e-business models. This suggested to us that customers experienced more value if the services were offered in bundles, which (in turn) led us to the analysis of the exchanged values between businesses. In particular, we examined a number of business models of intermediaries and we found reoccurring design fragments.

This chapter explains how we discover and document the knowledge of each design fragment and how we structure the fragment in a library for further use. In analogy with the areas of design patterns [22] in software engineering and Alexander's patterns [6] in building architecture, we call the design fragments patterns. Correspondingly, the library is called library of patterns.

Initially, we modelled a number of businesses from an economic value perspective; i.e., we focused on the exchanged value objects between businesses. We used the *e³-value* modelling notation [25, 27] (Appendix A *e³-value* Modelling Notation) and, correspondingly, we identified *e³-value* fragments which we call value-exchange patterns, or value patterns for short. Following our research interests, we took a second perspective, namely we modelled business processes. This led us to another set of models in which we discovered reoccurring design fragments again (not necessarily corresponding to the value patterns.) We used the Activity diagram modelling notation [45: pages 3-155—3-169] (Appendix B UML Activity Diagram Modelling Notation) and, therefore, our business process patterns are Activity diagram fragments. For short, we refer to business process patterns as process patterns. The two types of patterns constitute two libraries, one for value patterns and one for process patterns.

Both collections of patterns are drawn out of business models of market intermediaries that offer negotiation services. The intermediaries are selected as a result of literature and Internet surveys. This limits the knowledge in the library to business information systems of intermediaries. Nevertheless, the libraries were not intentionally restricted to intermediaries and negotiation. Therefore, patterns that are more general were not excluded from the libraries once spotted in the models.

2 What is a Pattern?

The concept of a pattern entered the vocabulary of software and information systems professionals by analogy with the patterns in buildings construction introduced by Christopher Alexander. He defines it as:

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice [6]."

¹ The observations were made in years 2002 and 2003.

Generalizing from the domain of architecture of buildings, a pattern is a reusable partial design that solves a particular problem. This makes the concept useful in other areas such as: the area of software engineering with design patterns [22], the area of information system architecture with architecture patterns [13], the area of conceptual modelling with analysis patterns [20], and the area of electronic commerce with e-business patterns [4, 31].

A pattern provides an abstract description of a design problem and a general solution to that problem. The problem description explains the problem and the context in which it appears. Further, it describes when the pattern should be applied: the problem may include a list of conditions that must be met before it makes sense to apply the pattern. The solution is template-like and it specifies in a particular notation how to apply the pattern. The description of the pattern includes also the consequences of selecting a pattern. This provides means of evaluating design alternatives and understanding the pros and cons of the application of the pattern.

Our definition of a

- pattern is a recurrent design fragment that solves a problem in a particular context.

Our patterns are identified in two different domains and the types of problem they solve divide them into two groups. The first group originates from business models which focus on the exchanged objects of economic value. We call these models value models. The respective patterns are called value patterns and they are descriptions of constellations of businesses exchanging objects of economic value to achieve their individual goals. The second group originates from business models which focus on the execution order of activities and the data exchanged between participants. We call these models business process models. The respective patterns are called process patterns and they are descriptions of sequences of actions and messages sent to achieve certain goals.

3 On Discovery of Patterns

Patterns are stable configurations of elements and relations in a particular modelling notation. They occur in models at various levels of abstraction or aggregation, in various domains¹ or layers² [64]. People's point of view affects what is interpreted as a pattern and what is not. What is considered a pattern by one can be someone else's primitive building block.

There is not a prescription of how to find patterns. A partial design is proposed as a pattern (1) because it solves a problem that is analogical to a problem from another domain or (2) because somebody observed it occurring many times in connection with similar problems. Once a design fragment is proposed as a pattern, it remains a pattern if other people find it useful.

Buschmann et al. [13: pp. 376-377] use the term pattern mining to describe the process of discovery of a pattern. Their method consists of six steps, namely:

1. Find at least three examples.

¹ Domain is a set of entities that a model is based on

² Layer is a logical grouping of entities in a domain

2. Extract the solution schema.
3. Declare the solution schema to be a 'pattern-candidate'.
4. Run a writer's workshop. (Describe and share the 'pattern-candidate' with colleagues.)
5. Apply the candidate pattern
6. Declare the candidate pattern to be a pattern (if its application is successful.)

These steps show that identifying a pattern is a subjective process. The usefulness of a solution to a particular problem is the criterion to regard a design fragment as a pattern.

In our approach, we are interested in fragments of value and process models that satisfy the following restrictions:

- A fragment must reoccur in several business models;
- Each fragment must reoccur in similar contexts; and
- Fragments must be a solution to a specific problem or achieve specific goals.

The restrictions follow from our definition of a pattern, (see Section 2.)

To identify a pattern, we use a similar approach to the pattern mining presented above. In the case of value models, we look for similar value exchanges occurring in various business models and within similar contexts. (This corresponds to the first step in Buschmann et al.'s approach.) Then, we check whether these model fragments represent a solution to a readily identifiable problem or achieve a business goal in the specific context. (This corresponds to the second step.) We consider such a fragment a pattern as long as its description is understandable to other people and they would agree that the fragment solves a problem. (This corresponds to the remaining steps three, four, five, and six.) The actual application of the identified patterns is limited to the demonstration examples we have built and the real-life example that we use to validate the whole approach. (The real-life example is presented in Chapter VIII Validation of the Design Framework.)

4 Documenting Patterns

The main purpose of patterns is reuse. This supposes that patterns are identified by one person (or group of people) and used many times by another person (or group of people.) To enable the process of reuse, patterns have to be well documented. In the definition of a pattern (see Section 2), we identify the following important properties of a pattern: the problem it solves, the solution itself, and the context in which the problem appears.

4.1 Pattern Description Structures

Usually, patterns are described in a table-like structure. That is, a list of fields each of which specifies a particular type of information. Various sources use different fields to describe patterns. For example, the structure, proposed and used by Gamma et al. [22: pp. 6-8] to document software engineering design patterns, contains among other the following fields: *Pattern name*, *Intent*, *Motivation*, *Applicability*, *Structure*, and *Participants*. In another example from the domain of software architecture patterns, Buschmann et al. [13: pp. 20-21] propose and use a documenting structure that contains

among other the following fields: *Name*, *Context*, *Problem*, *Solution*, and *Structure*. In our last example from the domain of e-business, Adams et al. [4: pp. 26-33] propose and use a structure that contains among other the following fields: *Pattern name*, *Business and IT drivers*, *Context*, *Solution*, and *Benefits*.

The three example structures come from three different domains; nevertheless, all of them cover the problem, solution, and context of a pattern. All three are intended for people. The description is mainly text and, although there are fields with schematic descriptions, these are not intended for machine processing. The structures over-specify the pattern with information interpretable only by people and do not provide machine-readable alternative of this information.

None of the existing structures suits our purpose because we require a specific use of the description, namely automated search and match based on requirements. This renders the three structures not optimal.

4.2 Our Documenting Structure

Our description of a pattern shares most of the fields contained in the example approaches above. We too document the solved problem, the context of the problem, and the solution itself. This information is in a form intended for people. However, we differ in the fields for automated processing. We need to extend the description structure such that we can manipulate the patterns during the development process, such as identify, interpret, and evaluate patterns.

The description structure for value and process patterns is generally the same. The differences are only in the names of the fields such that they match the specific modelling notation or domain terminology. Below, we present one structure and point out the variations per type of pattern. We document patterns using the following structure:

- **Name:** gives a name to the pattern;
- **Code:** gives a code to the pattern;
- **Headline:** gives a short description of the pattern;
- **Context:** describes the context in which the problem solved by the pattern occurs;
- **Description:** provides a detailed description of the pattern;
- **Problem:** describes the problems solved by the pattern;
- **Solution:** presents the value model (for value patterns) or the process model (for process patterns) of the solution to the problems;
- **Capabilities:** presents the capability model associated with the pattern;
- **Occurred in:** lists of models of businesses wherein the pattern was observed.

Below, the details for each field follow:

Name. This field contains the name to the pattern. A well-chosen name is important for an easy reference when people communicate about the pattern.

Code. This field contains the code of the pattern, which is used as a handle to operate with the pattern. Similar to the name, the code is used to identify the pattern when used in automated processing.

Headline. This field gives a short description of the pattern. It is intended for people only. It extends the name of the pattern with more information.

Context. This field describes the context in which the problem solved by the pattern occurs. The field contains textual description of the context. A different format is needed if the context is used in an automated process. In this work, the context is not formalised; thus, the context field is intended for people only.

Description. This field provides a detailed textual description of the pattern. It is intended for people and it contains a summary of all other fields.

Problem. This field describes the problems solved by the pattern. In business networks, businesses participate if their problems are solved. Therefore, it is important to distinguish whose problems are solved. Each business in the model in which the pattern was observed constitutes a role in the pattern. The problems of each role are the forces which drive the businesses. If a pattern resolves all forces then the problems are solved and the network of businesses is stable. Thus to describe better the problems solved by a pattern, we introduce two sub-fields: roles and forces.

- **Roles:** depending on the type of the pattern, a role is the associated behaviour with a business (in the case of value patterns) or with an interaction participant (in the case of process patterns). Roles determine the businesses or participants that (1) participate in the solution and (2) experience the forces in the pattern.
- **Forces:** describe the motivation for a business (in the case of value patterns) or for a participant (in the case of a process pattern) to implement the pattern. The forces capture the goals that each business or participant wants to achieve.

Solution. This field presents the solution that resolves the forces in the problem. It contains both textual and schematic description. The former is intended for people and it contains a brief explanation of the exchanged value objects (for the value patterns) or coordination messages (for the process patterns). The schematic description depends on the type of pattern: a value model describes the value patterns and a process model describes the process patterns. The schematic representation is intended both for people and for automated processing¹.

- **Value model:** is a schematic representation of the solution in an *e³-value* modelling notation (Appendix A *e³-value* Modelling Notation.) Nevertheless, we use the *e³-value* notation with a different meaning of the concepts because the patterns are template-like structures. This means that we use the *e³-value* actors and segments to represent roles played by the actors. We explain the need for this modification and the modification itself in Chapter VI Synthesis of Value and Process Patterns.
- **Process model:** is a schematic representation of the solution in an Activity diagram notation (Appendix B UML Activity Diagram Modelling Notation.) Similarly to the value models above, we use the Activity diagram swimlane to represent a role played by a concrete business, business unit or information

¹ The textual representation in some formal language of the schematic model is directly interpreted by machines.

system. Additionally, we introduce a concept in the Activity diagram notation to represent connection points. We name this *connector* and this is graphically represented by a hollow circle. Connectors model the points where the control flow enters and leaves the pattern. They are similar to the start and end symbols in Activity diagrams but in the scope of a pattern. See for an example Section 7.2. We explain the need for this modification and the modification itself in Chapter VI Synthesis of Value and Process Patterns.

Capabilities. This field describes the capabilities of a pattern. It contains a capability model and an operationalization of the capabilities in the model. These are placed in two sub-fields: capability model and operationalization. Both sub-fields are intended for automated processing and contain information based on which the pattern is selected to be part of a bigger design. We describe the Capabilities field in detail in Section 5.

Occurred in. Lists the models of the real-life business in which the pattern was observed.

4.3 Example of a Pattern

To illustrate our documenting structure for patterns, we give an example with one of our value patterns (Appendix E Library of Value Patterns, pattern 1. Advertising, on page 301.)

The example pattern is called **Advertising**. It was discovered in business models of intermediaries that had access to many clients. The offered intermediation service was free and the revenue was coming from advertising. That is, the intermediary offered content to its clients bundled with commercials of third parties. This information is captured in the following 5 fields of the pattern description structure:

NAME: Advertising

CODE: ADV

HEADLINE: An intermediary offers to broadcast a message to many receivers.

CONTEXT: A business, i.e. an advertiser, has a new or unknown product. It is interested in advertising the product to a great number of potential buyers. An intermediary has access to many clients. Moreover, it has a database with client information.

DESCRIPTION: An Intermediary offers a product¹ to a number of clients for free: i.e., the clients do not have to pay money to consume the product. What the intermediary gets in return is the attention of the clients. The intermediary makes money by bundling its product with an advertisement of a third party seeking exposure to potential clients. The Intermediary is able to target the advertisements as it has information about its clients' preferences.

The analysis of the business models in which the pattern occurred showed that there are three participating businesses. From this, we abstracted the three business roles in the

¹ The term product includes also services.

pattern. Furthermore, we identified one force per role driving a particular business to exchange values as in the pattern. The roles and the forces are documented in the *Problem* description field below:

PROBLEM:

- roles: The pattern includes three roles played by three different businesses:
- *Client* is the role played by the consumers of products provided by the *Intermediary*;
 - *Intermediary* is the role played by the business that provides a product bundled with an advertisement; and
 - *Advertiser* is the role played by the business seeking publicity.

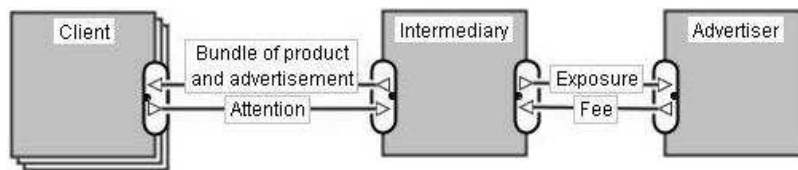
forces:

Role	Force	
Client	F1	Consume products for free
Intermediary	F2	Profit from access to and knowledge about clients
Advertiser	F3	Reach potential clients

The observed reoccurring fragment models the resolution of the forces driving the businesses to such an interaction. It is the solution offered by the pattern. For this, the documenting structure gives a textual and schematic description. In the particular case of a value pattern, the schematic description is in the *e³-value* notation (Appendix A *e³-value* Modelling Notation.) The *Solution* field follows:

SOLUTION: An intermediary broadcasts targeted advertisements to its clients in return for a fee.

value model:



The last field of the description structure lists the models in which the pattern was observed. (The list of all models is in Appendix D Value Models of Real-life Businesses.) The field looks such as:

OCCURRED IN:

- eBay
- PriceLine
- MySimon

The example above shows all description fields but the *Capabilities* field. For an example of this, look in the next section.

5 Annotating Patterns with Capability Models

The distinctive field in our pattern description is the *Capabilities* field. Although, this is present in other descriptions, its unique property is that it is intended for automated processing. The field contains a capability model and an operationalization of the capabilities in this model.

To understand the function and importance of the *Capabilities* field, we need to preview what a goal model is. (The full description of goal models and goal modelling is given in Chapter IV Goal-modelling.) A goal model is a graph structure, the nodes of which represent desired phenomena. Each node is a goal operationalized with a variable and an evaluation function. The value of the variable indicates whether the desired phenomena exist or not; the evaluation function determines with what value the goal is achieved.

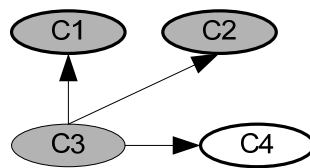
A goal model is used to prescribe what is desired to happen. By specifying an evaluation function for each goal in the model, we model what is required; thus, the goal model represents requirements. Another way to use a goal model is to describe what is already present. By giving values to goal variables, we model the capabilities of existing phenomena. Thus, a goal model with values is a capability model, and describes what a given solution can do.

A capability model, attached to a pattern, represents what a pattern can achieve. The operationalization of the capabilities contains the names of the variables representing the capabilities and the values of each variable. Below, we show an example of a *Capabilities* field of an example. (The full pattern description is available in Appendix E Library of Value Patterns, pattern 2. Client Connection, on page 303.)

The *Capabilities* field contains two sub-fields: *capability model* and *operationalization*. The former features a schematic representation of the capabilities and the relations among them. The notation is summarised in Appendix C Goal Modelling Notation; the full description of goal modelling is given in Chapter IV Goal-modelling. Moreover, it contains a table that links capabilities with roles and forces from the problem description. This information is needed for the synthesis of patterns. The latter, the operationalization field, presents in a table the capabilities the variable names and the values associated with each variable. This information is used for the selection of patterns.

CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Keep existing clients	Intermediary	F1

C2	Influence clients decisions	Intermediary	F2
C3	Provide clients with information about itself, products, services, and market analysis	Intermediary	
C4	Receive additional value	Client	F3

operationalization:

<i>Capability code</i>	<i>Capability</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Keep existing clients		
a1	Returning new clients	Percentage per month	76
C2	Influence clients decisions		
a2	Follow-ups from newsletters	Percentage sent newsletters	3
C3	Provide clients with information about itself, products, services, and market analysis		
a3	Newsletters	Natural numbers / per month	2
C4	Receive additional value		
a4	Perceived value	{spam, no value, useful}	Useful

6 Discovery of Value Patterns

We looked for value patterns in a set of e-business value models. Our selection of businesses was made based on the negotiation services they offered. We aimed at covering all types of negotiation services. According to the research on devising taxonomies of e-business models [48, 58, 59] (as we discuss in Chapter I, Section 1 Background), our business models are of intermediation type. In this section, we further distinguish between types of intermediary based on the type of negotiation service they offer. We use the following six groups:

- **Conflict resolution intermediary.** Due to many reasons, businesses disagree and break their relationships. Conflict resolution intermediaries exist to help them reach a settlement. They offer independent mediation to overcome the difficulties in communication caused by mistrust. An example of such a business is SquareTrade (Appendix D Value Models of Real-life Businesses, model 13. SquareTrade - eBay business model, on page 293.) SquareTrade offers eBay users to resolve their conflict through exchange of messages. If resolution is reached, SquareTrade removes negative feedbacks from the eBay database.

- **Negotiation support system.** Economic theories see businesses as rational agents which seek maximum benefits out of every interaction. In a real-life situation, businesses have opposing goals and optimal agreements are difficult to reach. To reduce the complexity, intermediaries offer negotiation support services. These services aid businesses in taking informed decisions in the process of negotiation. An example of such a business is SmartSettle (Appendix D Value Models of Real-life Businesses, model 10. SmartSettle, on page 290.) Negotiating businesses submit proposals and a facilitator, representing SmartSettle, works individually with all parties to reach on optimal agreement.
- **Auctioneer.** Auctions are market-matching mechanisms applied when there are many businesses interested in a product but only one supplier. Auctions are best applied on products with only one attribute, mainly price. They follow different protocols, e.g. with increment bids, with decrement bids, or with sealed bids. Auctions determine the best price for a business that delivers a scarce resource. (Auctioneers can apply different mechanisms for a revenue generation. For example, fees can be imposed to buyers, fees can be imposed to sellers, or advertisers can pay for promotions. In the classification, these differences are not taken into account.) An example of such a business is eBay (Appendix D Value Models of Real-life Businesses, model 2. eBay, on page 282.) eBay offers to setup an action for the product of a seller. The buyers, also users of the eBay services, bid for the product. eBay participates in the transaction only to determine the price.
- **Exchange.** An exchange is an organization, association, or group, which provides or maintains a marketplace where securities, options, futures, or commodities are traded. (Barter is also a kind of exchange, where goods or services of approximately equal value are provided and received.) An exchange can be seen as a double continuous auction at which buyers and sellers submit buying and selling bids. An example of such a business is SellXS (Appendix D Value Models of Real-life Businesses, model 9. SellXS, on page 289.) SellXS auctions excess inventory in the semiconductor industry. It is an exchange because the buyers in one transaction can be the seller in another.
- **Price discoverer.** A price discoverer is an intermediary that searches for products based on price. A price discoverer can employ different mechanisms in its searches. Examples of such searches are: an intermediary searches the Web pages of a number of retailers; an intermediary searches its own database with products and prices; or an intermediary negotiates with suppliers from its contact database. An example of such a business is PriceLine (Appendix D Value Models of Real-life Businesses, model 8. PriceLine, on page 288.) PriceLine asks clients to name a price for a product and then searches for the product at this price. If the product is available, a transaction occurs at the price proposed by the client.
- **Price comparator.** A price comparator is an intermediary that searches for products and presents these sorted according to price. A price comparator has a database of registered products and their relatively up-to-date prices. A price comparator does not involve in the later stages of the transaction. It only connects buyers with a seller chosen by them; the rest of the transaction is

executed without the price comparator. An example of such a business is MySimon (Appendix D Value Models of Real-life Businesses, model, 6. MySimon, on page 286.) MySimon compares offerings by different providers based on a client specification of a product. It does not sell or ship.

We consider the selected groups of businesses covering the spectrum of intermediaries that offer negotiation services. Moreover, we consider them focused on the domain of negotiation such that we can find reoccurring design fragments.

The instances of selected business models are provided in an executive summary in Appendix D Value Models of Real-life Businesses. Each intermediary is documented with references to the Web page of the intermediary or other relevant source of information, a concise textual description, and an e^3 -value model.

6.1 Classification of Selected Business Value Models

We selected 13 intermediaries that offer negotiation services. From available information on the Web sites of the intermediaries or published case descriptions, we reverse-engineered 19 value models. These are organized in the 6 groups listed above according to the type of negotiation service they offer. The type is determined by the core business activity of the intermediary. Figure II-1 shows the classification. The negotiation intermediaries are divided into conflict resolution intermediaries, intermediaries that offer negotiation support (negotiation support systems in the figure), auctioneers, exchanges, price discoverers, and price comparators. In each category, we show the classified real-life businesses: the boxes with underlined names are the real-life businesses.

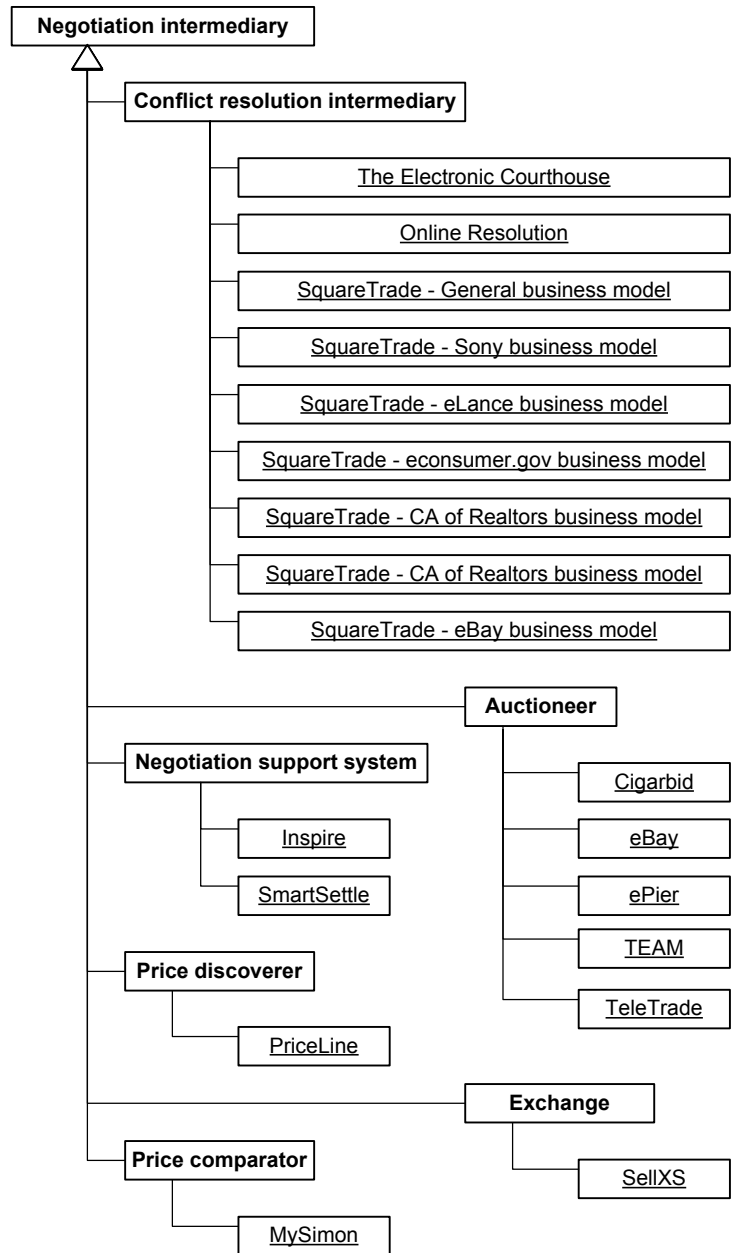


Figure II-1. Classification of real-life businesses as a source for discovery of value patterns

The selected real-life businesses presented in Figure II-1 are a sub-set of the collection of businesses we reverse-engineered. We believe they exemplify best the sources of the patterns documented in the library of value patterns.

6.2 Discovered Value Patterns

As we discuss in Section 3, to discover patterns we look for similar value exchanges in various business models. We ensure that the exchanges occur in a similar business context and that the e^3 -value fragments represent a solution to a problem. Following this informal process, we have identified 11 e^3 -value fragments that we consider patterns. Using the pattern document structure presented in Section 4, we present the identified patterns in Appendix E Library of Value Patterns.

The values assigned to capability variables are only an example. Their purpose is to demonstrate the applicability of the approach in general. A more rigorous approach is required to determine the exact values. We address the issue in the discussion of the limitations of our work (Chapter IX Conclusion, Section 3.1 Contingent limitations.)

7 Discovery of Process Patterns

We looked for process patterns in a set of business process models. We selected the businesses based on the same criteria as the ones we used for value patterns in Section 6. In sequel, the classification groups of intermediaries that offer negotiation services are the same, namely: Conflict resolution intermediaries, Intermediaries that offer decision support, Auctioneers, Exchanges, Price discoverers, and Price comparators. We consider the selected groups of businesses covering the spectrum of intermediaries that offer negotiation services. Moreover, we consider them focused on the domain of negotiation such that we can find reoccurring design fragments.

The instances of selected business process models are provided in an executive summary in Appendix F Process Models of Real-life Businesses. Each intermediary is documented with references to the Web page of the intermediary or other relevant source of information, a concise textual description, and one or more Activity diagrams.

7.1 Classification of Selected Business Process Models

We selected 12 intermediaries that offer negotiation services. From available information on the Web sites of the intermediaries or published case descriptions, we reverse engineered 12 business process models. The intermediaries are organized in 6 groups according to the type of negotiation service they offer. The groups are the same as the groups for the business models for value pattern discovery. Figure II-2 shows the classification. In each category, we show the classified real-life businesses: the boxes with underlined names are the real-life businesses.

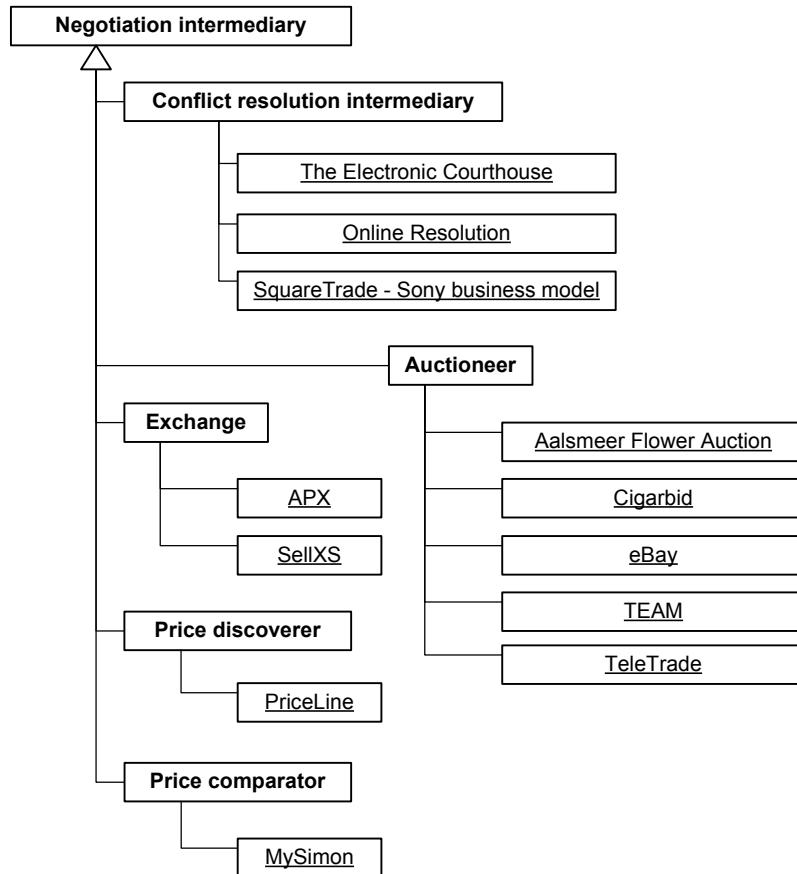


Figure II-2. Classification of real-life businesses as a source for discovery of process patterns

The selected real-life businesses presented in Figure II-2 are a sub-set of the collection of businesses we reverse-engineered. We believe they exemplify best the sources of the patterns documented in the library of process patterns.

7.2 Discovered Process Patterns

To discover patterns, we look for similar message exchanges and sequences of activities in various business models. We ensure that the fragments occur in similar contexts and that these represent a solution to a problem. Following this informal process, we have identified 12 Activity diagram fragments that we consider patterns. Using the pattern document structure presented in Section 4, we present the identified patterns in Appendix G Library of Process Patterns.

The values assigned to capability variables are only an example. Their purpose is to demonstrate the applicability of the approach in general. A more rigorous approach is

requires to determine the exact values. We address the issue in the discussion of the limitations of our work (Chapter IX Conclusion, Section 3.1 Contingent limitations.)

8 Relation between Value and Process Patterns

Value and process patterns originate from the models of one set of e-businesses¹. This enabled us to observe that value and process patterns are related with one-to-many relationships. This means that one value pattern can be realized individually by several different process patterns. Furthermore, the process patterns are generally finer grained than the value patterns. This means that several process patterns make one value pattern.

For the purpose of simplicity, we chose to abstract the information about relationships between patterns. We work with value and process patterns as with unrelated entities.

9 Structure of the Library

The library of patterns is a list of pattern descriptions. In its current release, it does not support links between related patterns, which is the reason to lack grouping of patterns into bigger patterns.

Adding and removing patterns is limited to including a description of a pattern into the library and, respectively, excluding a description.

In its current version, the library does not consider scalability issues. We do not expect growth in the number of patterns such that the search in the library is slowed down. In case this occurs, we believe that a suitable classification of the patterns will reduce the search time.

10 Related work

There are several schools of thought interested in business patterns, where business patterns exceed the domain of information systems and object-orientation. A business pattern is a construct that connects a certain problem-solving goal and a potential solution [4, page 14]. Below, we compare several approaches with ours.

IBM Patterns for e-business [4, 31] prescribes a structure of patterns in several levels. At the top level, there are four business patterns and two integration patterns. These patterns define the high-level structure of a business. They can be composed together to form composite patterns and they can be further instantiated by application patterns. In contrast with our approach, the IBM Patterns for e-business are a fixed number with complex relationships among them. They are presented in a single framework and their composition comes naturally. They look ‘designed’ for composition. In contrast, our

¹ This is true with a few exceptions

libraries are open for adding new patterns and do not restrict the possible compositions of patterns.

The IBM patterns for e-business are described in a diagramming technique which has the following constructs: participant, functional component, data storage, and links. These diagrams express synchronous or asynchronous communication among business entities and blocks of purposefully grouped functionality. This approach covers our communication perspective as it expresses a static view on the system components. The difference is the level of granularity: e.g., the entities participant and the functional component are coarser grained than the components in our approach.

Weill and Vitale [59] take another approach to business models. In their work, they identify a number of atomic business models. These are not specified explicitly as patterns; nevertheless, their descriptions contain sections for objectives and critical success factors which translate to goals and driving forces concepts from the object-oriented design patterns. Thus, we consider them patterns. The atomic business models emphasize the incompatibility of certain models, rather than the possible compositions of patterns.

A distinguishing property of the atomic business models is that they explicitly specify the value proposition between participants. The modelling technique includes concepts such as: money flow, product flow, information flow, participant's role, and relationships. Therefore, we classify the atomic models as similar to our economic value patterns. The difference is again in the level of granularity; the atomic models are coarser grained compared to our value patterns.

RosettaNet [51] standardizes Partner Interface Processes (PIPs) which are fragments of a process model which can be composed together to make one complete model. The collection contains more than 100 fragments at different stages of standardization. The library has a complex structure of clusters and segments, which groups the fragments according to the type of supported processes and the type of transactions within the process.

The PIPs share some similarities with our patterns, namely some of our pattern can be realized by a combination of PIPs. For example, the Payment process pattern (Appendix G Library of Process Patterns, Pattern 8. Payment, on page 367) is partially realized by PIP 3C3 or PIP 3C5 (see the online RosettaNet documentation [51].) Eventual transfer of PIPs as patterns in our process library would require their annotation with capability models such that they are identifiable by our selection procedure.

From a workflow perspective, patterns also exist. Wil van der Aalst and colleagues [2, 3] have developed a library of process patterns. These patterns are indicated as requirements for workflow languages. With respect to that, they are at a lower level of granularity and it is difficult to relate them to a goal expressed in business terms.

11 Summary

This chapter, Appendix E and Appendix G answered our research question **Q1: What is the design knowledge in existing e-business models?** It restricted the design knowledge we are interested in to design patterns in economic value and business process models. The chapter gave a discovery procedure for patterns and a structure to document

the patterns. Applying the discovery procedure to 19 value and 12 process models resulted in two libraries of 11 value patterns and 12 process patterns.

Chapter III

Design Framework for e-Business Models

This chapter elaborates on research questions **Q2** and **Q2.1**. It defines our design framework and design method. It begins with the key properties we want to have and an overview of existing frameworks. Then, it gives the motivation behind our choice of perspectives. Further, the chapter presents the structure of the framework. This includes the relationships between concepts of the framework and the sequence of development activities comprising the design method¹.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	Chapter V Selection Procedure
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

1 Introduction

Restating our research interest, we want to reuse design knowledge captured in patterns into the development process of new e-business models. The focus of the new specification is the use of ICT to support the day-to-day business operations. In Chapter II Design Knowledge in Existing e-Business Models, we presented the knowledge we

¹ This chapter is based on previous work by Zlatev, Z., Daneva, M. and Wieringa [74]

have documented, namely two libraries of value-exchange and business process patterns. In this chapter, we outline an approach to reuse these patterns.

We propose a framework that helps business analysts evaluate requirements for e-business systems. The framework defines a number of concepts to help structure the problem domain. In particular, it prescribes a number of perspectives to look at the e-business system, a requirement modelling technique, and relations with the available design patterns. In addition, we define a design method which instructs the business analyst how to use the concepts defined by the framework. The method gives a number of steps to follow designing a new specification from patterns. Throughout this thesis, we use the term framework implicitly assuming that there is a design method defined.

The composition of the framework and method are a result of creativity, experience in developing specifications, and study of similar design frameworks.

2 Motivation

Businesses, in general, are under constant pressure to innovate and adapt to the changing environment, while at the same time reuse as much as possible of their past investments. The electronic intermediaries that we investigate in Chapter II, Design Knowledge in Existing e-Business Models, are no exception. The need for a change is even more frequent as it follows the emergence of new information and communication technologies. Businesses need to respond fast in order to keep or create competitive advantages [29].

The identified patterns in Chapter II, Design Knowledge in Existing e-Business Models, are design decisions that solve particular problems that are tested in practice. They contain knowledge that is important to business analysts who want to reuse existing solutions. By means of patterns, a framework speeds up the development process and reduces costs [31]. For these reasons, businesses need development methods that utilize existing design knowledge. Hence, the design framework should allow:

- Reuse of existing design knowledge: the framework should have a notion of design patterns.

New business models succeed if they create value for the customers [46] rather than adopting the latest available technology. New designs should be driven by market opportunities and not by a technology push. Business analysts' main concern is achievement of new business goals by the new design. A design framework has to evaluate the viability of business goals. Thus, the framework has to be:

- Goal-oriented: the framework should capture the goals to be achieved by the system under development and provide means of evaluating the satisfaction of these goals by particular design decisions.

The specification of a business system is a complex process. Many stakeholders with distinctive interests are involved in the alignment of information system capabilities and business objectives. To manage the complexity of such a process, the responsibilities for design and analysis of distinctive aspects of the system should be separated. Therefore, we require from our design framework to be:

- **Multi-perspective:** the framework has to separate concerns in individual perspectives such that there is sufficient information to evaluate the requirements and at the same time minimize redundant information.

The independent development of perspectives introduces freedom that may result in an inconsistent final set of models. Management of inconsistencies, caused by stakeholders' conflicting goals and partial specifications, increases the design freedom and enables innovative solutions [17]. Therefore, our framework has to tolerate:

- **Late consistency check:** the framework needs to tolerate inconsistency by postponing resolution of conflicting specifications.

We consider the four listed above properties of a design framework essential to support business analysts in their work of analysis and evaluating new business models.

3 Terminology

There is a lot of literature [19, 25, 32, 38, 50, 53, 57, 72, 76] about design frameworks and requirements engineering approaches. Each proposal uses its own terminology. Below, we define the way we use the terms and use that terminology later in the discussion of related work.

Pattern. Pattern is a recurrent design fragment that solves a problem in a particular context (copied from Chapter II Design Knowledge in Existing e-Business Models, Section 2 What is a Pattern?, on page 12.) Its specific properties are persistent throughout its use in different systems.

Goal. Goal is a desired state of reality that a certain organization or individual wants to achieve (taken from Chapter IV Goal-modelling, Section 1 Introduction, on page 44.) Goals express stakeholders' desired functionality and quality from a system under development.

Perspective. (also called viewpoint [19, 44]) Perspective is a restricted in expressiveness standpoint to model a system. The resulting partial specification is limited by: (i) stakeholders' need of information, (ii) domain knowledge, (iii) design methodology, (iv) knowledge representation technique, and (v) a set of system's properties.

Notation. Notation is a particular modelling language chosen to represent the system under development from a particular perspective.

Model. Model is a partial specification [19] of the system developed within one perspective.

Specification. Specification is the collection of models that is considered to fully specify the system under development.

Consistency. A specification is consistent if it is possible to find an example of a system that conforms to all models in the specification [32]. In our particular case, a specification

is consistent if it is possible to build at least one system that implements correctly all models.

4 Review of Existing Frameworks

The purpose of this review of design approaches to business information systems is to find an approach that has the required properties listed in Section 2 Motivation. If none found, the review will help us compose a new framework.

We use the term framework to refer to all surveyed approaches including frameworks, reference models, design methodologies, modelling techniques, etc.

Various approaches have been proposed [19, 25, 32, 38, 50, 53, 57, 72, 76]. Each of them has a different focus of analysis and properties. In this section, we investigate the approaches that in our opinion are applicable to (re)design of business information systems.

4.1 Reviewed Approaches

In this section, we provide a short description of several approaches to information system analysis and design. We have selected ArchiMate, ViewPoints, Reference Model for Open Distributed Processing (RM-ODP), Dynamic Essential Modelling of Organisation (DEMO), and Multi Perspective Enterprise MOdelling (MEMO). We believe that these represent a wide spectrum of approaches and are representative examples.

ArchiMate. ArchiMate [57] is an integration framework for architecture models from different domains. It aims at developing a common architecture language which will allow different stakeholders to express their requirements. ArchiMate builds one meta-language by combining existing languages used to model information systems. In its version from January 2005, it defines 15 overlapping perspectives. The consistency is achieved by a two-way translation from every notation to the ArchiMate language; every modelling notation is wrapped with a mapping layer.

ViewPoints. ViewPoints [19, 44] is a meta-framework. It structures loosely coupled, locally managed, distributable objects encapsulating partial representation knowledge, development process knowledge and specification knowledge, about a system and its domain. These objects, together with the standpoint of a stakeholder, form viewpoints. A viewpoint forms a template for the generation of views. It does not restrict the selection of abstraction, notation, methodology, documentation, and consistency relationships. The consistency issue is left outside the framework.

RM-ODP. RM-ODP [32] is a framework for architecture specification of large distributed systems. It is a standard that aims at providing support for inter-working, interoperability and distribution, and therefore to enable the building of integrated, manageable, heterogeneous systems. The specification of a system in terms of RM-ODP has 5 separate but interrelated perspectives: Enterprise, Information, Computational,

Engineering, and Technology. With respect to consistency, the framework specifies an approach of translation of models from one representation to another. The inconsistency detection is in a form of conformance assessment. Such assessment is done at pre-determined points in two models. If two implementations of two models have the same observed reactions to particular stimuli then they are conformant.

DEMO. DEMO [50] is a methodology aimed at providing constructional¹ descriptions of business systems. It assumes as basic building blocks the communicative acts exchanged in a conversation. The modelling methodology begins with identification of all transactions that occur between the involved businesses. Derived from these, the methodology prescribes 6 perspectives: the Interaction model, the Business Process model, the Transaction Process model, the Action model, the Interstriction model, and the Fact model. The methodology does not discuss inconsistency as everything is derived from one set of transactions.

MEMO. MEMO [40] offers a framework to structure an enterprise and to support the development of enterprise models. The framework proposes 3 main layers of an enterprise: strategic, organizational, and information system. The layers are split into 4 perspectives, namely: resource, structure, process, and goal. For instance, business process models would be assigned to layer *organization* and the perspective *process*. MEMO defines a multi-layer language architecture to integrate the modelling languages and reduce redundancy between the perspectives. The consistency between different models is achieved through the meta-meta-model and pre-defined relationships between layers and perspectives.

4.2 Desired Properties and Existing Approaches

In this section, we evaluate the above listed approaches, except ViewPoints, as it is a template for frameworks. The evaluation is against the desired properties from Section 2 Motivation. Below, we list the properties together with the conditions for properties to hold true:

- Reuse of existing design knowledge. We consider a framework to reuse design knowledge if it utilizes knowledge contained in design patterns.
- Goal-oriented. We consider a framework goal-oriented if the developed specifications are evaluated against the goals the system under development has to achieve.
- Multi-perspective. We consider a framework multi-perspective if it has more than one perspective.
- Late consistency check. A framework has late consistency check if it allows inconsistencies, manages inconsistencies, and checks for consistency.

Table III-1 presents the selected frameworks and their scores with respect to the evaluation criteria. (The columns of the table present the frameworks; whereas the rows

¹ Constructional is used as complement of functional, the same way white-box and black-box descriptions are used

contain the evaluation criteria.) Our evaluation of the results from Table III-1 shows that none of the frameworks meets all of the required properties.

With respect to reuse of design knowledge, none of the frameworks has an intrinsic notion of design patterns. Neither of the frameworks evaluates the developed models against a goal model. MEMO has a perspective to model goals; however, this is independent of other perspectives. To the contrary, all frameworks are multi-perspective. With respect to handling inconsistencies, the majority of approaches do not allow inconsistencies as they develop proprietary common language from which all notations are derived. Only RM-ODP tolerates inconsistencies and offers means of checking consistency.

Table III-1: Comparison of approaches toward requirements specification

	ArchiMate	RM-ODP	DEMO	MEMO
Reuse of existing design knowledge	No	No	No	No
Goal-oriented	No	No	No	No
Multi-perspective	15	5	6	4
Late consistency check	No	Yes	No	No

The results of the comparison show that none of the presented frameworks has all the properties we require, especially the reuse of design knowledge and goal-orientation.

4.3 Lessons Learned

1. The reviewed frameworks do not consider patterns. None of the approaches defines pattern as a concept. We need to use our creativity to fit the design knowledge in the form of patterns into a framework.
2. In the reviewed frameworks, goals, if modeled, are part of the specification, not means of evaluating the specification. Only MEMO models goals and these are encapsulated in a perspective. The rest of the models do not depend on the goal model. In our framework, we should give the representation of goals a special status, not keep this in a perspective.
3. A standard set of perspectives does not exist in the reviewed frameworks. Every approach defines its own perspectives. Depending on the objective of the framework, these may overlap, be disjoint, or be derived from one-another. We are going to select perspectives that we can build with the available design knowledge.
4. Inconsistencies are not common in the reviewed frameworks. Only RM-ODP allows inconsistencies because it allows a free choice of modelling notation per perspective. We are taking the RM-ODP approach of defining consistency criteria and keeping the framework notation independent.

We can use ViewPoints as a skeleton and further customize it. We can take such perspective that we model the economic value exchanges, the coordination, and the communication in an e-business. Further, we can select modelling notations per perspective and define consistency relationship between models in these particular notations. Finally, we can define a design method within each perspective to (1) reuse

patterns and (2) evaluate the final specification with respect to the goal model of the desired system. In the next section, we present how we do that.

5 Objective and Available Resources

5.1 Objective

Our primary objective is to construct a framework for reuse of existing design knowledge into new models of e-businesses. Additionally, the framework has to have the four properties listed in Section 2 Motivation, namely (1) to reuse existing design knowledge, (2) to be goal-oriented, (3) to be multi-perspective, and (4) to have late consistency check.

Applying the framework should lead to:

- a consistent specification of a business information system,

where business information system is the organizational entities, people, procedures and software applications that jointly execute a particular business process to deliver value to clients.

The specification of such a system is a complex process. Many stakeholders with distinctive interests are involved in the alignment of information system capabilities and business objectives. A standard approach to manage the complexity of such a process is the adoption of a multi-perspective development method, where responsibilities for design and analysis of distinctive aspects of the system are localized in separate perspectives. The perspectives in the framework must be expressive and minimal: i.e., provide sufficient information to evaluate the new system requirements and, at the same time minimal, avoid unnecessary details and redundant information.

The definition of a business information system refers to software applications, business processes, people, organizational entities, activities, value objects, and clients. Our literature study led us to believe that the following three perspectives include all concepts from the definition and are conceptually distinct. The first perspective is the

- *economic value* perspective, in which we model the creation of value among networked businesses and analyze the incentives for them to take part in such a network.

The second perspective is the

- *coordination* perspective, in which we model the order of activities realizing the exchanges of economic values.

And the third perspective is the

- *communication* perspective, in which we model the data exchange among the components of the information system that supports the business.

5.2 Available Resources

The resources we have are two libraries of patterns. (Chapter II Design Knowledge in Existing e-Business Models describes the structure of patterns and libraries.) The first library contains patterns observed in value exchange models; the second contains patterns

from business process models. The patterns are design fragment represented in particular modelling notation: the value patterns are e^3 -value fragments and the process patterns are Activity diagram fragments.

5.3 Relationships between Libraries and Perspectives

The e^3 -value notation models the aspects of system looked at from an economic value perspective. Therefore, the library of value patterns is the resource to use in the economic value perspective.

The coordination perspective focuses on the sequence of activities performed by applications, individuals, or organizational units: it manages dependencies among activities. The important information in the model is the control flow among various actions executed in a business process. Exchanged messages are considered to be only triggering events for the receiving activity; their semantics is not interpreted.

The communication perspectives models the data exchanged among applications, processes, individuals, and organizational units: i.e., it models the data flow. The emphasis is on the semantics of messages and the senders and receivers of these messages, while the order of message exchanges is not interpreted.

The library of process patterns is derived from Activity diagrams. The Activity diagram notation covers more than the control flow. It can express information about who or what executes an activity, who or what sends a message, what is the semantics of a message, and how the execution is affected by the semantics of a message.

Summing up, the coordination and communication perspectives are both contained in the library of process patterns. This means that the two perspectives can be merged and a single new perspective, called *business process* perspective, based on the library of process patterns.

6 The Framework

6.1 Structure of the Framework

The available resources are two libraries of value and process patterns. The framework has to be able to develop a specification of a business information system, consisting of models from economic value and business process perspectives. The desired properties of the new system are given in a goal model.

Figure III-1 shows the big picture. The rectangle at the top depicts the desired system. At the bottom of the figure, we see the two pattern libraries and the goal model that **defines** the system under development. The results of two perspectives are shown in the middle of Figure III-1. The model to the left **prescribes** the system from an economic value perspective, using an e^3 -value modelling notation. The model to the right **prescribes** the system from a business process perspective, using an Activity diagram notation. Both models **use** patterns from the libraries to **implement** the goal model. The value and process models are related with a **consistency** relationship, which guarantees that the two models prescribe a system that can be built.

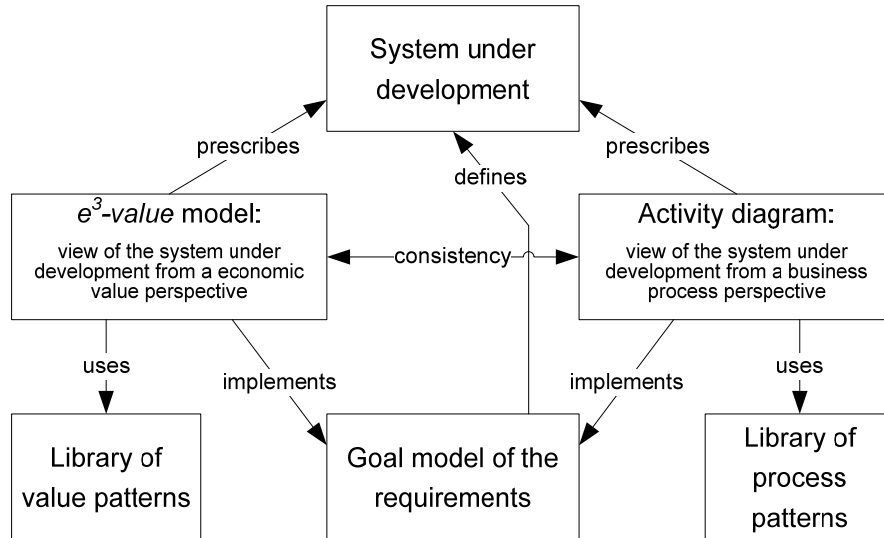


Figure III-1. Framework structure

The requirements of the system under development are captured in only one goal model. This does not exclude conflicting requirements: i.e., the partial satisfaction of such requirements requires trade-offs. However, the goal model is an independently validated domain theory. This means that the requirements, potentially conflicting, are consistent.

The implementation of the requirements in each perspective involves manual steps (Section 6.4.) Developed in such a way, the models are not necessarily a correct implementation of the goal model. Taking more perspectives (analogy with triangulation) increases the confidence that the final specification is a correct implementation of the requirements.

If value and process models are not consistent, the reasons could be: a mistake in a manual development step; a missing pattern in one of the libraries; or an error in the goal model. This means that inconsistent models could lead to improving the goal models but not in all cases the goal model is wrong. Correspondingly, consistent value and process models do not increase our confidence in the correctness of the goal model because we accept the goal model to be independently validated prior to the application of the framework and method.

The detailed description of the libraries of patterns, the goal-modelling of requirements, and the consistency between an e^3 -value model and an Activity diagram are given in Chapter II Design Knowledge in Existing e-Business Models, Chapter IV Goal-modelling, and Chapter VII Consistency between Value and Process Models, respectively.

6.2 Structure of a Library

Each pattern is annotated with a capability model. The capability models are the search keys in the library: a pattern is selected as relevant for a particular design based on its capabilities.

Further, we refer to patterns as in Figure III-2. At the top, the graph-like figure represents the capability model of the pattern. At the bottom, the lego-like figure represents the design fragment. The capability model and the fragment constitute the pattern.

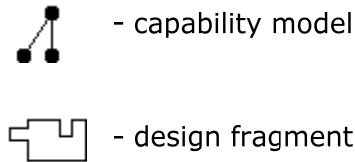


Figure III-2. An abstract representation of a pattern.

Figure III-3 represents a library of patterns. From left to right, five patterns, P1 to P5, are shown with their capability models and design fragments.

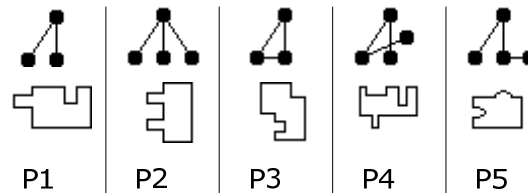


Figure III-3. A library of five patterns: P1 to P5.

The abstract form of the library hides the specificities of each design fragment. We use the representation in Figure III-3 for an explanation purpose only. We consider that the actual two libraries contain information about value exchanges and business processes.

6.3 Structure of a Single Perspective

This section presents the structure of a single perspective. Figure III-4 shows the components of a perspective and the relationships between these. A framework of a perspective consists of:

- a goal model – a model that defines what the system under development has to achieve;
- capability models – models that describe what design fragments can do;
- design fragments – partial designs exhibiting certain distinctive capabilities; and
- views of the system under development – models of the system under development prescribing a system that should achieve the goals in the goal model.

The system under development, to the right in Figure III-4, is not part of the perspective. It is shown to help relating the structure of the framework depicted in Figure III-1 and the structure of a perspective.

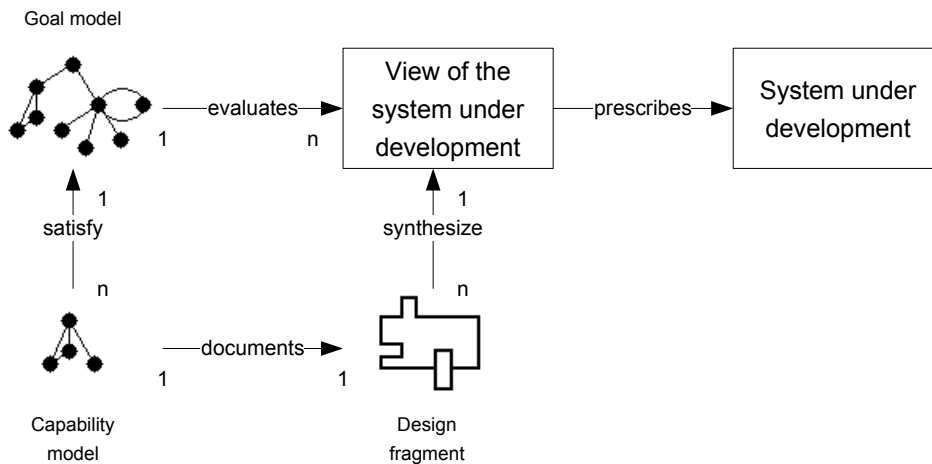


Figure III-4. Structure of a perspective

The requirements for the desired system are captured in a **goal model** (top-left in Figure III-4.) The purpose of the goal model is twofold. First, it is used to **evaluate** all designs generated within a perspective and to determine the best one. Therefore, it is related with one-to-many relationships with the **view of the system under development** (top-middle in Figure III-4.) Second, the goal model is used to select patterns for the design of a particular system under development. The relevance of patterns is determined based on their capability models (bottom-left in Figure III-4): i.e., if a capability model **satisfies** part of the goal model then the pattern of the capability models is selected. The goal model and a capability model are related with one-to-many relationship.

The use of the goal model to (1) select relevant design fragments and (2) to evaluate the final design constitutes the goal-orientation of the framework.

A **capability model** (bottom-left in Figure III-4) **documents** what a **design fragment** (bottom-middle in Figure III-4) can achieve. Although the patterns and the library are not explicitly shown in Figure III-4, the capability model and design fragment stand together for a design pattern.

Design fragments (bottom-middle in Figure III-4) are used as building blocks for the system specification. All selected patterns for a particular view of the system are **synthesized** to build one coherent model. This is represented in Figure III-4 as a many-to-one relationship from the **design fragment** to the **view of the system under development**.

The detailed description about capability models is given in Chapter IV Goal-modelling. The relation between the goal model and the capability models is explained in Chapter V Selection Procedure. The same chapter provides details about the evaluation of potential solutions with the goal model. The synthesis relation between the design

fragments and the view of the system under development is discussed in Chapter VI Synthesis of Value and Process Patterns.

6.4 Design Method within a Single Perspective

This section presents the design method used to generate a model of the system within one perspective.

Figure III-5 gives an overview of the design knowledge transformations within one perspective. The rectangles represent artefacts at various stages of the design; whereas, the arrows show which artefact is derived from which other artefact. The circles with numbers and the text next to them denote the sequence of steps to develop a view of the system.

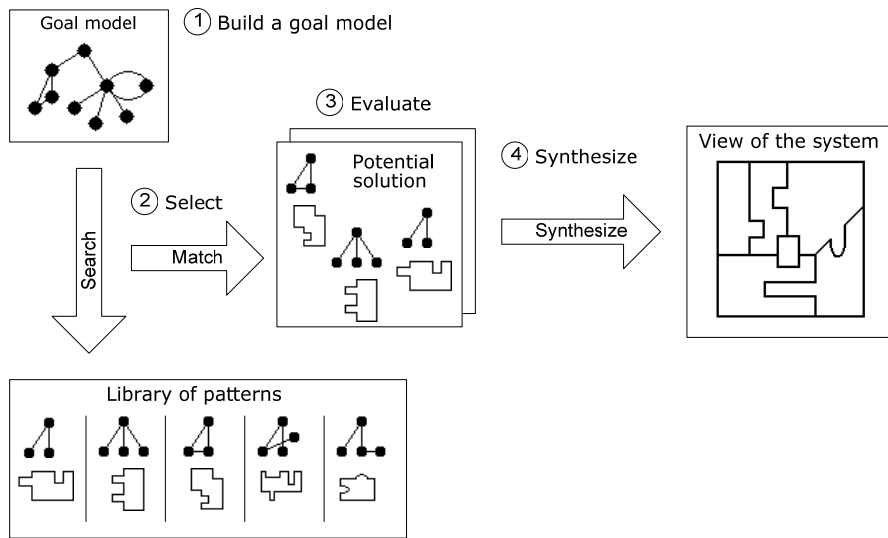


Figure III-5. Design method within a single perspective

The first step of the method is **Build a goal model** (see top-left in Figure III-5.) The requirements for the desired system are elicited and captured in a goal model. This model is further used to select relevant design fragments and evaluate potential solutions (the explanation follows.)

Selecting design fragments is the second step (see middle-left in Figure III-5.) From the available library of patterns, relevant design fragments are chosen such that their capability models match with the goal model.

The library contains more than one pattern that achieves the same goals in the goal model. This requires considering all combination of patterns with duplicating capabilities. Therefore, the result of the matching step is a number of sets of relevant patterns. Each set forms a potential solution (shown in the middle of Figure III-5.); a set does not contain patterns with overlapping capabilities. The potential solutions require the next step **Evaluate**.

The evaluation step ranks the potential solutions with the help of the goal model. The capability models of patterns contain information about the degrees of satisfaction of particular goals. This information is transferred into the goal model for every potential solution. Then the solutions are compared and one is selected as favourable design.

The best scoring set of design patterns is taken in the next **Synthesize** step. From the design fragments, a view of the system is developed, shown to the right in Figure III-5.

Step 2 and 3 are explained in detail in Chapter V Selection Procedure.

6.5 Design Method of the Framework

This section presents the design method used in the framework, among perspectives. It refers to Figure III-1 to illustrate the concepts of the framework.

We assume that the two libraries of value and process patterns are available to us to use. Therefore, the first step is representing the requirements of the system under development as a goal model. This step overlaps with the first step in a single perspective. The importance of mentioning here is that there is only one goal model in the framework for all perspectives.

The next step is developing a perspective. There is no order for developing the perspectives. The designers are free to choose perspective to begin with or develop both in parallel.

The next step to perform, after the two models are generated, is to check the consistency of the specification. Regardless the result of the consistency check, this is the last step prescribed in our framework.

There are several paths to continue. In case the specifications are consistent, one can check, e.g., for correctness or continue adding more information into the specifications. In case the specifications are not consistent, one can follow a path of managing inconsistencies and continue with development. Another option would be to change the specifications until they are consistent and yet another way would be to change the goal model and undergo a second iteration though the framework.

We describe in detail the consistency between models in Chapter VII Consistency between Value and Process Models.

7 Related work

Design theory [56] aims at prescribing computable models that instruct machines how to design new systems. It is usually applied to mechanical construction problems, wherein a new system is specified by evolutionary increments of the design. The latest specification is derived by adding new information to the model. The addition has to be consistent with the already existing specification. The consistency in mechanical systems is achieved through space coordination and known laws of physics, while the same consistency is achieved between, in particular, value and process models (see Chapter VII Consistency between Value and Process Models.)

From a Design theory point of view, our approach is a prescriptive model that is not computable because it includes a number of manual steps: the manual steps use

knowledge that is not part of the prescriptive design model. Comparing with an evolutionary design process model, our framework and method are a design process with one evolutionary step. This is adding the process model to the value model or the other way around, and, consequently, checking the specification for consistency.

8 Summary

This chapter proposed a design framework and method as an answer to our research question **Q2: How to reuse design knowledge in the development process of an e-business specification?** Moreover, it answered the research question **Q2.1: What makes an e-business specification?**

This chapter outlined the overall structure of our framework. Figure III-6 below illustrates the main concepts of the framework and the associated design method. It also points out the chapters of this thesis that provide the details.

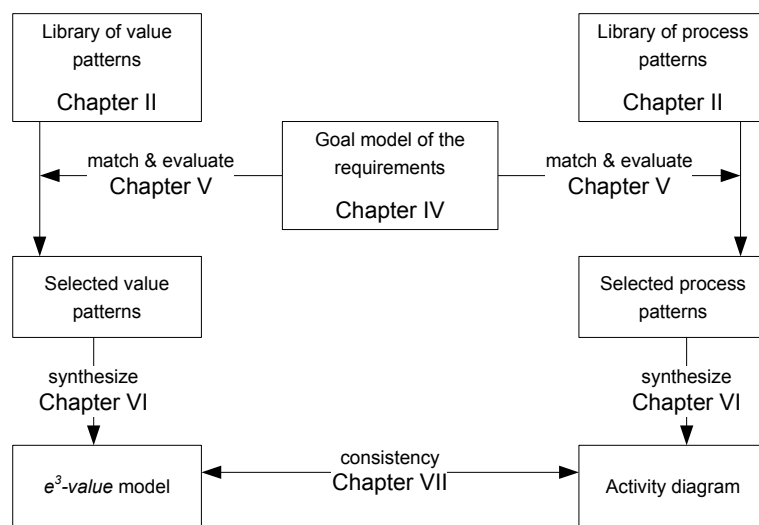


Figure III-6. Overview of the framework with references to the chapters of the thesis

Chapter II Design Knowledge in Existing e-Business Models provides details about the structure of the library of patterns and the documenting structure for patterns.

Chapter IV Goal-modelling defines the way to model requirements and capabilities.

Chapter V Selection Procedure details the procedure for selecting patterns and identifying the best matching combination to be further synthesized.

Chapter VI Synthesis of Value and Process Patterns provides details about the synthesis procedures for e^3 -value models and Activity diagrams.

Chapter VII Consistency between Value and Process Models defines consistency between e^3 -value models and Activity diagrams, and specifies a procedure to check consistency.

Chapter IV

Goal-modelling

Representing Business Requirements

This chapter partially answers our research question **Q2.2.1**. It presents our goal-modelling technique for representation of business requirements. It provides a definition of goal and consequently describes a representation that allows to measure goal satisfaction. Further, three types of relationships among goals are defined with the respective motivation and use of each one of them. Based on the permitted relationships among goals, the chapter gives the rules for constructing goal models. At the end, a mechanism for satisfaction calculation throughout the model is provided. This allows determining the satisfaction of a particular goal based on the values of related goals. The propagation of values is used to select relevant patterns, which is discussed in the next chapter.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling Chapter V Selection Procedure
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

1 Introduction

According to Merriam-Webster Thesaurus [41] a goal is “something that one hopes or intends to accomplish.” This definition states that goals belong to an individual or an organization but remain vague in specifying what has to be accomplished. The definition of WordNet [69] is: “the state of affairs that a plan is intended to achieve and that (when achieved) terminates behavior intended to achieve it.” This definition, though a bit more specific, fails to specify what ‘state of affairs’ is. We interpret ‘state of affairs’ as a phenomenon in the real world. Thus, our definition of

- goal is a desired phenomenon by a certain organization or individual, where a phenomenon is a state of reality, an activity, a fact or an event.

Depending on the nature of goals, some are difficult to operationalize and, therefore, difficult to measure and reason about. An example is *To build a maintainable system*: it is difficult to quantify maintainability. We operationalize the notion of a goal in Section 2, wherein we put some restrictions on the formulation of goals in the goal model. Goals are represented by a variable and an evaluation function. The value of the variable measures the satisfaction of the goal and the evaluation function determined the value that satisfies the goal.

Goals vary in their granularity, subject domains, importance, scope, etc; goals may subsume each other or partially overlap. Examples are *To be the largest multi-media content provider* and *To buy a weather forecast company*. These two goals are from different decision-making levels: the first one is a strategic goal, whereas the second is probably a tactical one. Additionally, the second goal is part of the plan to achieve the first one. The complexity of relations among goals is elaborated in Section 3; where, we define the allowed relationships between goals and constrains on these.

Goals and relationships form goal models. Section 4 discusses the structure of goal models. A goal model can be used to prescribe what is desired to happen. By specifying an evaluation function for each goal in the model, we model what is required; thus, the goal model represents requirements. We refer to such a goal model with stakeholders’ goals model, requirements goal model, or only goal model for short. A dual approach is to describe what is already present. By giving values to goal variables, we model the capabilities of existing phenomena. We use this to represent functionality exhibited by patterns. We refer to goal models with values as capability models.

The way goals are operationalized, allows us to measure whether a particular goal is satisfied or not, or how close it is to satisfaction. In a goal model, we usually know the satisfaction of some goals but want to measure the satisfaction of others. Section 5 describes how the satisfaction of a goal is calculated from the known satisfaction values of related goals.

The graphical modelling notation used to represent goal models is given in Appendix C Goal Modelling Notation.

2 Goal Representation

Goals may be formulated in many different ways: some may be very formal, others may be vague. In any form, they may be useful for different purposes, ranging from better understanding the issue at hand, through qualitative analysis, to quantitative evaluation of options.

For the purpose of our analysis, we require that goal descriptions have clear semantics and their satisfaction can be objectively measured. The goal *To be the largest multi-media content provider* is in general a valid goal but it does not explicitly point to an objectively measurable variable to assess the satisfaction of the goal. Furthermore, the context in which the goal must hold is not made explicit. One possible reformulation of the goal is: *To take 51% of the global multi-media market of content provision in 2 years time*. The measurable variable is market share and the objective is to increase the value to 51%. Alternatively, the goal could be formulated as *To serve more clients than competing multi-media content providers in 2 year time*. In this case, the measurable variable is the number of clients and the objective is to maximize it.

We model a goal as a tuple consisting of: (1) a single variable of particular domain and (2) an evaluation function over the variable domain. The former represents a measurement of something we want to achieve; the latter interprets the achieved value in terms of goal satisfaction.

2.1 Variable

Operationalization of a concept is making it measurable [34: pp. 39—43, 14: pp. 39—40, 383]. This is done by introducing indicators, where an indicator is a measurable variable that has a systematic relationship to the operationalized concept. A systematic relationship is a mapping that connects the observations with the variable values.

To operationalize a goal, we need a number of indicators. These have to be measurable and have to have a systematic relationship with the goal. For example, the goal *To take 51% of the global multi-media market of content provision in 2 years time* is operationalized with the indicators (variables) *Market share* and *Time*. Both can be measured and have clear correspondence with volume of sells and time interval.

To achieve a goal means to do something such that the chosen indicators have the values corresponding to the desired phenomenon. For the example from the previous paragraph, to satisfy the goal means to take some actions to increase the sells such that in two years time the market share measures 51%.

In our approach, a goal is operationalized by a single variable (indicator). This is a simplifying assumption which does not limit the expressiveness of the approach. In case a goal is too complex to be operationalized by a single variable, the goal is refined to a number of single-variable goals. This is described in Section 3. The opposite is also possible: a superfluous goal is introduced to combine several single-variable goals. The need of such a goal and the detail of composing it are discussed in Chapter V Selection Procedure, Section 5 Matching Goal and Capability Models, on page 102.

Variables should be meaningful in the universe of discourse of the goal. For example, the goal *To paint the car red* can be represented by a variable *car colour*. It can be also

represented by *Light wavelength*, but that choice is less meaningful to a car buyer or producer.

2.2 Variable Domain

Variable domain is the set of all valid values that a variable can take and existing relations between the values. We use as variable domains the scales for measurement as defined in behavioural sciences [34: pp. 189—198, 14: pp. 160-165]. Below, we summarize the different types of domain. The numbering has the meaning of specialization; i.e., a higher number domain has all the properties of a lower number domain.

1. Nominal – the values in this domain are not related among each other. An example of such domain is the classification schema for computer games, where the type of game may be one of Fighting, Racing, Role-playing, Simulators, or Strategy.
2. Partially ordered – a transitive comparison operation is defined for some of the values. An example of such domain is the classification schema for movie with the following categories: Action, Thriller, Horror, Comedy, Science Fiction and Fantasy. This represents a partially ordered domain if certain audience (1) has a preference for Comedy, Science Fiction and Fantasy over Action, Thriller, Horror and (2) is indifferent with respect to Action, Thriller, Horror.
3. Ordinal – a transitive comparison operation is defined for all values; i.e. the set of values has a total order. The usual example is the hardness of minerals and the capability of producing a scratch.
4. Ordered Metric – distances are defined between some or all of the values. A transitive comparison operation is defined between the distances. The ordered metric domain is a totally ordered set with partially ordered distances between the elements of the set. An example of such a domain is the FIA regulations for awarding points to the pilots in Formula 1. The first eight pilots get points but the difference in number of points between the first two is greater than between the last two.
5. Interval – distance is defined between all values and the distance between two neighbouring values is equal. The domain has an arbitrary origin; therefore, arithmetic operations can be applied on the distances but not on the values. Examples of such domains are the calendar, the Centigrade, and the Fahrenheit temperature scales.
6. Ratio – origin is defined in the domain, which allows for arithmetic operations between the values and conversion to other ration domains by multiplication of the values. An example is the domain kilograms for measuring weight. It can be easily transformed into the domain of grams by multiplying by 1000.

The classification above is based on the type of relationships defined between the values of the variable. Additionally, domains can be classified according to the number of elements (finite vs. infinite) or the countability of elements (discrete vs. continuous). Examples are:

- Finite discrete domain – *car colour*, with values red, green, blue and pink;

- Infinite discrete domain – *number of produced cars*, with values the natural numbers;
- Continuous domain – *response time*, with values the positive real numbers.

We do not restrict the types of domains. For simplicity only, we refer to domains with the 6 scales for measurement listed above.

Variables domains should be meaningful in the universe of discourse of the goal. For example, the goal *To paint the car red* can be represented by a variable car colour with a domain that has values red, green, blue, and pink. Another possible choice is a domain from the universe of discourse of colours of computer screen pixels: e.g. RGB scale with hexadecimal numbers. The latter, though, would be less meaningful to a car buyer who is unaware of such a standard.

2.3 Evaluation Function

The satisfaction of a goal is usually the most interesting property one wants to make statements about. We introduce an evaluation function which interprets the values of the variables in terms of goal satisfaction. Straightforwardly, a goal is achieved or not; e.g., the goal *To get certificate ISO 9000.1* is satisfied or not. Nevertheless, there are cases in which it is meaningful to say that the achieved state of the world (the result of the effort to satisfy the goal) satisfies the desired state (the goal) to a certain extent. For example, the goal *To offer customer support to 93% of my clients* is not satisfied in case that only 89% of the clients are served; nevertheless 89% is a much better result than, e.g., 47%. We introduce the term *closeness to a goal* to handle terminologically goals that are not fully fulfilled.

Closeness to a goal has different interpretations depending on the domain of the variable representing the goal. In the simplest case, the domain is a set of values without any relationships between them: e.g., the domain of car colour is a set consisting of elements red, green, and pink. In this case, the result of the evaluation function is a Boolean value which is interpreted as the goal is satisfied or the goal is not satisfied. The term closeness is not informative in such a domain. In a more elaborated case, a preference is defined among the values of the variable. In such case, the evaluation function can determine which value is closer to the desired one. For example, if the goal is to have a red car and green colour is preferred over pink then the evaluation function gives the following more informative answers for red, green, and pink colours, respectively: satisfied; not satisfied but better than pink; and not satisfied and worse than green. In even more advanced cases, the distance between values is measurable and comparable. In such domains, the evaluation function can determine how much closer one value is compared to another.

Below, we summarize the capabilities of the evaluation functions in different types of domain. Similarly to the domain numbering, the numbering of evaluation functions represents specialization. The higher the number, the more informative the evaluation result.

1. Nominal domain – the evaluation function can only assess if a goal is satisfied or not.

2. Partially ordered domain – in addition to the assessment of satisfaction or non-satisfaction, the evaluation function can produce qualitative statements of the kind closer, closest, further, and furthest for some of the values.
3. Ordinal – in addition to the assessment of satisfaction or non-satisfaction, the evaluation function can produce qualitative statements of the kind closer, closest, further, and furthest for all values.
4. Ordered Metric – in addition to the assessment of satisfaction or non-satisfaction of a goal and the qualitative analysis of a value, the evaluation function can produce qualitative statements about the distances between some values.
5. Interval – in addition to the assessment of goal satisfaction, qualitative statements about values and qualitative statements about distances, the evaluation function can produce quantitative statements about distances between all values. (For example, the distance between two particular values is this much.)
6. Ratio – in addition to the assessment of goal satisfaction, qualitative statements about values, and quantitative statements about distances, the evaluation function can produce quantitative statements about all values. (For example, value one is this much times bigger than value two.)

The two classifications, finite vs. infinite and discrete vs. continuous, are orthogonal to the six types of domains listed above and do not affect the evaluation function.

2.4 Examples

1. The goal is *To paint the car red*. The variable is *Car colour*. The domain is nominal and contains the following values: *pink*, *green*, *blue*, and *red*. The evaluation function is defined by following statements about the goal satisfaction:
 - if the colour is *red* the goal is satisfied;
 - if the colour is *pink* the goal is not satisfied;
 - if the colour is *green* the goal is not satisfied; and
 - if the colour is *blue* the goal is not satisfied.
2. The goal is *To paint the car red*. The variable is *Car colour*. The domain is partially ordered and contains the following values in the following order: *pink* < *green*, *blue* < *red* (< represents preference.) The evaluation function is defined by the following statements about the goal satisfaction:
 - if the colour is *red* the goal is satisfied;
 - if the colour is *pink* the goal is not satisfied and the desired value is the furthest;
 - if the colour is *green* the goal is not satisfied, it is closer to the desired value compare to *pink*, it is closest to the desired value and its closeness is incomparable with that of *blue*; and
 - if the colour is *blue* the goal is not satisfied, it is closer to the desired value compare to *pink*, it is closest to the desired value and its closeness is incomparable with that of *green*.

3. The goal is *To paint the car red*. The variable is *Car colour*. The domain is ordinal and contains the following values in the following order: *pink* < *green* < *blue* < *red* (< represents preference.) The evaluation function is defined by the following statements about the goal satisfaction:
 - if the colour is *red* the goal is satisfied;
 - if the colour is *pink* the goal is not satisfied and the desired value is the furthest;
 - if the colour is *green* the goal is not satisfied, it is closer to the desired value compare to *pink*, but it is further compare to *blue*; and
 - if the colour is *blue* the goal is not satisfied, it is closer to the desired value compare to *pink* and *green*, and it is the closest to the desired value.

4. The goal is *To paint the car red*. The variable is *Car colour*. The domain is an ordered metric defined with one-to-one mapping with the Natural numbers: *pink* maps to 5, *green* maps to 45, *blue* maps to 96 and *red* maps to 110. The evaluation function is defined by the following statements about the goal satisfaction:
 - if the colour is *red* the goal is satisfied;
 - if the colour is *pink* the goal is not satisfied, the desired value is the furthest and the distance is 105;
 - if the colour is *green* the goal is not satisfied, only *pink* is further from the desired value, it is closer to *pink* than to the desired value, the distance to the desired value is 65; and
 - if the colour is *blue* the goal is not satisfied, it is closer to the desired value compare to *pink* and *green* with 91 and 51, respectively, and it is closest to the desired value.

5. The goal is *To take 51% of the global multi-media market of content provision in 2 years time*. The variable is *market share*. The domain is a ratio with real numbers from 0 to 100. The evaluation function is defined by the following statements about the goal satisfaction:
 - if the market share is greater or equal to 51% then the goal is achieved; and
 - if the market share is less than 51% then the goal is not achieved.

6. The goal is *To get certificate ISO 9000.1*. The variable is *possession of a certificate*. The domain is nominal. The evaluation function is defined by the following statements about the goal satisfaction:
 - if the value is true then the goal is satisfied; and
 - if the value is false then the goal is not satisfied.

7. The goal is *To offer customer support to 93% of my clients*. The variable is *percent of clients*. The domain is a ratio with the real numbers from 0 to 100. The evaluation function is defined by the following statements about the goal satisfaction:
 - if the value is $93\% \pm 1$ then the goal is satisfied; and

- if the value is greater than 94 or less than 92 the goal is not satisfied and the distance to the goal in number of clients can be calculated with the following formula: $|\text{achieved percentage}-93|*\text{all clients}/100$.

2.5 Implication for Goal Model Analysis

Depending on the domain of the goal variable and the evaluation function, different types of analysis can be performed on the goal model. If the domain of the goal variable cannot express the stakeholders' preference order over the variable values then the analysis is limited to yes/no answers to the goal satisfaction question. In case the domain has a partial order, a more informative analysis can be performed, namely a qualitative analysis. The evaluation function can compare some of the value of the variable and tell which one is closer to the desired value. Finally, a quantitative analysis can be performed in a domain with metric. In such cases, the evaluation function can tell how far off target one is with respect to the goal. The examples 1, 2, 3 and 4, related to the same goal but with different domains and evaluation results, show how the details produced by the evaluation function increase with the choice of a variable domain.

3 Relations between Goals

In the previous section, we discuss the structure of a single goal. With the proposed operationalization, one can make a goal explicit to a certain extent. However, this may not be sufficient in case the goal is too general or too abstract, or too complex. Such a goal is usually represented by a variable and an evaluation function that are too complex to be useful. A common-sense strategy to solve this problem is the divide and conquer approach. Under the assumption that concrete goals are easier to reason about, a complex goal is divided into simpler sub-goals. In such case, the evaluation of the complex goal is performed based on the sub-goals and their relationships with the complex goal.

The relationships between a complex goal and the division of sub-goals are not the only type of relationship between goals. In general, relationships explicate dependencies between goals. Thus, they also model overlap, conflict, or reinforcement between goals.

In this section, we describe the types of relationships we use. As goals are represented by variables, the goal relationships are defined as relationships between variables. That is, a dependency relationship between goals is defined as a causal relationship between the variables representing the goals.

3.1 Causal Relationships

Two goals are related if there is a causal relationship between the variables representing them. Two variables are causally related if a change in the value of one of them results in a change of the value of the other one. Causal relationships are asymmetric. The variable that causes the effect is called an independent variable (IV); respectively, the variable that is affected is called a dependent variable (DV).

A causal relationship between two variables is difficult to establish [14: pp. 142—151]. To the best of our knowledge, we assume causal relationships coming from laws of nature (we believe that something happens according to the norm until we find a counter example) or empirically validated hypothesis (we observe that a relationship holds in sufficiently many real world examples.)

3.2 Types of Causal Relationships

Figure IV-1 shows an example of two goals, **G1** and **G0**, and two variables, IV1 and DV. The independent variable IV1 represents the goal **G1** and the dependent variable DV represents the goal **G0**. IV1 and DV are related with an asymmetric causal relationship which can have several interpretations as a goal relationship (the dashed line in Figure IV-1.) The goal relationship has a direction which repeats the direction of the causal relationship. For ease of reference, we call the goal, whose variable causes the effect, a tail goal and we call the goal, whose variable is affected, a head goal (see Figure IV-1.)

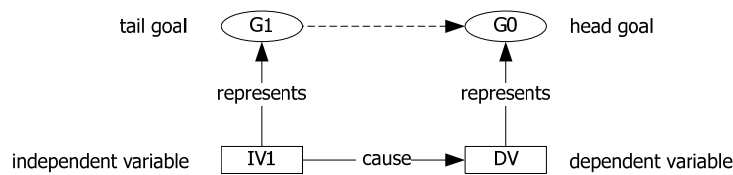


Figure IV-1. Causal relations between two variables

Asymmetric causal relationships manifest themselves in several different types in the universe of discourse of goal satisfaction. Based on the interpretation of the variable values in terms of goal satisfaction, we distinguish two types of asymmetric causal relationships.

- *Determinative*. This is a causal relationship from IV1 to DV which has the following property: a value of the independent variable IV1 (see Figure IV-1) that satisfies goal **G1** causes a value for the dependent variable DV which satisfies goal **G0**. (There exists a value that causes satisfaction.) The interpretation as a goal relationship is: the satisfaction of goal **G1** results in the satisfaction of goal **G0**.
- *Influential*. This is also a causal relationship from IV1 to DV but it has the following property: none of the values of the independent variable IV1 causes a value for the dependent variable DV which satisfies goal **G0**. (There is not a value that causes satisfaction.) The interpretation as a goal relationship is: the satisfaction of goal **G1** does not result in the satisfaction of goal **G0**. There is a causal relationship across the variables, but goal satisfaction does not propagate along this relationship.

Figure IV-1 contains only two goals and, respectively, shows one independent and one dependent variable. Often, a goal depends on more than one goal. In such a case, more than one independent variable affects a single dependent variable. Figure IV-2 illustrates this. The goal **G0** (the head) depends on goals **G1** and **G2** (the tails). The head is represented by a dependent variable DV; the tails are representing by independent

variables IV1 and IV2, respectively for **G1** and **G2**. It is important to note that goals **G1** and **G2** are not related. Correspondingly, variables IV1 and IV2 are not related.

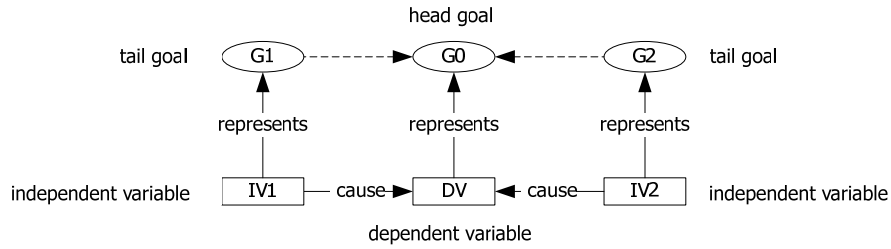


Figure IV-2. Causal relations among three variables

Causal relationships manifest themselves differently in the universe of discourse of goal satisfaction if they affect a single dependent variable. Based on the interpretation of variable values, we classify a group of causal relationships as *individual* or *collective*.

- *Individual*. A group of causal relationships are classified as individual if none of the independent variable can cause a better value in terms of goal satisfaction if acting in combination with another independent variable.
- *Collective*. A group of causal relationships are classified as collective if none of the independent variable can cause in isolation as good value in terms of goal satisfaction as if acting together with the other independent variable.

The two classifications are orthogonal. A causal relationship can be determinative or influential and, in addition to that, individual or collective. (The second classification makes sense only as a part of a group.) The combined classifications have the following meaning:

- *Individually determinative* – each relationship from the group independently of the other relationships affects the dependent variable as an only determinative relationship.
- *Individually influential* – each relationship from the group independently of the other relationships affects the dependent variable as an only influential relationship.
- *Collectively determinative* – only all relationships from the group together affect the dependent variable as a single determinative relationship. Independently of the other relationships, each relationship is an only influential relationship. The latter holds with exception of the special case when all other independent variables evaluate to satisfaction. Then the relationship is determinative.
- *Collectively influential* – each relationship, independently or with the others, is only an influential relationship.

3.3 Types of Goal Relationships

The different types of causal relationships have different interpretations as relationships between goals. We distinguish between three types of goal relationships, namely: *decomposition*, *substitution* and *dependency*. Table IV-1 defines the correspondence between causal and goal relationships. The first column shows the number of causal

relationships (assuming that the many causal relationships share one dependent variable.) The second and the third columns contain the types of causal relationships and goal relationships, respectively. Further below, we explain each of the goal relationships.

Table IV-1. Types of causal and goal relationships

Number of causal relationships	Type of causal relationship between variables	Type of goal relationship
1	Determinative	Substitution
	Influential	Dependency
> 1	All are Individually determinative	Substitution
	All are Individually influential	Dependency
	All are Collectively determinative	Decomposition
	All are Collectively influential	Dependency
	Only some are Individually determinative	
	Only some are Collectively determinative	
	Some are Individually influential and some are Collectively influential	Dependency

3.4 Substitution Relationship

A substitution relationship means that the plan of actions to achieve the head goal can be replaced with a plan of actions to achieve a tail goal: i.e., achieving any of the tail goals guarantees the satisfaction of the head goal.

Table IV-1 defines the substitution relationship based on causal relationships. It contains two definitions of substitution. The first one is based on a single isolated determinative causal relationship; where the second is based on a group of individually determinative causal relationships. The latter subsumes the former. The definition requires (1) that every causal relationship has to be determinative and (2) that every causal relationship has to act independently of the others. This guarantees that the satisfaction values of only one of the tail goals will cause a satisfaction value for the head goal.

From a goal satisfaction point of view, a substitution relationship is a relationship between a single goal and an arbitrary number of other goals, the satisfaction of any of which guarantees the satisfaction of the single goal.

Figure IV-3. illustrates through an example the visual notation for representing substitution. The relationship between tail goals and the head goal is expressed with arrows from the tail goals pointing at a circle that denotes the type of relationship with ‘s’ letter in it. The circle is connected with a line to the head goal. From the example goals on the right-hand side of Figure IV-3., one can see that the goal **G0** can be achieved through two alternatives, goals **G1** and **G2**.

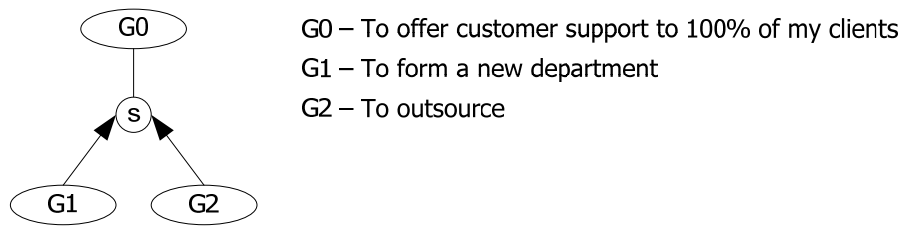


Figure IV-3. Example of substitution relationship

3.5 Decomposition Relationship

A decomposition relationship means that the plan of actions to achieve the head goal can be replaced with plans of actions to achieve all tail goals: i.e., achieving all tail goals guarantees the satisfaction of the head goal.

Table IV-1 defines the decomposition relationship on the basis of causal relationships. It contains one definition which is based on a group of collectively determinative causal relationships. The definition requires that every causal relationship is collectively determinative. This guarantees that the satisfaction values of all tail goals will cause a satisfaction value for the head goal.

The difference between a substitution and decomposition relationships is that the tail goals in a substitution relationship can separately guarantee satisfaction of the head goal, whereas the tail goals in a decomposition relationship can only jointly guarantee the satisfaction of the head goal.

A decomposition relationship is a breakdown of a goal into a number of sub-goals. The newly introduced sub-goals jointly fully replace the original goal.

Figure IV-4. illustrates through an example the visual notation for representing goal decomposition. The relationship between the tail goals and the head goal is expressed with lines from the tail goals to a circle that denotes the type of relationship with 'd' letter in it. The circle is connected with the decomposed goal with an arrow pointing at the goal. From the example goals on the right-hand side of Figure IV-4., one can see that the goal **G0** can be achieved only if both goals **G1** and **G2** are achieved.

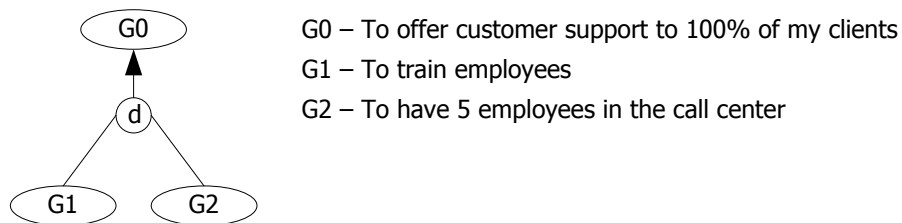


Figure IV-4. Example of decomposition relationship

3.6 Dependency Relationship

A dependency relationship means that the head goal is affected from the plans of actions to achieve the tail goals but never fully satisfied: i.e., achieving the tail goal changes the

head goal but does not result in the satisfaction of the head goal. The effect of a tail goal may even be harmful to the satisfaction of the head goal.

The dependency relationship is meaningful to define only if the head goal is represented by a variable domain different from nominal. This restriction comes from the general requirement for detecting a relationship: there is a relationship if the evaluation function of the head goal can distinguish between the effects caused by the value of the tail goal. In a dependency relationship where the head goal is represented by a nominal domain, the resulting value of a head goal evaluates always to non-satisfaction. Therefore, we assume that the head goal involved in a dependency relationship is not modelled with a nominal domain.

Table IV-1 defines the dependency relationship on the basis of causal relationships. It contains four definitions all of which include only influential causal relationships. The definition requires that no determinative relationships constitute a dependency relationship. This is a premise for the non-satisfaction of the head goal.

As mentioned above, the dependency relationship can model harmful influences. This is an important feature, as the other two relationships cannot model that. Therefore, we distinguish between two types of dependency relationships: a *positive dependency relationship* and a *negative dependency relationship*. A positive dependency relationship means that the closer to satisfaction the goal causing the effect is, the closer to satisfaction the affected goal is. Correspondingly, a negative dependency relationship means that the closer to satisfaction the goal causing the effect is, the further from satisfaction the affected goal is.

A dependency relationship is a weaker relationship in comparison to substitution or decomposition. It cannot guarantee satisfaction of the head goal under any conditions. Still, the relationship is useful to model qualitative relationships or ones without enough knowledge for the actual causal relationship between variables.

Another distinctive property of the dependency relationship is that it can express negative influences¹. Both substitution and decomposition include only tail goals that have positive influence on the head goal.

Figure IV-5 shows the visual notation for representing dependency relationships. The relationship between two goals is expressed with an arrow that originates in the goal that causes the effect and points to the affected goal. There are two types of arrowheads: a solid arrowhead that denotes a relationship with positive effect and a hollow arrowhead that denotes a relationship with negative effect. From the example goals on the right-hand side of Figure IV-5, one can see that the goal **G0** is affected positively by goals **G1** and **G2**, and negatively by goal **G3**.

¹ A relationship with negative influence is conceptually different from a negated goal. Every goal can be negated by reformulating it in such a way that the evaluation function is satisfied by values previously evaluating to non-satisfaction. A negative dependency relationship will not turn into positive if the tail goal is negated. The rejection of the tail goal will not satisfy or positively contribute to the satisfaction of the head goal. Any value of the tail goal will always, strongly or weakly, negatively influence the head goal.

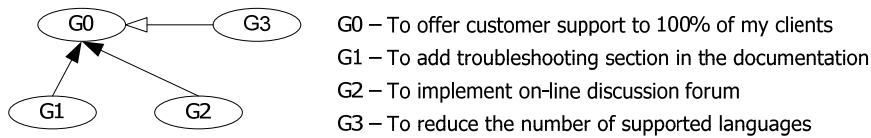


Figure IV-5. Example of dependency relationship

3.7 Restrictions on Goal Relations

For the purpose of our modelling needs, we find the substitution, decomposition, and dependency types of relationships sufficient. Below, we survey the combinations of goal relationships that we do not allow. We argue the introduced constrains with the interpretation of causal relationships between variables.

A goal may be a tail goal of any type of relationship and, in the general case, a single goal may be a head of more than one type of goal relationship. For the purpose of conceptual clarity, we restrict a goal to be a head of only one type of relationship: substitution, decomposition, or dependency. Table IV-1 reflects this restriction at causal relationships level. It does not define goal relationships for the three cases of mixed causal relationships.

1. A mixture of an individually determinative causal relationship and any influential causal relationship would translate to substitution and dependency relationships with a single head goal. Figure IV-6. (a) shows an example of such a case. A goal **G0** is dependent on a goal **G3** and at the same time substituted by goals **G1** or **G2**. The work-around solution to dependency and substitution relationships sharing a single head goal is to assume that the tail goal of the dependency relationship will also affect the substituting goals. The assumption allows us to transfer the dependency relationship to the substituting goals. Figure IV-6. (b) illustrates the outcome, where goal **G3** affects goals **G1** and **G2** instead of **G0**.

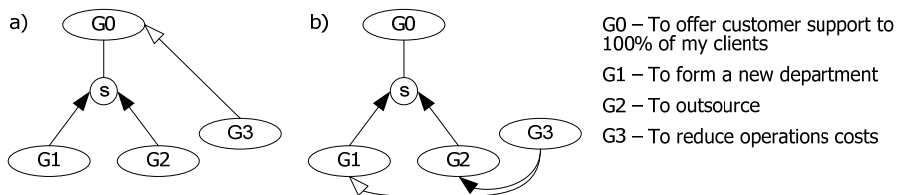


Figure IV-6. Example of mixed individually determinative and influential causal relationships: (a) – an invalid model with mixed relationships; and (b) – a valid model, result of a work-around

2. A mixture of individually determinative and collectively determinative causal relationships would translate to substitution and decomposition relationships with a single head goal. Figure IV-7 (a) shows an example of such a case. A goal **G0** can be substituted by goals **G1** and **G2** or by goals **G3** and **G4**. The work-around solution is to introduce superficial intermediary goals, see Figure IV-7

(b). Goals **G5** and **G6** represent two alternatives for achieving goal **G0** and they are composed of the previously sub-goals of **G0**.

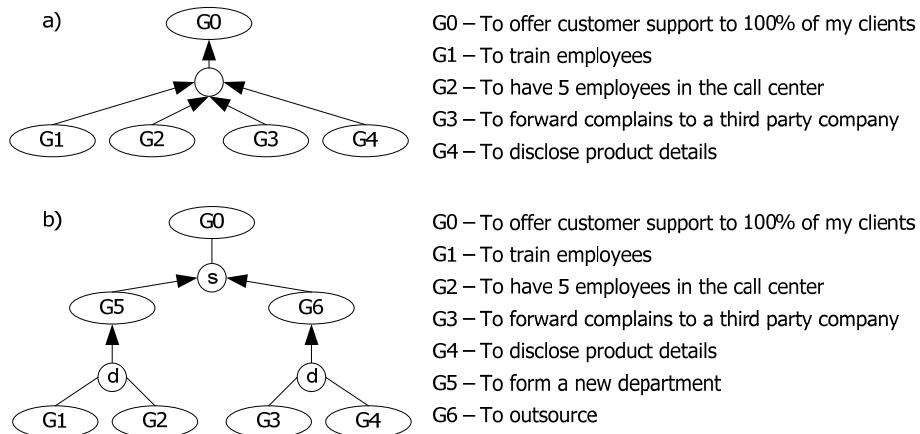


Figure IV-7. Example of mixed individually determinative and collectively determinative causal relationships: (a) – an invalid model with mixed relationships; and (b) – a valid model, result of a work-around

3. A mixture of a collectively determinative causal relationship and any influential causal relationship would translate to decomposition and dependency relationships with a single head goal. Figure IV-8 (a) shows an example of such a case. A goal **G0** is dependent on a goal **G3** and at the same time decomposed to goals **G1** or **G2**. The work-around solution to dependency and decomposition relationships sharing a single head goal is to assume that the tail goal of the dependency relationship will also affect the sub-goals. The assumption allows us to transfer the dependency relationship to the sub-goals. Figure IV-8 (b) illustrates the outcome, where goal **G3** affects goals **G1** and **G2** instead of **G0**.

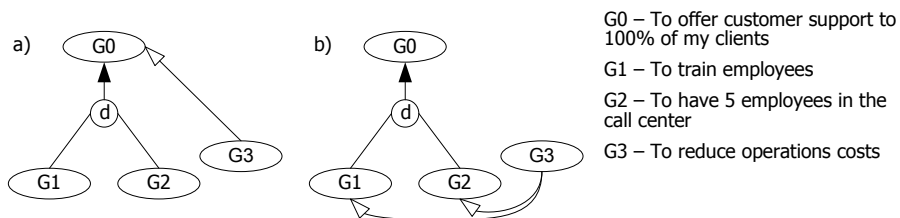


Figure IV-8. Example of mixed collectively determinative and influential causal relationships: (a) – an invalid model with mixed relationships; and (b) – a valid model, result of a work-around

To this end, we restricted the type of relationships in which a single goal is a head. With the same purpose of conceptual clarity in mind, we restrict the cardinality of relationships. This is as following: a goal is a head of (1) only one substitution

relationship or (2) only one decomposition relationship. The dependency relationships are not restricted.

The first restriction, on the cardinality of the substitution relationship, is only notational. Two separately modelled substitution relationships have the same meaning as one relationship with a union of the alternatives. The cardinality of decomposition relationships can be avoided introducing superficial intermediary goals (the same approach as described in Figure IV-7), even if a sub-goal takes part in several decompositions.

These restrictions do not limit the expressiveness of the modelling approach.

4 Goal Model

A requirements goal model represents a state of the world that a person or organization wants to achieve. This is captured in one or several goals which we call top-level goals¹. Top-level goals are further clarified by either substituting them with goals that are more comprehensible or by decomposing them in measurable sub-goals. Each top-level goal is potentially a root of a substitution or decomposition tree; thus, a goal model is, in the general case, a **forest**.

In contrast to the regular structure resulting from the substitution and the decomposition, the dependency relationship does not follow a hierarchical structure. The dependency relationship is binary, can originate from every goal and can (complying with the restrictions) link to an arbitrary goal in the model. The dependency relationship appears between branches in one tree and between goals from different trees in the forest. Thus, a goal model is a **graph** structure consisting of: at least one top-level goal, substitution and decomposition relationships forming trees, and dependency relationships linking branches and trees.

The goal relationships are asymmetric, which turns the goal model into a **directed graph**. As there are no constraints on the chain of relationships, there is a possibility of **cycles** occurring in the graph structure. The existence of cycles is apparent from the example in Figure IV-9 which we discuss next. In summary, a goal model is a directed graph that may contain cycles.

¹ The top-level goals are determined by the stakeholders of the system under development. They represent in a concise form what the system has to achieve. They are the starting point for the development of the goal model. Top-level goals have no implication on the structure or semantics of the goal model. We use them for easier navigation and explanation of the goal model.

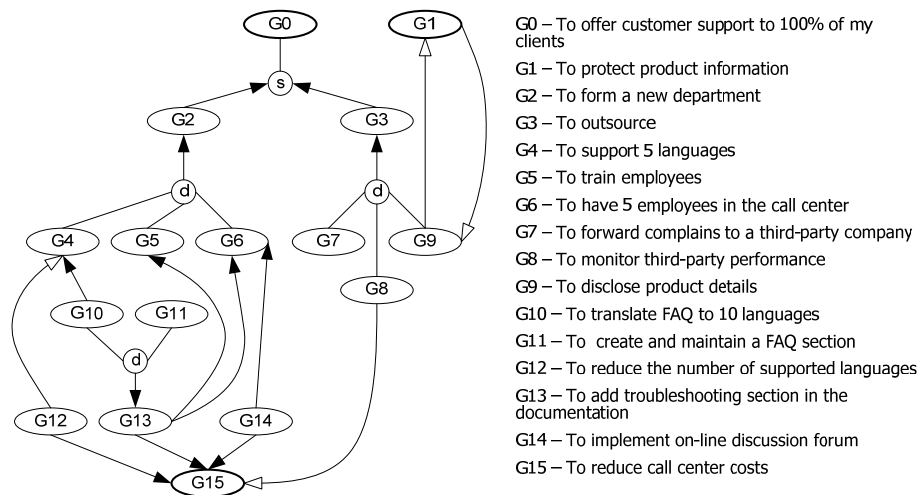


Figure IV-9. Example of a goal model

Figure IV-9 presents an example goal model, wherein the various possible relationships between goals are shown. (The graphical modelling notation is given in Appendix C Goal Modelling Notation.) The model has three top-level goals: **G0**, **G1** and **G15**. Following the textual description to the right in the figure, one can see that goal **G0** can be achieved by two alternative paths: either achieving **G2** or **G3**. Both **G2** and **G3** are too complex to be assessed directly; therefore, they need to be decomposed into the sub-goals from **G4** to **G9**. Goal **G15** is different because its fulfilment cannot be guaranteed. This is captured by dependency relationships with the rest of the model. Goal **G1** is interesting with the cyclic dependency relationship with goal **G9**. The two goals negatively influence each other and need a trade-off to determine an optimal solution. The remaining dependency relationships show the synergy (positive effect) and contradictions (negative effect) between goals. Goals **G13**, **G10** and **G11** show that a root of decomposition tree can be any goal, not only a top-level one.

Labelling some of the goals as top-level goals leads to the introduction of levels of abstraction in a goal model. For easy navigation through the model, we assume that the top-level goals are the most abstract ones. Clarifying the top-level goals with related goals refines the top-level goals to less abstract goals. A leaf-goal¹ is a goal that is not a head in any relationship. This means that the leaf-goals are the finest refinement of the top-level goals.

Capability models use also the notion of top-level capability, abstraction levels, and leaf-capabilities. Correspondingly to the requirements goal model, a top-level capability is a capability chosen by a person to represent the most important capability of an exiting phenomenon. A leaf-capability is a capability that is only tail in its relationships with other capabilities.

¹ A top-level goal and a leaf-goal are not opposites. A business analyst chooses a goal to be a top-level. A leaf-goal can be a top-level goal. A leaf-goal is a goal that is not influenced by any other goal because it is only a tail in its relationships

5 Satisfaction Calculation

We use a requirements goal model to evaluate the satisfaction of a specific goal, usually a top-level goal. This is done by applying the evaluation function on the goal variable. The goals of a requirement model do not have assigned values. They receive these from matching capability models of patterns. The problem is that capabilities rarely match top-level goals. At a certain stage, we know the value of some of the goal variables, usually leaf-goals. In order to compute the value of a top-level variable, we have to propagate the known values using the relations among goals. This section describes how this is done.

The variables of some goals receive their values as input data for the goal model. (They cannot change these initial values during the evaluation process.) The rest of the variables get their values as result of calculation from already known values. In a single relationship, the value of the variable representing the head goal is calculated from the values of the variables of the tail goals, assuming the variables of the tail goals are known.

A single relationship includes several alternative ways to calculate the value of the head goal. For example, a substitution relationship includes tail goals, the variables of which individually cause the effects on the variable of the head goal. That is, a single tail goal is sufficient to calculate the value of the head goal. In the particular case of substitution relationship, the number of possible ways to calculate the value of the head goal is equal to the number of tail goals. In general, the possible ways to calculate the value of the head goal is equal to the number of combinations of 1 to n elements, where n is the number of tail goals. We denote the alternative ways for value calculation with *alt*. The maximum number of alternatives is given by the following formula:

$$\sum_{k=1}^n \binom{n}{k} = \sum_{k=1}^n \frac{n!}{k!(n-k)!}$$

Every relationship is associated with a value propagation function *VPF* that calculates the value of the head goal. The *VPFs* are unique for every relationship; nevertheless, they share common properties. Before we discuss these, we introduce the notation we use. Figure IV-10 shows the three types of relationships: (a) substitution, (b) decomposition and (c) dependency. Every goal in the relationships is annotated additionally with a variable that represents the goal. Further, we refer to the goals and variables with the symbols from the figure. For every type of relationship:

- the head goal is denoted with **G0**;
- the dependent variable representing **G0** is denoted with *y*;
- the tail goals are denoted with **G1**, **G2**, ..., **Gn**;
- the independent variables representing **G1**, **G2**, ..., **Gn** are denoted with x_1, x_2, \dots, x_n , respectively;
- n is the number of tail goals in the relationship.

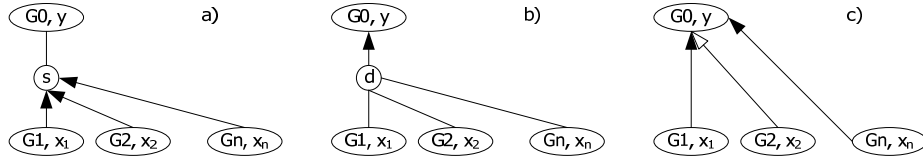


Figure IV-10. Abstract propagation functions: (a) – substitution propagation function; (b) – decomposition propagation function; and (c) – dependency propagation function

The form of the value propagation function VPF for every type of relationship is:

$$y = VPF(x_1, x_2, \dots, x_n, alt).$$

In general, the VPF depends on all independent variables representing tail goals and on the particular combination of tail goal taking part in the calculation. An alternative way to represent the value propagation function is

$$y = \begin{cases} f_1(x_k, \dots, x_l), alt = 1 \\ f_2(x_p, \dots, x_q), alt = 2 \\ \dots \\ f_m(x_s, \dots, x_t), alt = m \end{cases},$$

where:

- f_1, f_2, \dots, f_m are total single-valued multivariate functions representing a law of nature, empirical equation or any other formula that holds in the universe of discourse of the particular modelled system. (To avoid confusion with the value propagation function VPF , these functions are referred to as alternative propagation function APF ;))
- $k, l, p, q, s,$ and t are indexes from 1 to n representing an arbitrary number of function arguments; and
- m is an arbitrary number from 1 to $\sum_{k=1}^n \frac{n!}{k!(n-k)!}$ representing the last alternative for the particular relationship.

The general form of the value propagation function is further specialized for every type of goal relationship.

5.1 Identification of Propagation Functions

The propagation functions are causal laws in a particular domain. They encode domain and business knowledge. A domain expert, supported by a knowledge engineer and a business analyst, should formulate the propagation functions.

The identification of the propagation functions is not an easy matter because this is done before a business is started. For some of the functions the existing domain and business knowledge may not be enough. Additional research or forecasting may be required.

5.2 Substitution Propagation Function

A substitution relationship offers one or more alternatives to achieve a goal. Each alternative is a single tail goal and it is analyzed separately. The implication for the value propagation is that only one of the tail goals determines the value of the substituted goal. Therefore, the value propagation function *VPF* consists of as many alternative propagation functions *APFs* as the number of tail goals.

The *VPF* for a substitution relationship is defined with the following formula

$$y = \begin{cases} fsub_1(x_1), alt = 1 \\ fsub_2(x_2), alt = 2 \\ \dots \\ fsub_n(x_n), alt = n \end{cases},$$

where:

- *fsub* is an alternative propagation function of a single argument; and
- the number of alternative propagation functions is equal to the number of tail goals *n*.

Example of Substitution Propagation Function. Let us assume that we have a goal **G0** that is related with two other goals **G1** and **G2** with a substitution relationship (see Figure IV-3.). The three goals are operationalized with the following variables and domains. (The evaluation functions are not specified because they are irrelevant for the propagation of values.)

1. Goal G0: To offer customer support to 100% of my clients.
 - goal variable: *y* – number of clients
 - variable domain: natural numbers
2. Goal G1: To form new department
 - goal variable: *x*₁ – number of trained employees
 - variable domain: [0 .. 10]
3. Goal G2: To outsource
 - goal variable: *x*₂ – number of forwarded calls
 - variable domain: natural numbers

As an example of a value propagation function assigned to the substitution relationship we take:

$$y = \begin{cases} (allclients - callclients) + callclients\sqrt{x_1/10}, alt = 1 \\ (allclients - callclients) + x_2, alt = 2 \end{cases},$$

where:

- *allclients* is a constant representing the number of clients. Its value is 1000;
- *callclients* is a constant that represents the number of clients with problems that call for help. Its value is 320;
- $\sqrt{1/10}$ is arbitrary chosen as an example.

Alternative One. In case goal **G1** is selected for analysis, the propagation function for the substitution relationship is

$$y = f_{sub_1}(x_1) = (allclients - callclients) + callclients \sqrt{x_1/10}.$$

Figure IV-11 presents the propagation functions of the first alternative. It shows that all clients will be offered customer support when 10 trained employees take their calls.



Figure IV-11. Example of a substitution propagation function for alternative one

Alternative Two. In case goal **G2** is selected for analysis, the propagation function for the substitution relationship is

$$y = f_{sub_2}(x_2) = (allclients - callclients) + x_2.$$

Figure IV-12 presents the propagation functions of the second alternative. It shows that the insourcing company has unlimited capacity and can serve all forwarded call. For the outsourcing company, the number of helped client is linearly dependent on the number of forwarded call.

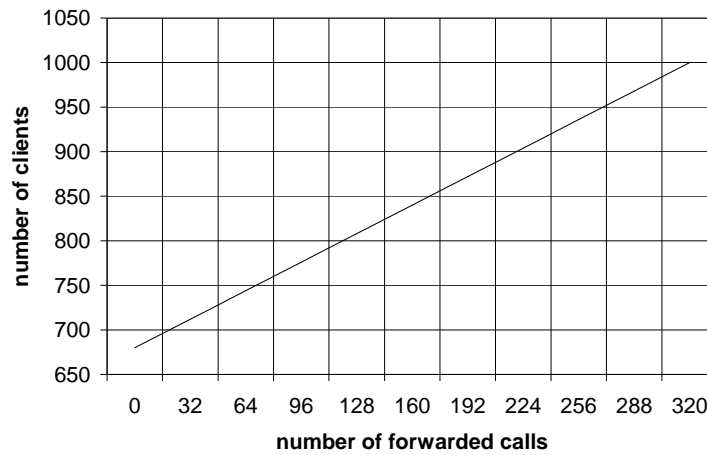


Figure IV-12. Example of a substitution propagation function for alternative two

5.3 Decomposition Propagation Function

A decomposition relationship is a substitution of a single goal by a collection of sub-goals. The sub-goal can achieve the decomposed goal only jointly. The implication for the value propagation function is that it consists of a single alternative propagation function that dependent on all tail goal variables.

The *VPF* for a decomposition relationship is defined with the following formula

$$y = VPF(x_1, x_2, \dots, x_n, 1) = fdec(x_1, x_2, \dots, x_n),$$

where:

- *fdec* is an alternative propagation function of n arguments; and
- the number of alternative propagation functions is 1.

Example of Decomposition Propagation Function. Let us assume that we have a goal **G0** that is related with two other goals **G1** and **G2** with a decomposition relationship (see Figure IV-4.). The three goals are operationalized with the following variables and domains. (The evaluation functions are not specified because they are irrelevant for the propagation of values.)

1. Goal G0: To offer customer support to 100% of my clients
 - goal variable: y – percent of clients
 - variable domain: real number from 0 to 100
2. Goal G1: To train employees
 - goal variable: x_1 – trained
 - variable domain: Boolean
3. Goal G2: To have 5 employees in the call centre
 - goal variable: x_2 – number of employees
 - variable domain: [0 .. 5]

As an example of a value propagation function assigned to the decomposition relationship we take:

$$y = \begin{cases} 100(allclients - callclients + callclients\sqrt{x_2/5})/allclients, & x_1 = true \\ 100(allclients - callclients)/allclients, & x_1 = false \end{cases}$$

where:

- *allclients* is a constant representing the number of clients. Its value is 1000;
- *callclients* is a constant that represents the number of clients with problems that call for help. Its value is 320;
- $\sqrt{1/5}$ is arbitrary chosen as an example.

Figure IV-13 presents the propagation functions. It is a function of two variables presented in a two-dimensional diagram. The diagram contains two lines to represent the dimension of the Boolean variable x_1 . Figure IV-13 shows that without trained employees the percentage of helped clients remains 68%; whereas, if the employees are trained the percentage increases with the number of employees.



Figure IV-13. Example of a decomposition propagation function

5.4 Dependency Propagation Function

A dependency relationship offers one or more alternatives to influence a goal. Each alternative is made of one or more tail goals and it is analyzed separately. The implication for the value propagation is that one or more tail goals determine the value of the substituted goal. Therefore, the value propagation function *VPF* may consist of as many alternative propagation functions *APFs* as the number of combinations of tail goals.

One difference between the dependency relationship and the other two is in the uncertainty of head goal satisfaction: the head goals in substitution and decomposition relationships can be satisfied under certain conditions holding for the tail goals; whereas, the satisfaction of the head goal in a dependency relationship cannot be guaranteed under any conditions on the tail goals.

The dependency relationship stands in-between the decomposition and the substitution relationships. That is, a single tail goal can act as a substitution goal but cannot guarantee the satisfaction of the head goal. Correspondingly, a group of dependency tail goals can jointly affect the head goal such as a decomposition relationship but not be able to cause the head goal satisfaction. Additionally, a single tail goal can be part of several groups of jointly acting dependency relationships. Therefore, the value propagation function *VPF* consists of an arbitrary number of alternative propagation functions *APFs*.

The *VPF* for a dependency relationship has the general form of *VPF*. It is defined with the following formula:

$$y = \begin{cases} fdep_1(x_k, \dots, x_l), alt = 1 \\ fdep_2(x_p, \dots, x_q), alt = 2 \\ \dots \\ fdep_m(x_s, \dots, x_t), alt = m \end{cases},$$

where:

- *fdep* is an alternative propagation function of an arbitrary number of arguments; and
- *k..l* are indexes of the elements of an arbitrary subset of the set of all tail goal variables. *p..q* and *s..t* are the same kind of indexes.
- *m* is an arbitrary number between 1 and the maximum number of alternative ways to propagate value in a relationship (discussed in the beginning of the section).

Despite the complex form of the value propagation function, it is rare to have it in full length. The goal model analyst aims for simplicity; therefore, it is uncommon to overload the dependency relationships in such a way. Most frequently, the dependency relation looks such as a decomposition relation, with one alternative propagation function on all variables

$$y = fdep(x_1, x_2, \dots, x_n).$$

Another common form of the dependency propagation function looks such as a substitution relation, where every tail goal independently of the others influences the head goal

$$y = \begin{cases} fdep_1(x_1), alt = 1 \\ fdep_2(x_2), alt = 2 \\ \dots \\ fdep_n(x_n), alt = n \end{cases}.$$

There is no difference in the form of the last two variations of the dependency propagation function, and the substitution and decomposition propagation functions. The

difference is in the propagated values. The dependency *VPF* does not guarantee that the propagated value will evaluate to satisfaction.

Example of Dependency Propagation Function. Let us assume that we have a goal **G0** that is related with two other goals **G1** and **G2** with a dependency relationship (see Figure IV-5; goal **G3** is not part of the example). The three goals are operationalized with the following variables and domains. (The evaluation functions are not specified because they are irrelevant for the propagation of values.)

1. Goal G0: To offer customer support to 100% of my clients
 - goal variable: y – percent of clients
 - variable domain: real number from 0 to 100
2. Goal G1: To add troubleshooting section in the documentation
 - goal variable: x_1 – number of questions and answers
 - variable domain: Natural numbers
3. Goal G2: To implement an on-line discussion forum
 - goal variable: x_2 – number of topics
 - variable domain: Natural numbers

As an example of a value propagation function assigned to the dependency relationship we take:

$$y = 100(\text{allclients} - \text{callclients} + A(x_1 / \text{FAQ}) + B(x_2 / \text{topics})) / \text{allclients}$$

where:

- *allclients* is a constant representing the number of clients. Its value is 1000;
- *callclients* is a constant that represents the number of clients with problems that call for help. Its value is 320;
- *A* is a constant that represents the number of clients with problems that can help themselves with FAQ. Its value is 120;
- *FAQ* is a constant that represents the number of questions and answers included in the troubleshooting section. It has value 35;
- *B* is a constant that represents the number of clients with problems that can help themselves with an on-line discussion forum. Its value is 50;
- *topics* is a constant that represents the number of topics in an on-line discussion forum. It has value 15.

Figure IV-14 presents the propagation functions. It is a function of two variables presented in a 3-D diagram. The figure shows that in both dimensions with the increase of FAQ or topics the percentage of served clients increases but cannot reach 100%.

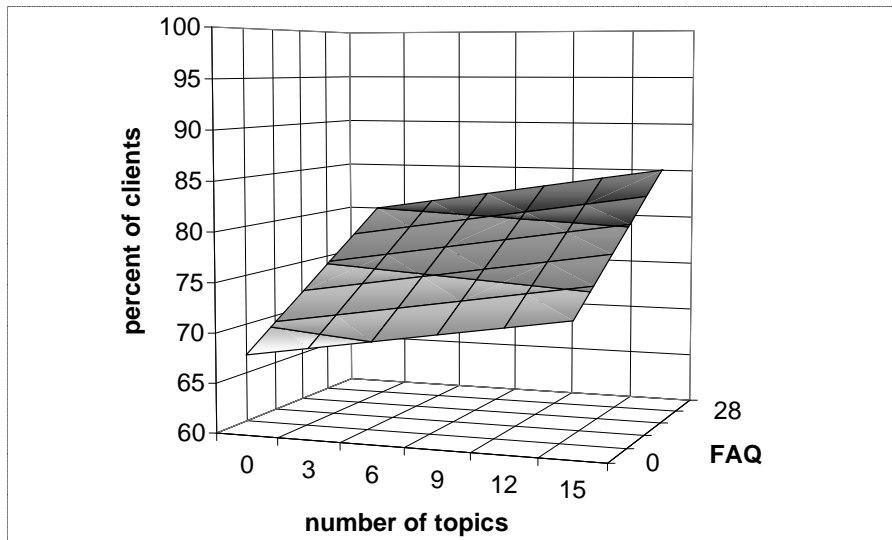


Figure IV-14. Example of a dependency propagation function

In this particular example, the dependency propagation functions cannot be distinguished from decomposition propagation function. The distinction is possible with the evaluation functions. The dependency propagation function will never (in this example) propagate value that the evaluation function of the head goal evaluate to satisfaction, where the decomposition propagation function will.

5.5 Propagation throughout the Model

The calculation of goal satisfaction throughout the model begins from the top-level goals; there is at least one in every model. Usually, a top-level goal has relationships with other goals always as a head goal. Thus, its value is determined from the values of the tail goals in these relationships and one of the three types of propagation functions described above. In case the values of the tail goals are unknown, a recursive calculation is initiated until goals with known values are reached.

In summary, the propagation of values follows the direction of the relationships in the model, while the recursive calculation is triggered from a top-level goal and spread opposing the direction of relationships. The approach works as long as the model is an acyclic directed graph. Next, we discuss the satisfaction propagation in cycles and show that with a simple algorithm of remembering the visited nodes the approach works also in cyclic directed graphs.

5.6 Propagation in Cycles

Cycles may include an arbitrary number of goals and any type of relationships. We discuss first the simple case of a two-goal cycle with only causal relationships; later, we

generalize. Figure IV-15 presents an abstract example of a two-goal cycle. The value of **G1** is causally dependent on the value of **G2**, which negative dependency propagation function is captured by the function $fdep_1$. Similarly, **G2** has a negative dependency propagation function with **G1**, captured by the function $fdep_2$. The explanation below uses the symbols from the figure.

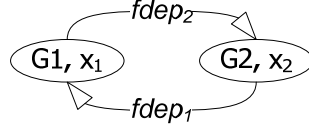


Figure IV-15. Abstract example of a two-goal cycle with dependency relationships

The value of x_1 (see Figure IV-15) cannot be calculated because it is dependent on x_2 which in turn is dependent on x_1 . The cyclic propagation must terminate and for the value of x_1 this will stop if $x_1 = fdep_1(fdep_2(x_1))$; i.e., the composition $fdep_1 \circ fdep_2$ of the two propagation functions has a fixed point. Correspondingly, the propagation value for x_2 will stop if $x_2 = fdep_2(fdep_1(x_2))$; i.e., the composition $fdep_2 \circ fdep_1$ of the two propagation functions has a fixed point. Usually, we calculate the fixed point of only one of the two compositions, e.g. $fdep_1 \circ fdep_2$. In Annex - Fixed points on page 73, we prove that the existence of a fixed point for $x_1 = fdep_1(fdep_2(x_1))$ is sufficient to guarantee the existence of at least one fixed point for x_2 . Thus, the existence of a fixed point for one of the variables is sufficient to guarantee that the cyclic propagation of values will terminate.

Let us assume that the calculation process needs the value of **G1** to propagate it to a third goal not shown in Figure IV-15. In the simplest case, the value of x_1 is initially given and we do not need to calculate it. In more complex cases though, the value of x_1 is unknown. Moreover, the value of x_2 may be also unknown. Thus, there are four possible scenarios:

- The value of x_1 is given, while the value of x_2 is unknown. In this case, the calculation may continue without taking into account the goal **G2**. Nevertheless, it must hold that $x_1 = fdep_1(fdep_2(x_1))$ to ensure that we have a correct model.
- The value of x_1 is unknown but the value of x_2 is given. In this case, the value of x_1 is calculated with the $fdep_1$ function and checked for correctness with the $fdep_2$ function. That is, the value of x_1 is equal to $fdep_1(x_2)$ and $x_2 = fdep_2(fdep_1(x_2))$ must hold.
- The values of x_1 and x_2 are given. In this case, the calculation may continue without taking into account the goal **G2**. Nevertheless, it must hold that $x_1 = fdep_1(x_2)$ and $x_2 = fdep_2(x_1)$ to ensure that the goal model is consistent.
- The values of both x_1 and x_2 are unknown. In this case, if $fdep_1 \circ fdep_2$ does not have a fixed point then the goal model is wrong. If $fdep_1 \circ fdep_2$ has a single fixed point then the value of x_1 is the fixed point. Finally, if the $fdep_1 \circ fdep_2$ has more than one fixed point then all possible values are considered. The case in which the values of both x_1 and x_2 are unknown is feasible. Its occurrence is a sign that somewhere in the model a goal, which normally will propagate a value to at least one of the goals in the cycle, did not receive its initial value. The consequence of

assuming a value determined by the fixed-point calculation is that such a value may be practically not possible to achieve.

The same line of reasoning holds for cycles with more than two goals and not only dependency relationships. Figure IV-16 shows an abstract example of a three-goal cycle with one substitution and two dependency relationships. Depending on the values that are given, we can (1) check the model with the given values and (2) calculate the unknown values. This is possible if one of the compositions of the propagation functions has a fixed point; e.g., $x_1 = fdep_1(fdep_2(fsub_3(x_1)))$. The existence of a fixed point for one composition is sufficient to guarantee the existence of at least one fixed point for any other composition. Annex - Fixed points on page 73 proves that statement.

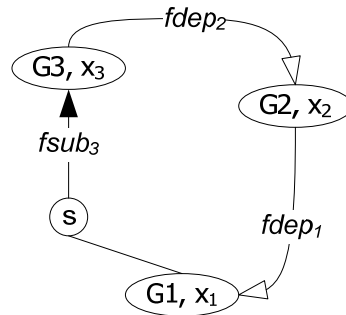


Figure IV-16. Abstract example of a three-goal cycle with two dependency and a substitution relationships

The same results can be generalized for cycles with multivariate propagation functions. Figure IV-17. shows an abstract example of a two-goal cycle with one decomposition and one dependency relationship. As in the other cases, model checking and calculation of values is possible if one of the compositions of the propagation functions has a fixed point, e.g., $x_1 = fdep_1(fdec_2(a, x_1))$. In Annex - Fixed points on page 73, we prove that the existence of a fixed point for one composition is sufficient to guarantee the existence of at least one fixed point for any other composition. That is, the cyclic propagation of values terminates. In the particular case of cycles with multivariate function, a fixed point has to be calculated for every value of the variables not in the cycle.

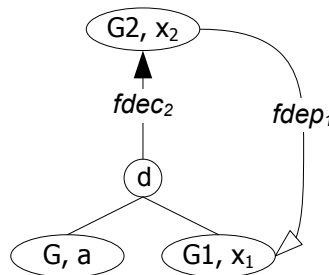


Figure IV-17. Abstract example of a two-goal cycle with a dependency and a decomposition relationships

Example of Fixed Point Calculation. A proper propagation of values through the model can be ensured if the propagation of values in a cycle has a fixed point. That is, (i) starting with the value of a particular goal, (ii) propagating that value through the cycle of goals, and (iii) arriving at the original goal with a final value, the initial and the final value of the original goal are the same. Figure IV-18 shows a goal model where the fixed point approach can be applied. Further in the example, we focus on goals **G3** and **G4** and the dependency relationships, $fdep_3$ and $fdep_4$, between them.

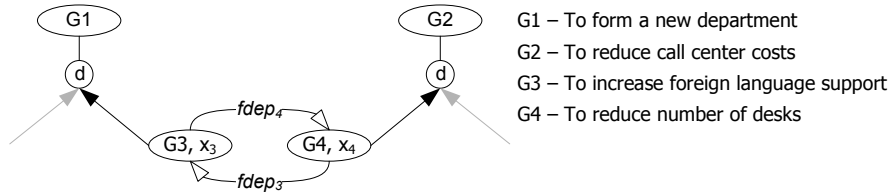


Figure IV-18. Example of a cycle

The two goals **G3** and **G4** are operationalized with the following variables and domains. (The evaluation functions are not specified because they are irrelevant for the propagation of values.)

1. Goal G3: To increase foreign language support
 - goal variable: x_3 – number of supported languages
 - variable domain: Natural numbers
2. Goal G4: To reduce number of desks
 - goal variable: x_4 – number of desks
 - variable domain: Natural numbers

The value propagation functions assigned to the dependency relationships are:

$$x_3 = fdep_3(x_4) = init_desks + 6\sqrt{x_4/8},$$

where:

- $init_desks$ is a constant represent the starting number of desks. Its value is 1;
- $6\sqrt{1/8}$ is arbitrary chosen as an example;

and

$$x_4 = fdep_4(x_3) = \sqrt{x_3}.$$

The graphical representation of the two propagation functions is given in Figure IV-19. The fixed point is reached when the desks x_4 are 2 and the languages x_3 are 4.

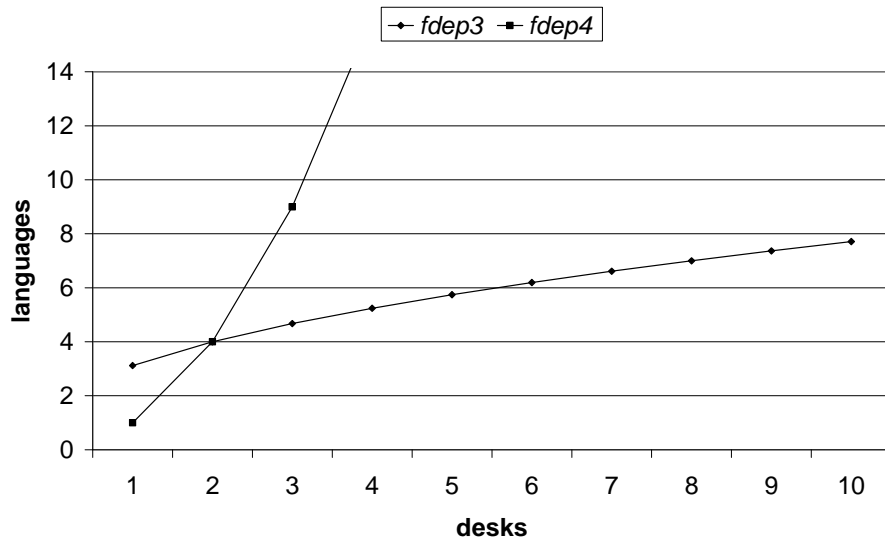


Figure IV-19. Example of a fixed point

6 Related work

Our goal models are theories about causal relationships among goal variables. We use it to represent requirements for or capabilities of a system. A requirements goal model has evaluation functions assigned to variables to tell us when a goal is achieved. A capability goal model has values set to variables to tell us what can be achieved.

Goal-oriented requirements engineering (GORE) is an area of research that uses goal models to represent requirements at various development stages of a system. Below, we review some of the approaches that share properties with ours.

*i** [72] is a goal-modelling technique that focuses on the ‘early-phase’ of the requirements engineering. It is meant to answer how the intended system would meet the organizational goals, why the system is needed, what alternatives might exist, what the implications of the alternatives are for various stakeholders, and how the stakeholders’ interest and concerns might be addressed. The technique includes modelling concepts to represent actors, tasks, resources, goals, and soft-goals, and relationships between these; where soft-goals are those goals that are difficult to objectively specify and to measure. Our approach includes actors, goals, and soft-goals but does not have a notion of tasks and resources. The most important difference is in the relations between goals. *i** gives means-to-achieve-an-end semantics to relationships between goals; whereas, our approach models causal relationships among the variables operationalizing the goals.

TROPOS [11] is a project that researches, among others, the applicability of *i** to multi-agent systems. The project builds up on *i** and NFR [43] reasoning with goals. Giorgini, P. et. al. [24] propose a technique to reason with partially satisfied goals. In their quantitative approach, each goal is assigned a value from 0 to 1 to represent the

evidence of satisfiability of a goal. The types of relationships among goals are functions that calculate the satisfiability evidence of head goals from tail goals. Although we have a similar approach to satisfaction calculation, we differ in the domain of value assigned to goals and propagation functions. In TROPOS, evidence of satisfiability has a value from 0 to 1, which can be interpreted as, e.g., a probability of a goal to be satisfied or percentage of satisfaction. In our approach, goal variables have a domain and value corresponding to the goal that they operationalize. This enables designers and specialists to express and read their concerns in the particular domain. Moreover, the evaluation functions in our approach provide a measure of satisfaction of goals. Consequently to the choice of representing the values assigned to goals, TROPOS has types of propagation functions per goal relationship; i.e., each function has a fixed formula per type of relationship. In our approach, each relationship has a propagation function that matches the causal relationship between the goal variables.

KAOS [36] is a framework for modelling, specifying, and analysing requirements. It uses temporal logic to classify goals and relations among these. The goal modelling refines goals to leaf-goals that are assignable to actors and measurable. Using this, the design method from system goals to software architecture [37] is analogous to our approach of matching goals and pattern capabilities (see for explanation Chapter V Selection Procedure). Both match ability of an actor (in KAOS) or role (in a pattern capability model from our approach) to satisfy a particular goals. In an extension of the KAOS method, Letier, E. and Lamsweerde van, A. [39] propose an approach to reasoning with partial goal satisfaction that steered the direction taken in our goal-modelling technique. The authors put forward a formalization of goals with variables and objective functions from the universe of discourse of the goal. Further, they suggest propagation functions assigned to relationships between goals. In our approach, we restrict the variables and objective (evaluation) functions to one and allow the relationships to be not only a refinement between goals but any causal relationship between the variables.

7 Summary

This chapter presented a goal modelling approach that we use to represent stakeholder requirements and capabilities of design patterns. The resulting goal and capability models are used to select relevant pattern. This chapter answers partially our research question **Q2.2.1: How to identify the relevant design patterns given a set of requirements?** The next chapter, Chapter V Selection Procedure, provides a selection procedure, which completes the answer of **Q2.2.1**. It describes in details the application of goal and capability models.

Annex - Fixed points

This appendix contains the formal proofs of four assertions about the existence of fixed points. The assertions are made about the values of goal variables in a cyclic value

propagation relationship. The assertions cover cases with a varying number of goals in the cycle and various types of relationships.

Assertion 1. If one of the variables in a two-goal cycle has a fixed point value then the other variable has at least one fixed point value. Formally:

$$\exists x_1 : x_1 = f_1 f_2(x_1) \Rightarrow \exists x_2 : x_2 = f_2 f_1(x_2)$$

Proof:

$$x_1 = f_1 f_2(x_1) \tag{1}$$

$$x_2 \stackrel{def}{=} f_2(x_1) \tag{2}$$

$$f_2 f_1(x_2) \stackrel{(2)}{=} f_2 f_1 f_2(x_1) \stackrel{(1)}{=} f_2(x_1) \stackrel{(2)}{=} x_2$$

Assertion 2. If one of the variables in a cycle with arbitrary number of goals has a fixed point value then any other variable has at least one fixed point value. Formally:

$$\exists x_1 : x_1 = f_1 f_2 \dots f_i \dots f_n(x_1) \Rightarrow \exists x_i : x_i = f_i \dots f_n f_1 \dots f_{i-1}(x_i)$$

Proof:

$$x_1 = f_1 f_2 \dots f_i \dots f_n(x_1) \tag{1}$$

$$x_i \stackrel{def}{=} f_i \dots f_n(x_1) \tag{2}$$

$$f_i \dots f_n f_1 \dots f_{i-1}(x_i) \stackrel{(2)}{=} f_i \dots f_n f_1 \dots f_{i-1} f_i \dots f_n(x_1) \stackrel{(1)}{=} f_i \dots f_n(x_1) \stackrel{(2)}{=} x_i$$

Assertions 1 and 2 assume that all value propagation functions are functions on one variable. Below, we relax this constrain and prove assertions 1 in case of multivariate functions. We assume assertions 2 also hold without proving it.

Assertion 3. In a two-goal cycle with multivariate propagation functions, if one of the goal variables has a fixed point value then the other goal variable has at least one fixed point value for particular values of the remaining function variables. Formally:

$$\begin{aligned} \exists x_1 : x_1 = f_1(a_1, \dots, a_{i-1}, f_2(b_1, \dots, b_{j-1}, x_1, b_{j+1}, \dots, b_m), a_{i+1}, \dots, a_n) \Rightarrow \\ \exists x_2 : x_2 = f_2(b_1, \dots, b_{j-1}, f_1(a_1, \dots, a_{i-1}, x_2, a_{i+1}, \dots, a_n), b_{j+1}, \dots, b_m) \end{aligned} , \text{ where } a_i \dots$$

a_n and b_l . b_m are constant values assigned to function arguments.

Proof:

$$x_1 = f_1(a_1, \dots, a_{i-1}, f_2(b_1, \dots, b_{j-1}, x_1, b_{j+1}, \dots, b_m), a_{i+1}, \dots, a_n) \tag{1}$$

$$x_2 \stackrel{def}{=} f_2(b_1, \dots, b_{j-1}, x_1, b_{j+1}, \dots, b_m) \tag{2}$$

$$f_2(b_1, \dots, b_{j-1}, f_1(a_1, \dots, a_{i-1}, x_2, a_{i+1}, \dots, a_n), b_{j+1}, \dots, b_m) \stackrel{(2)}{=}$$

$$f_2(b_1, \dots, b_{j-1}, f_1(a_1, \dots, a_{i-1}, f_2(b_1, \dots, b_{j-1}, x_1, b_{j+1}, \dots, b_m), a_{i+1}, \dots, a_n), b_{j+1}, \dots, b_m) \stackrel{(1)}{=}$$

$$f_2(b_1, \dots, b_{j-1}, x_1, b_{j+1}, \dots, b_m) \stackrel{(2)}{=} x_2$$

Chapter V

Selection Procedure

Identifying Relevant Patterns by a Goal Model of the Requirements

The chapter completes the answer of research question **Q2.2.1**, which also answers research question **Q2.2**. It describes our selection procedure for value and process patterns. It begins with an example that is later used as an illustration of the selection procedure. The chapter presents the sequence of steps that makes the procedure, which includes (1) the generation of all possible alternatives to satisfy the top-level goals in the goal model of requirements, (2) matching capabilities of patterns and goals from the model, (3) integrating the capability model of each matching pattern into the goal model, (4) propagation of satisfaction values from capabilities to top-level goals, (5) rating the alternative solutions made of various pattern combinations. Each step is explained in detail and demonstrated with an example.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	Chapter V Selection Procedure
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

1 Introduction

A requirements goal model is independent of any design that potentially realizes its goals. We use requirements goal models as a computation structure to measure how well a particular design satisfies the goals in the model. That is, goal models evaluate how well design alternatives meet the expectations of the stakeholders.

A design alternative is a potential solution specification of the system under development. It consists of design fragments each of which is selected to be a part of the design alternative because it satisfies a goal in the goal model.

Recalling from Chapter II Design Knowledge in Existing e-Business Models, Section 4 Documenting Patterns, patterns are design fragments annotated with capability models that represent what goals a fragment can achieve and how well. More, capability models are goal models with values assigned the variables. This means that patterns are evaluated as potential fragments to reuse based on their capability models.

A pattern is selected to be a part of a design alternative if a capability from its capability model matches a goal from the requirements goal model. The matching goal from the requirements goal model receives the value of the matching capability. The values from all selected patterns to be part of a design alternative are propagated to the top-level goals in the requirements goal model. The design alternative achieving the best values for the top-level goals is chosen for further synthesis into a solution specification.

The sequence of steps to select patterns as parts of a design alternative and to determine the best alternatives is called a selection procedure¹. This chapter describes this selection procedure.

2 The Running Example

For explanation purposes, we present an example case which we use throughout this chapter. The case is about a business, e.g. a software company, that produces a certain product. The business wants to develop an online product support system. The stakeholders capture their requirements in a requirements goal model. Additionally, the designers use a library containing three design fragments. Each design fragment is represented by its capability model. The intention is to reuse the existing design knowledge in the new design by selecting appropriate fragments from the library.

Figure V-1 shows the goal model of the system under development. The most important requirement is captured in the top-level goal **G1: To offer online technical support**. The goal **G1** is decomposed into three sub-goals, the first of which, **G2**, is dependent on two other goals **G3** and **G4**. The second sub-goal of goal **G1**, **G5**, is further decomposed into two sub-goals, **G6**, and **G7**. The last of the sub-goals of goal **G1**, **G8**, is not related to other goals.

¹ The term *procedure* is intentionally chosen to avoid overloading the term *process* as it is used with another meaning throughout this dissertation.

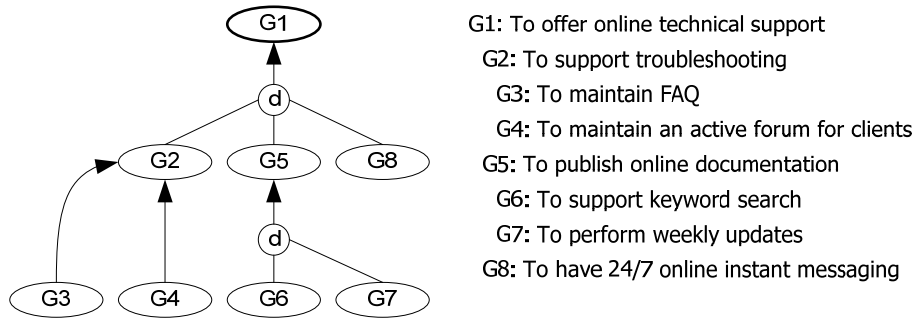


Figure V-1. Requirements goal model of the running example

The goals from the goal model are operationalized by assigning a variable to each of them. Table V-1 presents the goals, the goal variables and the domains of the goal variables.

Table V-1. Operationalization of the goals from the running example

Goal code	Goal	
Variable code	Variable	Variable domain
G1	To offer online technical support	
v ₁	Quality of service	Real numbers between 0 and 1
G2	To support troubleshooting	
v ₂	Percentage unanswered question	Percentage
G3	To maintain FAQ	
v ₃	Number of question and answers	Natural numbers
G4	To maintain an active forum for clients	
v ₄	Percentage active one-week-old topics	Percentage
G5	To publish online documentation	
v ₅	Percentage of inaccessible pages	Percentage
G6	To support keyword search	
v ₆	Percentage not indexed pages	Percentage
G7	To perform weekly updates	
v ₇	Number updates	Natural numbers per year
G8	To have 24/7 online instant messaging	
v ₈	Online hours	Natural numbers per week

The goal model above captures the requirements of the software company. This is our starting point. The resources we use to synthesize a system specification are organized in a library of design patterns. Below, we describe only the capability models of the available design patterns. We denote a capability with a capital letter C and a capability model with capital letters CM.

Figure V-2 contains the capability model of a design fragment called *Sustain Forum*. The fragment stimulates online users to spend time in the forum answering questions to other users. They are attracted by the possibility to win a prize if they are one of the most active forum members. The top-level goal **C1** is decomposed into two sub-goals, **C2** and **C3**. Table V-2 contains the operationalization of the capability model and the achieved values of the variables.

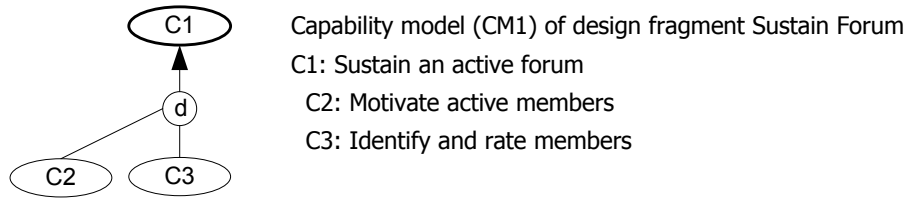


Figure V-2. Capability model (CM1) of design fragment Sustain Forum

Table V-2. Operationalization of the capabilities from capability model 1

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Sustain an active forum		
a ₁	Percentage active one-week-old topics	Percentage	73
C2	Motivate active members		
a ₂	Number of presents	Natural numbers per month	10
C3	Identify and rate members		
a ₃	Anonymous logins	Natural numbers per week	55

Figure V-3 contains the capability model of a design fragment called *Online Help*. To guarantee usefulness of the information, the fragment maintains up-to-date information and structures the information for ease of access. This is captured in the two goals, **C2** and **C3**, into which the top-level goal **C1** is decomposed. Table V-3 contains the operationalization of the capability model and the achieved values of the variables.

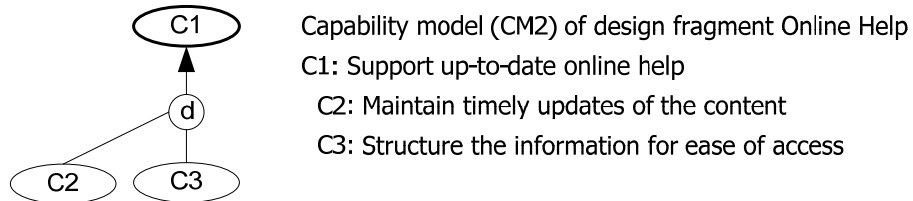


Figure V-3. Capability model (CM2) of design fragment Online Help

Table V-3. Operationalization of the capabilities from capability model 2

<i>Capability code</i>		<i>Capability</i>	
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Support up-to-date online help		
a ₁	Percentage of inaccessible pages	Percentage	7
C2	Maintain timely updates of the content		
a ₂	Update period	Natural numbers of days	3
C3	Structure the information for ease of access		
a ₃	Number of topics	Interval of natural numbers	[8..19]

Figure V-4 contains the capability model of a design fragment called *Call Center*. The fragment implements an online helpdesk solution for customers. The model contains only one goal, which is the top-level goal; (the details reflecting other aspects of the fragment are not shown in the capability model.) Table V-4 contains the operationalization of the capability model and the achieved value of the variable.

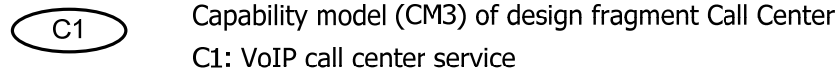


Figure V-4. Capability model (CM3) of design fragment Call Center

Table V-4. Operationalization of the capabilities from capability model 3

<i>Capability code</i>		<i>Capability</i>	
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	VoIP call center service		
a ₁	Online hours	Natural numbers per week	40

3 Selection Procedure

Below, we describe how design fragments are selected to be part of a design alternative and how a design alternative is selected to be a solution specification of the system under development.

To determine a solution specification means to evaluate all design alternatives and select the one that best satisfies the top-level goals. To make the evaluation possible, each alternative has to propagate the values of its capabilities to the top-level requirements goals. This, correspondingly, requires that every design fragment assigns its capability values to goals in the goal model.

To form a design alternative means to select certain design fragments by matching their capability models with the goal model. The appropriate design fragments are

difficult to determine because fragments have complex relationships with the goal model and among themselves. Therefore, the selection procedure considers all combinations of relevant design fragments¹.

3.1 Sketch of the Selection Procedure

Here, we sketch the selection procedure and outline the problems preventing us from developing an automatic selection. Later, we describe each step in detail.

Figure V-5 shows the selection procedure in an UML Activity Diagram, where each step of the procedure is represented by an activity. In the following description, we assume no difference between a step in the procedure and an activity in the activity diagram. The objects represent input and output data passed between the activities. The notes in the diagram explain the semantics of the data.

The first activity to execute is the *Generation* activity. It receives as input data the goal model representing the requirements for the networked businesses. The output is a list of alternative goal models, where an alternative goal model represents one of many possible ways to achieve the top-level goals. An alternative goal model is a data representation in the goal-modelling notation that contains the same top-level goals and a sub-set of remaining goals in the goal model.

The *Matching* activity is executed for each alternative goal model in the list. It tests one by one all leaf-goals in the alternative goal model for a match with capabilities of the design fragments. The result from the Matching activity is a list of four-tuples. This output data describes the matched goals and capabilities. The list is the input for the Integration activity.

The *Integration* activity adds goal relationships among the alternative goal model and the capability models of matching design fragments. The output is an integrated alternative goal model. This is, then, the input for the Propagation activity.

The *Propagation* activity calculates the values of the top-level goals in an integrated alternative goal model from the capability values. It produces as an output an alternative solution capability model which is an integrated alternative goal model with calculated values for the top-level goals.

The *Evaluation* activity collects all alternative solution capability models and rates these. The output of the activity is the result of the procedure, namely a ranking of alternative solution capability models.

¹ A relevant design fragment is a fragment that has a capability model with capabilities matching the goals from the goal model.

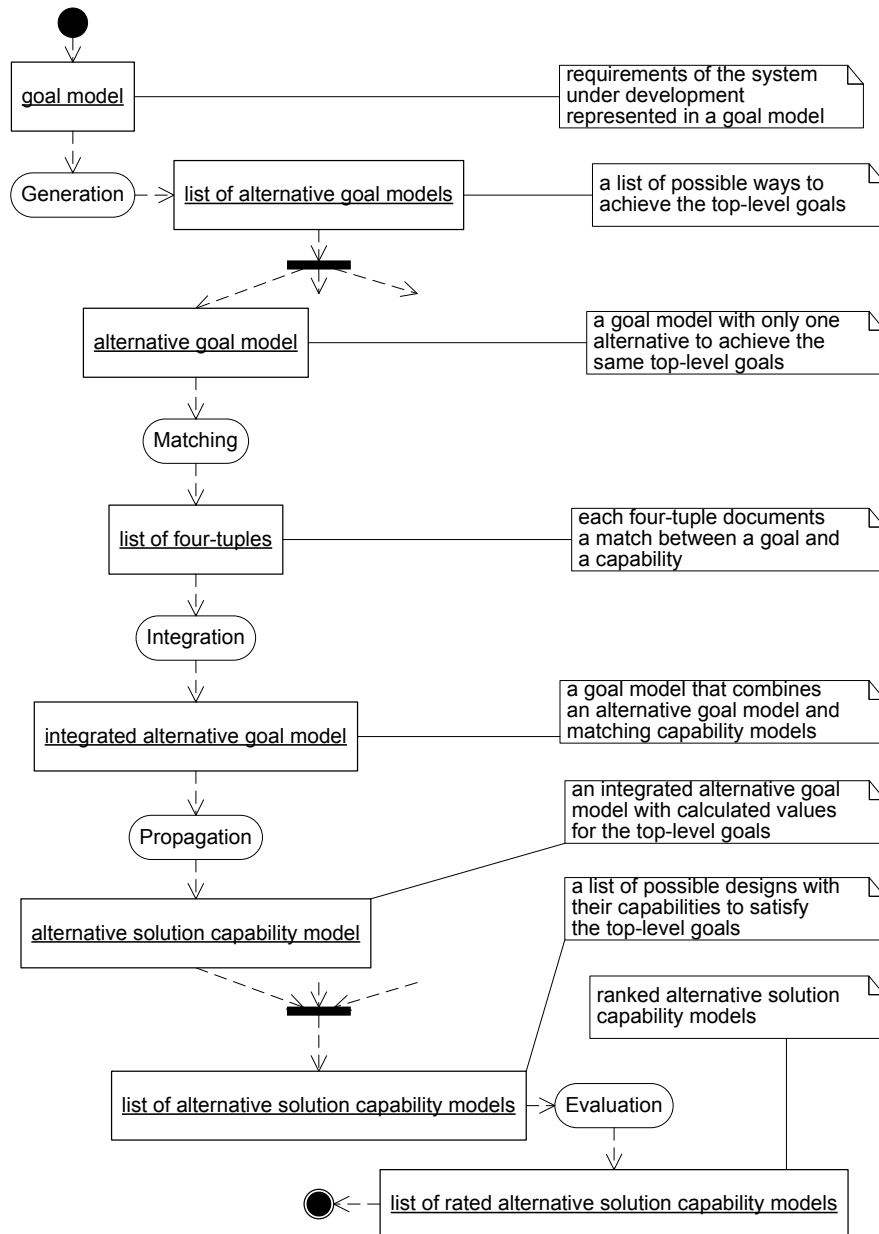


Figure V-5. Activity diagram of the selection procedure

Below the procedure is discussed in more detail:

Step One: Generation. This step traverses the goal model to allow matching of pattern capabilities with every goal of the goal model.

Ideally, every design fragment is represented by a single capability that matches a single leaf-goal in the goal model. This means that the level of detail in the goal model exactly matches the level of abstraction of the capability models. (Recalling, we introduced levels of abstraction to mark the top-level goals and capabilities as most abstract and leaf-goals as most detailed.) In reality, a capability of a design fragment can match any goal from the goal model. Matching a goal leads to setting the value of the variable equal to the value of the capability. In case the goal is a head-goal in a relationship then the relationship would be ignored. To avoid this, we traverse the goal model starting from the top-level goals down to leaf-goals until we match the level of abstraction of the capability models.

The Generation step takes as an input the goal model and produces as an output a list of new goal models. We call the new goal models alternative goal models because each of these shows a way to achieve the top-level goals. Each alternative goal model is a path from some goals to the top-level goals. In this way, every goal from the goal model is a leaf-goal in at least one alternative goal model. In the subsequent steps of the selection procedure, all leaf-goals of each alternative goal model are matched with capabilities from the capability models of the available design fragments.

For example, the two extremes of alternative goal models are an alternative goal model that contains only the top-level goals and an alternative goal model that equals the entire original goal model itself. The former alternative goal model assumes that there are available design fragments to offer a ready solution to the problem at hand. The latter alternative goal model assumes that the design fragments are at finer granularity and match with the leaf-goals from the goal model. In-between, there are many alternative goal models that resemble the original goal model but have certain branches cut. In the detailed description below, we explain how all alternative goal models are generated.

The step requires no manual work.

Step Two: Matching. This step matches goals from the goal model with capabilities representing design fragments.

A goal and a capability match if they refer to the same desired phenomenon¹. Matching them is not a straightforward task because the goal and capability models are developed by different people with different purposes in mind. Incompatibility of vocabularies, abstraction levels, knowledge, perspectives and contexts hinder the automation of the matching process. In the database area, a similar problem of finding equivalent nodes in trees is known as the Schema matching problem. Various approaches to this problem exist [16, 47] and all of them require human intervention.

Our matching problem is even more complex as the goal and the capability models are directed graphs, potentially with cycles. To simplify the matching, we make a number of assumptions which we explicate in the detailed description of the Matching step further below.

The assumptions reduce the manual work per match (between one goal and one capability.) Their impact is significant because the number of matches is potentially high.

¹ Our definition of a goal is a desired phenomenon by a certain organization or individual, where a phenomenon is a state of reality, an activity, a fact or an event (Chapter IV Goal-modelling, Section 1 Introduction)

The actual number of matches¹ is the number of goals in the goal model multiplied by the sum of all top-level capabilities in all capability models.²

Despite all simplifying assumptions, the Matching step (i) remains complex; (ii) requires manual work; and (iii) is executed many times.

The input for the Matching step is a single alternative goal model: the step is executed per alternative goal model. The output is a list of four-tuples: one³ list per alternative goal model. Every four-tuple contains the following elements⁴: a *goal*, a *capability*, a *value*, and a *design fragment*. The *goal* element represents a leaf-goal from the alternative goal model. This goal matches a capability, captured in the *capability* element. The particular value, coming from the matching capability, assigned to the variable of the matching goal is recorded in the *value* element. The *design fragment* element represents the design fragment that achieves the matching capability. For example, an alternative goal model has two leaf-goals: **LG1** and **LG2**. There is one design fragment with a capability model CM. The capability model has two capabilities **C1** and **C2** the values of which are 5 hours and 53 pages per minute, respectively. The result of the Matching step is that capability **C2** matches leaf-goal **LG2**. The resulting list of four-tuples looks such as: <LG1, , , >, <LG2, C2, 53, CM>.

The size of the four-tuples list is equal to the number of leaf-goals in the currently processed alternative goal model; i.e. every leaf-goal, and only a leaf-goal, from the alternative goal model has one and only one corresponding four-tuple in the list.

Not all alternative goal models are considered in the next step. If an alternative goal model has all its leaf-goals matched with capabilities then the corresponding design fragments form a design alternative. If the leaf-goals of an alternative goal model cannot be matched with any capability then the alternative goal model is discarded as not leading to a design alternative⁵.

Step Three: Integration. This step extends the goal model with capability models by adding relationships between non-matching goals and capabilities. The matching capability replaces the matching goal; the remaining capabilities from the capability model could relate to goals from the alternative goal model.

A design fragment, correspondingly its capability model, can affect the values of more than the matching goal from the goal model. Moreover, a design fragment can be in a conflict with another matched design fragment. These specificities require integration of

¹ The actual number of matches is the number of goals in the goal model multiplied by the total number of all capabilities in all capability models. Later with one of our assumptions, we reduce the number of matches to the number of goals in the goal model multiplied by the total number of only top-level capabilities in all capability models.

² The Matching step is executed as many times as alternative goal models there are, multiplied by the number of goals of each alternative goal model. The number of iterations of the Matching step is not the actual number of matches. A single execution of the Matching step makes use of past executions and do not match goals and capabilities that were already matched.

³ We later show that the lists of four-tuples can be more than one per alternative goal model, see Section 5.1

⁴ A four-tuple contains placeholders for a goal, a capability, a value and a design fragment. This is necessary as some of the elements in the four-tuple may not be specified.

⁵ The condition for an alternative goal model to form a design alternative is that all leaf-goal variables have values. Later, we introduce leaf-goals with predetermined values which allows an alternative goal model to form a design alternative even if not all the leaf-goals are matched.

the capability model into the goal model by adding relationships; i.e., we extend the goal model by adding capability models and linking these with the rest of the model.

The Integration step includes problems known in the database area as Schema integration problems [16, 47]. The existing approaches to these problems require human intervention.

The Integration step adds information to the goal model in the form of causal relationships between the variables of capabilities and goals. This information is not present in the capability model and has to be obtained on a case-by-case basis. The step has to be executed as many times as matches between a goal and a capability model there are; this is a potentially high number. The number of iterations and the need for additional information make this step the bottleneck of the selection procedure. Further below, we propose a trade-off solution between precision and pragmatism by postponing the Integration step.

The input of the Integration step is a list of four-tuples containing information about the matched leaf-goals and capabilities. Implicitly, the Integration step receives the alternative goal model from which the list of four-tuples resulted. Furthermore, it derives the capability models that have a matching capability from the list of four-tuples. Thus, the Integration step merges an alternative goal model with a number of matching capability models to produce a new goal model called an integrated alternative goal model.

Step Four: Propagation. This step propagates the achieved value by one design alternative to the top-level goals.

Once all relevant design fragments are identified and their capability models are integrated with the goal model, the variable values are propagated through the goal model to the top-level goal. The propagation is described in Chapter IV Goal-modelling Section 5 Satisfaction Calculation on page 60.

The input of the Propagation step is an integrated alternative goal model in which only some of the goals have values. The output is the same integrated alternative goal model with variable value propagated to the top-level goals. We call such an integrated alternative goal model an alternative solution capability model.

The Propagation step is executed per every design alternative. Manual intervention is required only to break cycles without a fix-point.

Step Five: Evaluation. This step rates all design alternatives and selects the best one with respect to the satisfaction of the top-level goals in an alternative solution capability model.

In a nutshell, the step selects the best of relatively many options based on a limited number of parameters, where the options are the alternative solution capability models and the parameters are the top-level goals. This is solving a Multiattribute Problem from the area of Multiple-Criteria Decision Making [42: page 17]. Various approaches have been proposed as a solution to this problem, e.g.: Scoring Method, Multiattribute Value Function, Multiattribute Utility Function, Analytic Hierarchy Process, ELECTRE I, II, III and Data Envelopment Analysis (for comparison see [42: pages 78-79 and 88-89]). We use Scoring Method because of its simplicity; nevertheless, any other method is a valid choice.

The input of the Evaluation step is a list of alternative solution capability models. Each of the models represents one design alternative. The output of the step is the same list where every element is assigned a number representing its score with respect to the evaluation criteria.

The Evaluation step is executed per every design alternative and requires no manual work.

3.2 Trading off Generality and Precision against Automation

The Matching and Integration activities require the most manual work. They consume a lot of time and effort for every alternative goal model to be assessed and only one to be selected at the end. This is undesirable considering the potentially big number of alternative goal models¹. To improve the performance of the process, we make trade-offs such as: (1) trading representation details in the Matching step for automation and (2) trading precision in the Integration step for speed and ease.

To automate the Matching step, we restrict the languages used to describe the goals and capabilities. By using a shared vocabulary between requirements goal models and capability models to represent goals, evaluation functions, variables and domain, we eliminate the need of a semantic match of goals done by people. In this way, we reduce the matching problem to syntactic matching of strings.

The trade-off in the Integration step is between the precision of the values of the top-level goals and the efficiency of the selection procedure. The Integration step adds relationships between non-matching goals and capabilities, which affects the propagation functions in the goal model. We assume that the information added to the goal model by the relationships between non-matching goals and capabilities is significantly less than the information added by matching capabilities replacing goals. In the trade-off, we ignore the information from added relationships by postponing the execution of the step. The integration is moved to a later stage in the selection procedure, where it is applied on fewer design alternatives.

Consequently, we introduce an additional step called Pre-evaluation step to filter design alternatives that poorly satisfy the top-level goals. The step is the same step as the Evaluation step with the difference that it produces several design alternatives as potential solution specifications. The Integration step is applied on the best scoring design alternatives that come out of the Pre-evaluation step.

Figure V-6 shows the improved selection procedure. The difference with the selection procedure from Figure V-5 is that the execution of the Integration activity does not follow the Matching activity. The improved procedure ignores the additional information from the integration and proceeds with the propagation of values from the leaf-goals to the top-level goals. The new activity, the Pre-evaluation activity, is executed over the list of alternative solution capability models. It selects a fixed number of best alternative solution capability models for further processing. The limited number of high-scoring alternative designs is considered our best alternative designs.

¹ The number of alternative goal models depends on the number of goals, number of relationships, type of relationships in the goal model. Details can be found in Section 4 Generation of Alternative Goal Models

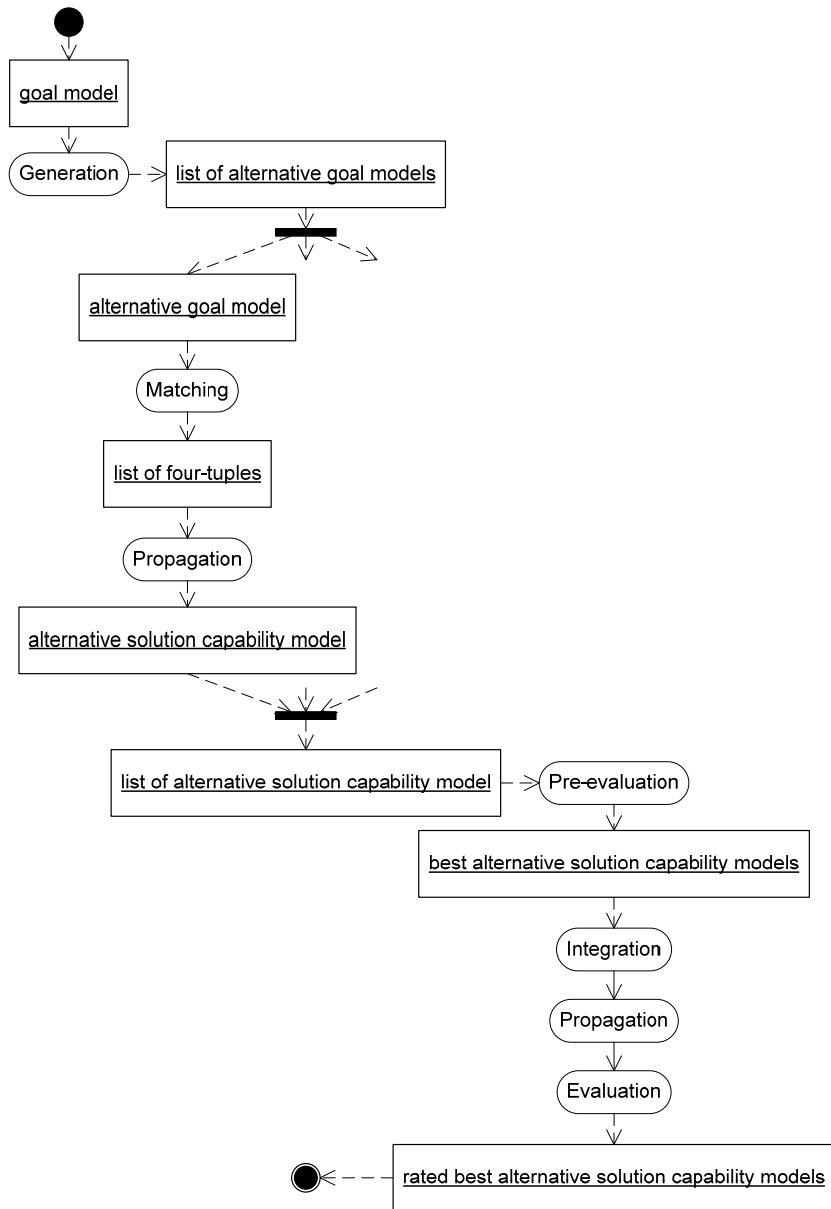


Figure V-6. Activity diagram of the improved selection procedure

The best alternative solution capability models are the input data for the Integration activity. During the integration, the ranking of design alternative may change. Therefore, the Propagation and Evaluation activities are performed second time over the best alternative solution capability models.

The result from the improved procedure is (1) automated Matching step and (2) a parameterizable fixed number design alternatives to undergo the Integration step.

4 Generation of Alternative Goal Models

Naturally, the procedure of identifying suitable design fragments begins with matching of capabilities and goals, which is a difficult task and, for feasibility reasons, requires a number of assumptions (see Section 5.)

The necessity of the Generation step is motivated not only by the complexity of the matching between capabilities and goals but also by the difference in level of granularity of the goal and capability models. It is rare that the refinement of the goal model stops exactly at the level of detail of the capability models in the library. Thus, two disadvantageous situations may occur: (1) the granularity of the leaf-goals in the goal models is coarser than the one of top-level capabilities in the capability models and (2) the granularity of the leaf-goals in the goal models is finer than the one of top-level capabilities in the capability models. In the former case, nothing can be done in the selection procedure; the goal model has to be refined. In the latter case though, if the goal model is traversed from top-level goals to leaf-goals then a match must be attempted at any level of granularity.

For ease of explanation, we assume for the scope of this section that: (1) a goal from the goal model and a capability always match one-to-one and (2) the capability model of every design fragment contains only one capability. Later in the next section, we relax both restrictions.

The Generation step takes as an input a goal model and produces a list of alternative goal models. This is done by traversing the graph behind the goal model. In the process of visiting nodes, we extract sub-graphs that correspond to meaningful goal models. The algorithm we follow is a modification of the breadth-first search algorithm for traversing graphs [65], in which we take into account the semantics of the goal model and the specificities of the graph representing the goal model. The following properties of our goal modelling approach affect the generation algorithm:

- three types of goal relationships. The goal model differentiates three types of relationships between goals and these have different interpretations of the relation between tail-goals and head-goals. The semantics of the relationships is used to distinguish the valid alternative goal models. For example, if a decomposition relationship in the goal model appears without one of the sub-goals in an alternative goal model then the alternative goal model is not valid. A decomposition relationship must be present in all alternative goal models with all of the related goals. The substitution and dependency relationship impose their restrictions on the validity of alternative goal models.
- the goal model is a directed graph (see Chapter IV Goal-modelling, Section 4 Goal Model.) The direction of a goal relationship (the direction of the arc follows the direction of the goal relationship) represents that a tail-goal determines the satisfaction of a head-goal. This means that if a tail-goal is turned into a capability (i.e. if a tail-goal is matched with a capability) then the satisfaction of the head-goal can be determined from the relationship. Whereas, the opposite is

not possible: the satisfaction of a tail-goal cannot be determined from the satisfaction of a head-goal. The consequence for a top-down traversing algorithm is that we visit nodes against the direction of the arcs; i.e., from a given visited node, we continue traversing only nodes linked with arcs pointing at the current node. The reason for going against the direction of the arcs is that, when starting the traversal from a top-level goal, we consider only goals that affect the currently visited node, not goals that are affected;

- the graph is in general¹ cyclic (see Chapter IV Goal-modelling, Section 4 Goal Model.) The presence of cycles in the graph has implications mainly for the implementation of the generation algorithm. The traversal must be interrupted if an already visited goal is selected for a subsequent visit. This breaking of cycles² is inline with the semantics of fixed-points in the goal model: the effect of a relationship (arc) is invariant to the number of iterations through the cycle;
- the graph is in general not connected. Every top-level goal is refined with sub-goals which are not necessarily related to the refinement goals of the other top-level goals. This leads to disconnected sub-graphs, which has implications for the implementation of the generation algorithm. Picking an arbitrary node from a connected sub-graph to continue the traversal after another connected sub-graph is completely visited does not make sense because there is no guarantee that all nodes will be visited. Accidentally, the specificity in the next bullet solves the problem of disconnected graphs;
- each connected sub-graph contains at least one node that corresponds to a top-level goal. This is a consequence of the same reason that causes the disconnected sub-graphs (see previous bullet.) The advantage for the algorithm is that we have a particular node or nodes to start the traversing in the disconnected sub-graph; and
- all top-level goals are equally important. The top-level goals jointly determine the system under development. Therefore, they have to be present in every alternative goal model. The consequence for the generation algorithm is that all nodes corresponding to top-level goals have to be visited simultaneously in the first step of the algorithm. To work around this problem, we introduce a superfluous goal³ which is a composition of the top-level goals; i.e. the superfluous goal is related to all top-level goals in a decomposition relationship (see for illustration Figure V-7 (a).) The superfluous goal is represented in the graph with a node that we call a root. The root is the first node that we visit in our traversing algorithm. As we show later this is consistent with the constraint that the first generated alternative goal model is the one that contains only and all top-level goals. The work-around solves also the problem of disconnected graphs.

¹ 'in general' here means that although most of the goal models may not result in graphs with cycles, our approach works with cyclic graphs.

² By breaking cycles, we mean that the traversal stops and does not go into an endless loop. The cycles remain in the alternative goal models (see Section 4.2.)

³ To have a unified approach in the Generation step, we introduce a superfluous goal even in cases of only one top-level goal.

Figure V-7 presents an example that we use to describe the traversing and generation of alternative goal models. Figure V-7 (a) shows a goal model with two top-level goals, **G1** and **G2**. The model contains both substitution and decomposition relationships: goal **G1** is related to goals **G3** and **G4** with substitution relationship; and goal **G2** is related to goals **G5** and **G6** with decomposition relationship. Because the model has two top-level goals and the corresponding graph is disconnected, we introduce the superfluous goal Sup. The goal Sup is decomposed to the top-level goals **G1** and **G2**. Figure V-7 (b) shows a concise representation of the goal model which we use to illustrate the Generation step.

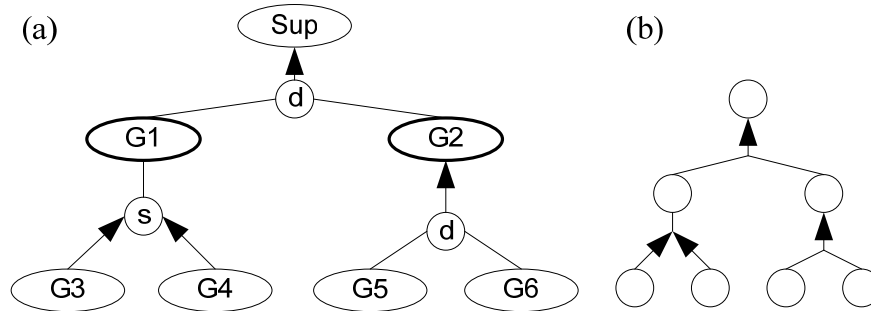


Figure V-7. Illustrative example: (a) – goal model with a superfluous goal Sup; and (b) – concise representation of the goal model from (a)

Below, we refer to the representation in Figure V-7 (b) as to the graph of the goal model. Goals are referred to nodes; goal relationships are referred to arcs. All arcs are considered bilateral and directed.

4.1 Algorithm for Generation of Alternative Goal Models

In a nutshell, the algorithm is as follows: (1) the graph is traversed starting from the root node; (2) in every step, nodes, linked with an outgoing arc to already visited nodes, are visited; (3) in every step, all visited nodes form an alternative goal model. To explain the algorithm in detail, we need to define the following terms. Figure V-8 shows their graphical representation.

- *Label* is a symbol with some meaning attached to a node. The labels are two types: *visited* and *next*. With the labels, we keep track of the progression of traversing and the starting for next visits.
- Label *visited*. The symbol for the label is the letter 'o'. It is placed in the circle representing the node. The meaning is that the traversal algorithm has visited this node. Figure V-8 (a) shows an example of a visited node.
- Label *next*. The symbol for the label is a triangular. It is placed next or under the circle representing the node. The meaning is that in the next step the traversal algorithm will visit the nodes linked to the labelled one. Figure V-8 (b) shows an example of two equivalent ways to label a node with visited and next labels¹.

¹ A node that is labeled with a label next has already a label visited.

- *Neighbouring node (neighbour)* is an unvisited node that is linked with an outgoing arc to a node labelled with visited and next labels. Figure V-8 (c) shows an example of one visited and next node and its neighbour.
- *Marking* is a graph in which some of the nodes have labels. Markings have two representations: (1) *full* – all nodes in the graph are shown with their respective labels (Figure V-8 (d) shows an example of a full marking) and (2) *contracted* – only visited nodes are shown with their respective labels (Figure V-8 (e) shows an example of a contracted marking.) The contracted representation focuses on the starting nodes for the traversal in the next step of the algorithm.

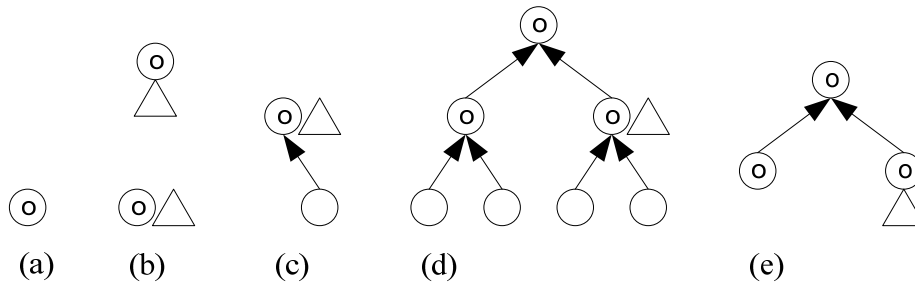


Figure V-8. Labels and markings: (a) – visited node; (b) – node with labels visited and next; (c) – one visited and next node and its neighbour; (d) – example representation of a full marking; and (e) – example representation of a contracted marking of the same marking as in (d)

Figure V-9 shows an UML Activity diagram of the algorithm for generation of alternative goal models. The algorithm starts with an activity that creates a pool for markings, namely the Create and initialize the pool of markings activity. The pool is initialized with one marking and that is always the root node of the input goal model with labels visited and next. The algorithm contains four more activities. These are executed in a sequence for every marking in the pool. The second and the last activities (see the numbering inside the activities of Figure V-9) maintain the pool of markings. Activity number two, Take a marking from the pool, picks indifferently of the order a marking from the pool; whereas, activity number five, Return new markings in the pool, returns newly created markings into the pool, if there are any. The actual generation of alternative goal models is done in activity number three, Generate alternative goal models. The marking received as an input is copied a number of times¹ and a different combination of neighbouring nodes is visited in every copy. The copies are modified markings which are the source for generation of alternative goal models: an alternative goal model is created by extracting a sub-graph of only visited nodes from a particular marking. The copies are sent to the next activity, Generate new markings, to produce new markings. The activity number four generates new markings which are returned to the pool in activity five for further processing. Not every copy can produce new markings.

¹ The exact number is discussed in the detailed description of the activity.

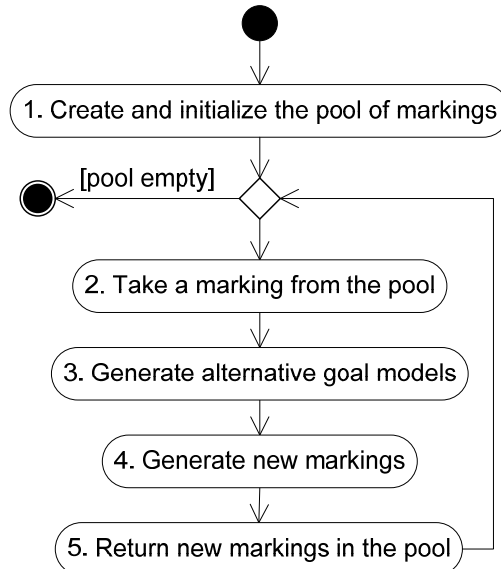


Figure V-9. Activity diagram of the generation algorithm

The two activities that generate the alternative goal models and the new markings depend (i) on the current marking, (ii) on the number of neighbouring nodes in the graph, and (iii) on the types of relationship in the goal model. Below, we describe each of the two activities in detail.

Generation of Alternative Goal Models. The generation works the following way. The current marking is copied several times. In every copy, a different combination of neighbouring nodes are visited. The visited nodes in every copy determine one alternative goal model. The number of copies and the valid combinations of neighbouring nodes depends on:

- the type of relationship with the neighbouring nodes. These determine the valid combinations of neighbouring nodes to visit.
- the number of neighbouring nodes. These are the neighbouring nodes of the nodes labelled with next. They determine the set of nodes to be visited in the current iteration; and
- the current marking. The marking determines (1) the already visited nodes and (2) the starting nodes for new visits. It restricts and gives direction to the traversing;

The bullets above follow a reverse order of influence: the last bullet restricts the possible combination stronger than the first one. Nevertheless, we explain them in the order above for clarity.

Type of Relationship. A goal model has three types of relationships: substitution, decomposition, and dependency. The specific semantics of each one of them determines a different set of alternative goal models. For every type of relationship, we show a small

example of four goals, one head-goal and three tail-goals (see Figure V-10, Figure V-11 and Figure V-12.) For each example goal model, we show a full marking in which only the node corresponding to the head-goal has labels visited and next. Further, we show the copies of the marking with newly visited nodes. Finally, we show the valid newly generated alternative goal models.

- Substitution Relationship.** Figure V-10 (a) shows an example of a goal model containing four goals related with a substitution relationship. The corresponding graph is presented in Figure V-10 (b). The graph of Figure V-10 (b) is also a full marking of the graph as it contains labels visited and next. The marking is such that the next nodes to be visited are the ones corresponding to the sub-goals. The neighbouring nodes are visited independently of each other because the substitution relationship is a relationship in which every sub-goal acts independently. Thus, the marking is copied three times and in every copy, only one of the neighbouring nodes is visited. The copies are the input for the fourth activity in Figure V-9. Figure V-10 (c), (d) and (e) show them. The alternative goal models are derived by removing unvisited nodes from the copies; Figure V-10 (f), (g) and (h) show them.

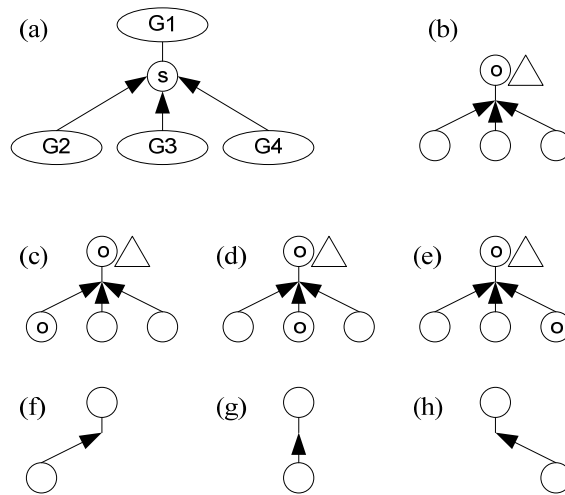


Figure V-10. Alternative goal models derived from a substitution relationship: (a) – goal model of substitution relationship; (b) - full marking of the graph corresponding to the goal model; (c), (d), and (e) – copies of the marking in (b); and (f), (g), and (h) – alternative goal models

- Decomposition Relationship.** Figure V-11 (a) shows an example of a goal model containing four goals related with a decomposition relationship. The corresponding graph is presented in Figure V-11 (b). The graph of Figure V-11 (b) is also a full marking of the graph as it contains labels visited and next. The marking is such that the next nodes to be visited are the ones corresponding to the sub-goals. The neighbouring nodes are visited all together because the decomposition relationship is a relationship in which all sub-goals act jointly. Thus, the marking is copied once and in the copy, all neighbouring nodes are

visited. Figure V-11 (c) shows the copy which is the input for the next activity. The alternative goal model is derived by removing unvisited nodes; Figure V-11 (d) shows it.

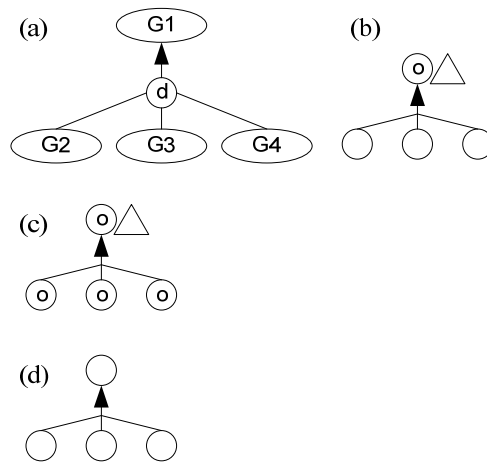


Figure V-11. Alternative goal models derived from a decomposition relationship: (a) – goal model of decomposition relationship; (b) - full marking of the graph corresponding to the goal model; (c) – copy of the marking in (b); and (d) – alternative goal model

- Dependency Relationship.* Figure V-12 (a) shows an example of a goal model containing four goals related with a dependency relationship. The corresponding graph is presented in Figure V-12 (b). The graph of Figure V-12 (b) is also a full marking of the graph as it contains labels visited and next. The marking is such that the next nodes to be visited are the ones corresponding to the sub-goals. The neighbouring nodes are visited in all possible combinations because the dependency relationship is a relationship in which sub-goals can act independently or jointly¹. Thus, the marking is copied seven times and every copy has a different combination of visited neighbouring nodes. Figure V-12 (c), (d), (e), (f), (g), (h) and (i) show these copies which are the input for the fourth activity in Figure V-9. The alternative goal models are derived by removing unvisited nodes from the copies; Figure V-12 (j), (k), (l), (m), (n), (o) and (p) show them.

¹ The number of combinations can be reduced if the independently and jointly acting sub-goals are known. This information is present in the propagation function of the relation. For simplicity of explanation, we assume that we do not have this information. The additionally generated alternative goal models are irrelevant because in the Propagation step the propagation function will not propagate any values for them.

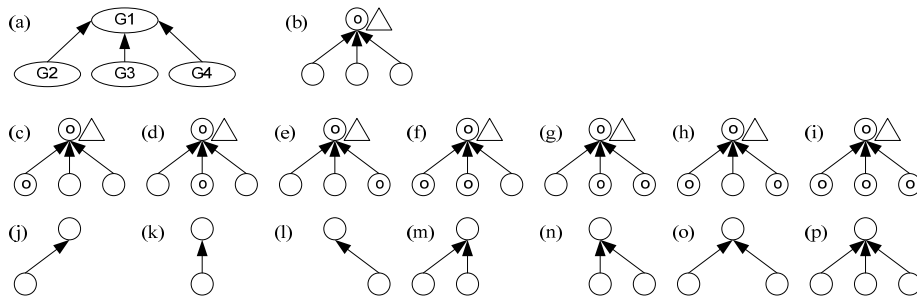


Figure V-12. Alternative goal models derived from a dependency relationship: (a) – goal model of dependency relationship; (b) - full marking of the graph corresponding to the goal model; (c), (d), (e), (f), (g), (h), and (i) – copies of the marking in (b); and (j), (k), (l), (m), (n), (o), and (p) – alternative goal models

In summary, the type of relationship determines the valid combinations of visited neighbouring nodes, independently of the number of nodes. In case of a substitution relationship, all nodes are visited individually in separate alternative goal models; in case of a decomposition relationship, all nodes are visited together in one alternative goal model; and finally in case of a dependency relationship, the neighbouring nodes are visited in all possible combinations.

Number of Neighbouring Nodes and Type of Goal Relationship. The number of nodes to be visited starting from one node with a label next determines the number of generated alternative goal models from the particular marking. For each of the relationships, the number of alternative goal models in case of n neighbouring nodes is as follows:

- substitution relationship: n – each node is visited in a separate model;
- decomposition relationship: 1 – all nodes are visited in one model; and
- dependency relationship: $\sum_{k=1}^n \frac{n!}{k!(n-k)!}$ – the nodes are visited in all

combinations. $(\sum_{k=1}^n \frac{n!}{k!(n-k)!} = 2^n - 1, \text{ see [67]})$

More Than One Next Node. The current marking determines the starting point for further visits. It determines which and how many of the nodes have a label next. Every node with a label next is independent from the others. Moreover, the visits from every node are simultaneous. That is, a valid alternative goal model has at least one visited node from the neighbouring nodes of every next node. The consequence is that the number of alternative goal models generated from one marking is the multiplication of the combinations generated from each next node. Figure V-13 illustrates this.

Figure V-13 (a) shows a goal model. (It is the same model as in Figure V-7 in the beginning of Section 4.) At a certain point in the execution of the generation activity, we have to process the marking shown in Figure V-13 (b). The specificity of this marking is

that there are two nodes with label next. The neighbours of each next node are visited separately. The substitution relationship results in two copies of the marking (not shown in the figure) and the decomposition relationship results in one copy (also not shown in the figure). The total number of copies is the multiplication of the copies from each next node, thus two ($1 \times 2 = 2$). The copies of the marking with the newly visited nodes are shown in Figure V-13 (c) and Figure V-13 (d). They are the input for the next activity. Correspondingly, the two derived alternative goal models are shown in Figure V-13 (e) and Figure V-13 (f).

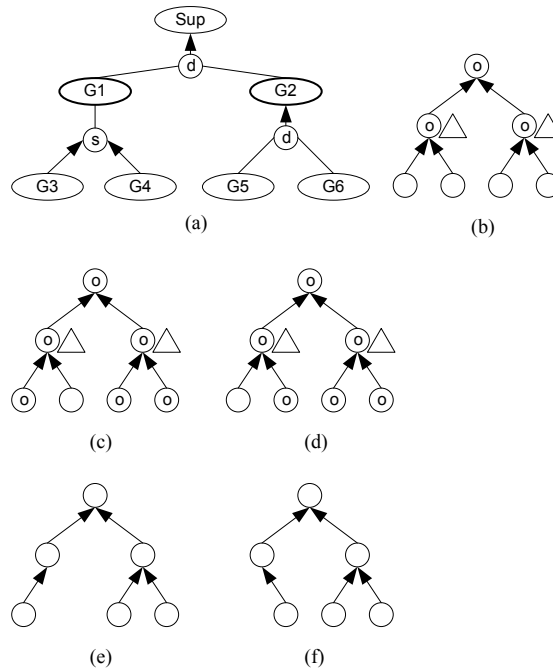


Figure V-13. Example of generation of alternative goal models from marking with two next labels:
 (a) – example goal model, the same as in Figure V-7 (a); (b) - full marking of the graph corresponding to the example goal model; (c) and (d) – copies of the marking in (b); and (e) and (f) – alternative goal models

Summary. Above, we describe the third activity of the generation algorithm (see Figure V-9.) This shows how alternative goal models are generated from a single marking and how the type of goal relationship, the number of neighbouring nodes, and the current marking affect the number and structure of generated models.

Generation of New Markings. The input that this activity receives is a number of copies of a marking generated during the third activity in Figure V-9. The generation of new markings works the following way:

1. The labels next are removed from every of the copies sent by the previous activity.
2. Every newly visited node in every copy is checked for neighbouring node.

3. In case a node has no neighbours, it remains visited but it is excluded from further consideration as a next node.
4. From the remaining newly visited node (the ones that have neighbouring nodes) in every copy, all combinations of nodes receiving the next label are considered.

We illustrate the generation of new markings on behalf of the same example goal models used to illustrate the types of relationships (see Figure V-10, Figure V-11 and Figure V-12.) We assume that every of the three goals **G2**, **G3** and **G4** is a head-goal in a relationship not shown in the example.

Substitution Relationship. Figure V-14 shows the derived marking in case of a substitution relationship. Figure V-14 (a) and (b) are respectively the goal model and the particular marking we want to investigate. The copies produced by the activity Generate alternative goal models are presented in Figure V-14 (c), (d) and (e). These are the input for the activity Generate new markings. Following the five steps listed above, we receive as new markings the markings in Figure V-14 (f), (g) and (h). Every copy has only one newly visited node; therefore, the number of new markings is the same as the number of copies of the original marking.

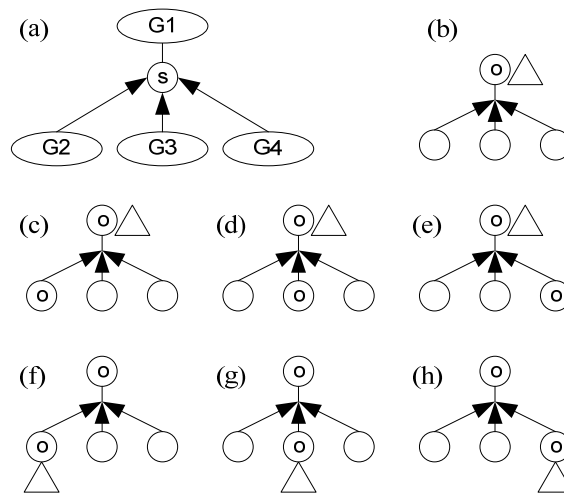


Figure V-14. New markings from a substitution relationship: (a) – goal model of substitution relationship; (b) - full marking of the graph corresponding to the goal model; (c), (d), and (e) – copies of the marking in (b); and (f), (g), and (h) – new markings

Decomposition Relationship. Figure V-15 shows the derived marking in case of a decomposition relationship. Figure V-15 (a) and (b) are respectively the goal model and the particular marking we want to investigate. The copy produced by the activity Generate alternative goal models is presented in Figure V-15 (c). This is the input for the activity Generate new markings. Following the five steps listed above, we receive as new markings the markings in Figure V-15 (d), (e), (f), (g), (h), (i) and (j). The

only copy has all three nodes newly visited; therefore, the number of new markings is seven¹.

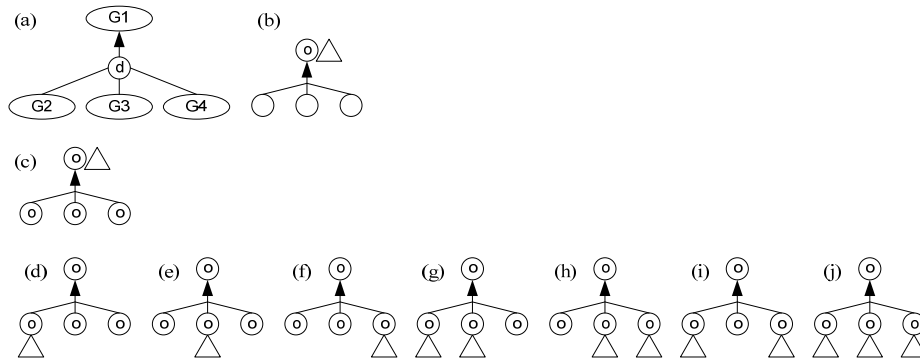


Figure V-15. New markings from a decomposition relationship: (a) – goal model of decomposition relationship; (b) - full marking of the graph corresponding to the goal model; (c) – copy of the marking in (b); and (d), (e), (f), (g), (h), (i), and (j), – new markings

Dependency Relationship. Figure V-16 shows a part of the derived marking in case of a dependency relationship. Figure V-16 (a) and (b) are respectively the goal model and the particular marking we want to investigate. The copies produced by the activity **Generate alternative goal models** are presented in Figure V-16 (c), (d), (e), (f), (g), (h), and (i). These are the input for the activity **Generate new markings**. Copies (c), (d) and (e) are the same as in a substitution relationship. Correspondingly, the new markings are the same (cf. Figure V-14 (f), (g), and (h) with Figure V-16 (j), (k), and (l).) Copies (f), (g), and (h) are unique for the dependency relationship. They have two newly visited nodes each. This leads to a generation of three new markings from every copy. Figure V-16 presents the new marking under the copy that originated them: i.e., the markings from copy (f) are markings (m), (n), and (o); the markings from copy (g) are markings (p), (q), and (r); and the markings from copy (h) are markings (s), (t) and (u). The new markings generated from the copy (i) are the same as the seven marking in the decomposition relationship discussed above. For reasons of space, they are not shown in Figure V-16. (For details, see Figure V-15 (d), (e), (f), (g), (h), (i), and (j).)

¹ There are seven possible combination based on three elements

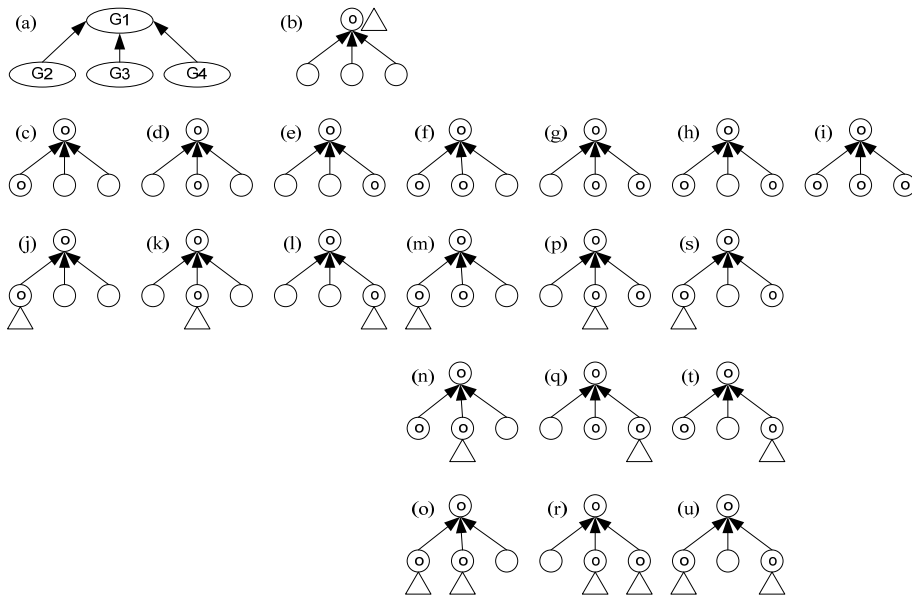


Figure V-16. New markings from a dependency relationship (incomplete: the markings originating from copy (i) are show in Figure V-15) : (a) – goal model of decomposition relationship; (b) - full marking of the graph corresponding to the goal model; (c), (d), (e), (f), (g), (h), and (i) – copies of the marking in (b); and (j), (k), (l), (m), (o), (p), (q), (r), (s), (t), and (u) – new markings

Summary. Above, we describe the fourth activity of the generation algorithm (see Figure V-9.) This shows how new markings are generated and how these depend on the type of goal relationship.

Step-By-Step Example of the Algorithm. The examples above are goal models with only one relationship, which shows only one iteration through the algorithm in Figure V-9. Next, we show a goal model with a graph that is a tree. Figure V-17 presents the example, the markings and the generated alternative goal models. The example is shown in Figure V-17 (a). It is the same example as the one in Figure V-7 (a). The goal model includes the superfluous goal Sup which is considered part of the model. Correspondingly, Figure V-17 (b) represents the same concise goal model as Figure V-7 (b). From the concise model, we construct the first marking: the root goal only with markings visited and next. The first marking is the only one initially present in the pool of markings. Figure V-17 (c) shows the marking.

The second activity of the generation algorithm (see Figure V-9) picks a marking from the pool, namely the only one (see Figure V-17 (c).) The third activity receives this as an input and visits the neighbours of the only node with label next. The resulting copy of the marking is shown in Figure V-17 (d). It is the source of the single generated alternative goal model, shown in Figure V-17 (e). The resulting alternative goal model is only one because the relationship between the superfluous goal and the two top-goals is one of type decomposition. The fourth activity of the algorithm generates new markings from the only copy (Figure V-17 (d)) produced during the generation of alternative goal

models. The newly generated markings are three because the copy has two newly visited nodes. Figure V-17 (f), (g) and (h) show the new markings. The last activity in the algorithm returns the new markings into the pool.

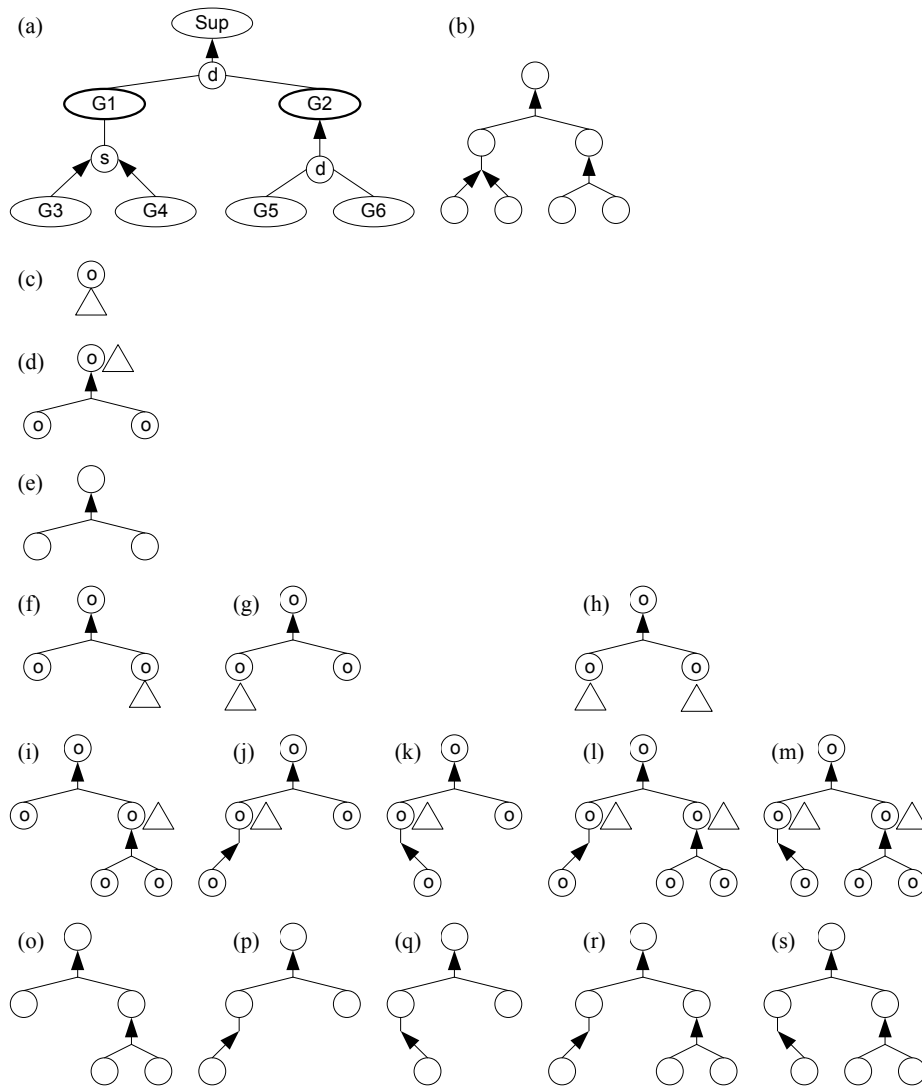


Figure V-17. Step-by-step example of the generation algorithm: (a) – example goal model, the same as in Figure V-7 (a); (b) – concise representation of the goal model from (a), the same as in Figure V-7 (b); (c) – the first marking, the root goal with markings visited and next; (d) – copy of the first marking in (c); (e) – alternative goal model from (d); (f), (g), and (h) – new markings from (d); (i) – copy of the marking in (f); (j) and (k) – copies of the marking in (g); (l) and (m) – copies of the marking in (h); (o) – alternative goal model from (i); (p) – alternative goal model from (j); (q) – alternative goal model from (k); (r) – alternative goal model from (l); and (s) – alternative goal model from (m)

The execution continues with activity number two because the pool contains three markings now. Indifferently of the order, one of these is selected, e.g. Figure V-17 (f). The third activity produces only one copy of the marking (Figure V-17 (i)) and only one alternative goal model (Figure V-17 (h)) because the goal relationship between **G2**, **G5** and **G6** is one of type decomposition. The fourth activity generates no new markings because both goals **G5** and **G6** are not head-goals in any relationship. In the next iteration, the marking in Figure V-17 (g) is selected. It produces two alternative goal models (shown in Figure V-17 (p) and (q)) because the relationship between goals **G1**, **G3** and **G4** is one of substitution. The copies of marking (g), namely (j) and (k), also generate no new markings because both goals **G3** and **G4** are not head-goals in any relationship. The last remaining marking in the pool, the one in Figure V-17 (h), has two nodes with label next. This leads to two alternative goal models (Figure V-17 (r) and (s)) which are combinations of the alternative goal model from Figure V-17 (o), (p) and (q). Correspondingly, copies (l) and (m) generate no new marking because there are no further nodes to visit.

The result of the Generation step is the six alternative goal models shown in Figure V-17 (e), (o), (p), (q), (r), and (s).

4.2 Resolving Graph Structures and Cycles.

In the explanation of the algorithm above, we made implicitly the assumption that all goal models have tree-like graph structure. Here, we discuss the implications for the algorithm in case of the general graph structure, namely a directed cyclic graph.

From the construction of the algorithm, we infer that it is applicable without changes for directed cyclic graphs. The key to understand the indifference of the algorithm with respect to trees and directed cyclic graphs is the definition of a neighbour. A neighbour or a neighbouring node is an unvisited node that is linked with an outgoing arc to a node labelled with visited and next labels (Section 4.1 Algorithm for Generation of Alternative Goal Models.) This definition guarantees (1) that the visiting of nodes spreads opposing the direction of the arcs and (2) that no node will be visited twice. This means that no cyclic paths will be followed during generation but the cycles will remain in the alternative goal models.

The way the markings are constructed guarantees also that beginning from the root node all trees in the graph will be generated as an alternative goal model. An alternative goal model is constructed by removing nodes that are not visited; goal relationships are not removed. This means that every possible value propagation trajectory in the graph is present in at least one alternative goal model.

4.3 The Running Example.

This section presents the results of the execution of the first step of the selection procedure over the example in Section 2. Figure V-18 (a) shows the same example goal model as Figure V-1. The goal model does not contain a superfluous goal because it has only one top-level goal. For uniformity of the selection procedure, and more specifically the generation step, we always add a superfluous goal and link it with the top-level goal or goals. Thus, the concise goal model in Figure V-18 (b) contains a superfluous goal.

After the execution of the Generation step, nine alternative goal models are generated. Figure V-18 (c), (d), (e), (f), (g), (h), (i), (j), and (k) represent them. The alternative goal models are sent to the Matching step, where every leaf-goal in an alternative goal model is matched with capabilities.

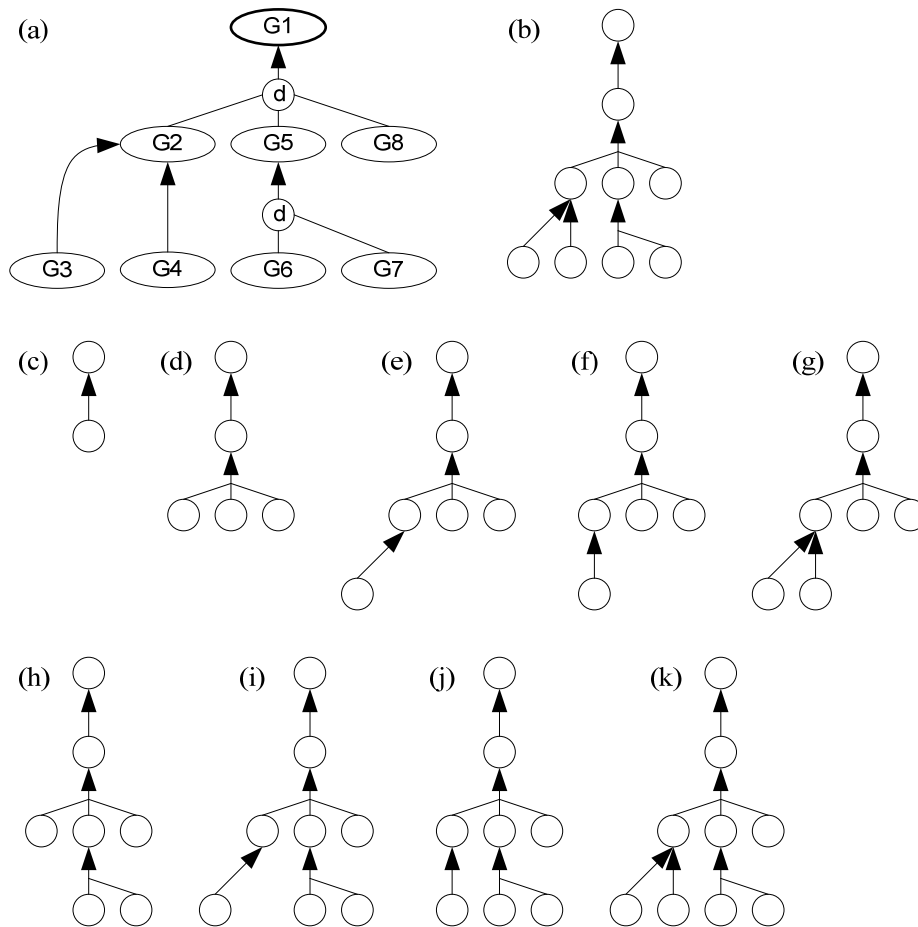


Figure V-18. Alternative goal models of the running example: (a) – goal model of the example; (b) – concise representation of the goal model, including a superfluous goal; and (c), (d), (e), (f), (g), (h), (i), (j), and (k) – alternative goal models

4.4 Summary

The generation step receives a graph representing the goal model as an input and produces a number of alternative goal models. The alternative goal models represent all valid combinations of goals and relationships such that the consequent steps can work only with the leaf-goals.

5 Matching Goal and Capability Models

Matching capability and goal models is in essence establishing a relation between offered abilities and desired functionality. This is the step, where the requirements of the system under development, represented by goals in the goal model, are bridged with existing design fragments, annotated with capability models.

To fill the gap of design knowledge, it would be enough to match every leaf-goal in the goal model with a single capability representing a unique design fragment. This is the ideal scenario. In reality however, the design fragments are not unique; the capability models contain many goals; moreover, the capability models contain many top-level capabilities; and finally capabilities and capability models do not match only with leaf-goals. Thus, relating goal and capability models is not a straightforward task.

This section has four main parts. The first one describes the assumptions we make to simplify the real case of matching goal models to the ideal case of matching only goals. The second part defines when a goal and a capability match. Then, the third one describes in detail the Matching step as used in the selection procedure in Figure V-5 and Figure V-6. Finally, part four applies the Matching step on the running example.

5.1 Simplifying the Match between Goal and Capability Models

The initial situation is that a goal model and a number of capability models have to be matched. The objective is to simplify the matching to a matching of a single leaf-goal with a single capability.

Temporarily, we constrain our discussion to only one goal model and only one capability model.

Reducing Capability Models to a Single Capability. The first step towards matching goals and capabilities instead of models is to reduce the capability models to a capability.

- *Simplifying assumption 1:* The top-level capabilities capture all the information about a design fragment.
- *Consequence:* A capability model is matched only at the level of its top-level capabilities.

The assumption is justified because the top-level capabilities capture the essential information about the functional and quality properties of a design fragment. Thus, the detailed information in the remaining capabilities in a capability model can be neglected as it is already propagated to the top-level capabilities. Later in the Integration step of the selection procedure, every capability can be linked to the goal model and impact the value of some goals.

With simplifying assumption 1, we reduce a capability model to top-level capabilities only. Nevertheless, a capability model has in general more than one top-level capability. These cannot be unified to a single capability because important information will be lost. Thus, we need to show that top-level capabilities can be matched separately with goal models.

- *Simplifying assumption 2:* The top-level capabilities in one capability model are always independent of each other.

- *Consequence*: A design fragment annotated with a capability model having several top-level capabilities is regarded as several design fragments with capability models having a single top-level capability.

The assumption is justified because the values of the variables in the capability model are already propagated. That is, even if two top-level capabilities are related, their relationship will not further influence the values of the top-level capabilities. And a match of a design fragment with one of the top-level capabilities is a sufficient condition for the selection of the fragment in the design alternative. The fragment is not duplicated in a design alternative if a second top-level capability is matched.

Restricting Capabilities to Match Only With Goals. Reducing a goal model to a single goal makes sense only for the capability models. Losing information from the requirements goal model of the system under development by abstracting goals and relationships is not desirable. Therefore, we need another approach to matching a single goal with a single capability. We need to restrain the possibility of a capability matching part of a goal model.

A capability matches a number of goals if it is an aggregation or abstraction of these goals. (Henceforth, we use for ease of explanation only abstraction but we mean everywhere aggregation or abstraction.) The consequence is that the top-level capabilities match the abstraction of several goals.

To be able to match capabilities with goals, we need to ensure that the abstraction of any set of goals from the goal model that can possibly match a capability is also a goal in the goal model. From the discussion above follows that the top-level capabilities are equally or more abstract than the leaf-goals in a goal model. To ensure that the top-level goals in the goal model are the most abstract goals, we assume:

- *Simplifying assumption 3*: The levels of abstraction or aggregation in the goal model always include the level of abstraction or aggregation of any top-level capability.
- *Consequence*: A capability cannot match several top-level goals.

The assumption is justified because usually the goal model represents the composite design and the top-level goals are usually the most abstract ones. The capabilities, on the other side, are the basis for a composite design. Even if a design fragment is relatively big, it never equals the final design.

With simplifying assumption 3, we guarantee that the top-level capabilities are less abstract than the top-level goals and more or equally abstract than the leaf-goals from the goal model. However, we still need to assume that:

- *Simplifying assumption 4*: A capability that matches a number of goals from a goal model always matches another more abstract or aggregated goal in the goal model.
- *Consequence*: A capability always matches one goal in the goal model.

We justify the assumption with the argument that if the abstraction of certain goals is important for the system under development then it is represented in the goal model as separate goal. If a capability matches a number of goals but does not match any single goal from the goal model then we consider that the capability achieves something that is not intended in the goal model.

In a nutshell, if a capability that matches a number of goals from a goal model does not match another more abstract or aggregated goal in the goal model then the design fragment is not relevant for this goal model.

Matching Leaf-Goals Only. The result of the simplifying assumption 1, 2, 3 and 4 is that a top-level capability matches any goal from the goal model. The desired situation is that a single capability can match leaf-goals only.

The possibility to match a capability with any goal from the goal model is due to the difference of abstraction or aggregation. One way to achieve matching of leaf-goals only is to remove information from the goal model by cutting detailed leaf-goals. By doing this, abstract goals become leaf-goals. An alternative approach is to ‘grow’ the goal model from the top-level goals. We introduce the Generation step (see Section 4) as a preceding of the Matching step. The Generation step produces alternative goal models, one per each sub-graph of the goal model containing the top-level goals. The consequence is that we can cover the goal model by matching only leaf-goals in each given alternative goal model.

Cardinality of Match. In the beginning of the section, we constrain the matching to only one goal model and only one capability model. Here, we relax the restriction and discuss the cardinality properties of the matching relationship between goals and capabilities.

The design space of available design fragments is not restricted: new fragments may be added and offered capabilities may be duplicated. This leads to the questions of (a) how many goals one capability can match and (b) how many capabilities one goal can match. These two questions are addresses as follows:

Many Goals Match One Capability. In this case, several goals from the goal model can be achieved by one design fragment. We distinguish two types of design fragments: scalable and non-scalable. The former design fragments are capable of achieving the values in the top-level capabilities regardless of the number of matched goals. Contrarily, the non-scalable design fragments are fully utilized by one goal and every subsequent goal that matches with the same capability needs another copy of the design fragment in the implementation.

In the process of matching, we do not need to know if the design fragment is scalable or not. We always match one goal with one instance of the design fragment. The consequence is that in case of a scalable design fragment we have redundancy in the final specification. This drawback is compensated in the Integration step, where unnecessary capability models (design fragments) are removed.

One Goal Matches Many Capabilities. In this case, one goal is achieved by several design fragments. All fragments are valid design choices; thus, they all have to be considered as a part of a potential design alternative. We accommodate this case by adapting the selection procedure. We consider all possible matches of a goal with capabilities in separate design alternatives. The consequence for the Activity diagrams

from Figure V-5 and Figure V-6 is that the Matching activity produces not one but several lists of four-tuples¹.

Summary. The simplifying assumptions and workarounds in this section lead from matching goal and capability models to one-to-one matching of leaf-goals and top-level capabilities.

5.2 Matching Goals

A capability and a goal match if they refer to the same phenomenon². This relation is difficult to establish because goals and capabilities are typically specified by different people, with different purposes in mind, using different perspectives, different contexts, incompatible vocabularies, and incompatible knowledge.

We assume (1) that goal and capability models are developed by two different groups of people and (2) that goals and capabilities are matched by a third group of people. Therefore, the matching is always performed on the representation of the goals and capabilities; i.e., we have available only the operationalization of goals and capabilities.

Each goal and capability is operationalized through a variable of certain domain and an evaluation function. Matching goals translates to matching the variables that represent the goals. The evaluation function is disregarded as in the context of the capability model it may have different interpretations. That is, the value of a variable in the capability model may not be evaluated as satisfying by the capability evaluation function but still is sufficient for the evaluation function in the goal models. That is, a value that does not satisfy a capability may be an option worth considering in the goal model.

Capabilities are goals that have assigned values to their goal variables. For this reason and because the matching process does not consider the value of the variable and the evaluation function over that value, we refer to the matching of a goal and a capability as to matching of two goals.

- *Criterion for a match between goals:* two goals (a goal and a capability) match if the variables that represent them match.

The variable representing a goal or a capability is a measurement of an indicator assigned to a phenomenon. Therefore:

- *Criterion for a match between variables:* two variables match if they measure the same indicator of the same phenomenon in the same context and they take values from one domain.

This is a strong criterion, having in mind that the goal and capability models are developed independently and by different people. The following two problems may arise because of that:

¹ Due to the type of match when one goal matches many capabilities, the output of the Matching step is a list of lists of four-tuples. For simplicity of presentation, the selection procedures in Figure V-5 and Figure V-6 present the Matching step as producing a single list of four-tuples. Further in the chapter, we refer to the Matching step as resulting in one list of four-tuples. This simplification does not affect our argumentation.

² Our definition of a goal is a desired phenomenon by a certain organization or individual, where a phenomenon is a state of reality, an activity, a fact or an event (Chapter IV Goal-modelling, Section 1 Introduction)

Undetected Matches. The first potential problem is that two goals may match but remain undetected because they were operationalized using different terms. To discover such matches, a person has to interpret every goal specification and reason about a potential match with every other goal. This is a time and effort consuming task. As we show in the discussion of the selection procedure, where Matching is the second step (see Section 3), the number of checks for a match between the goal model and all capability models is potentially high. To make the process feasible and to enable the automation of matches, we require that the conceptual models used to describe the universe of discourse of goal and capability models are the same. That is, we refer to phenomena, indicators, variables, and variable domains with the same names in all models.

- *Simplifying assumption 5 (Assumption about common conceptual models):* the goal and capability models use the same concepts to refer to identical phenomena, indicators, variables, and variable domains.

A common conceptual model means that one variable is represented by the same term in all models. This reduces the matching process to a string matching. With common terminology, we can do the matching of goals automatically.

The assumption of common conceptual models may be considered too restrictive and, therefore, over-simplifying the problem of matching. We select it because: (1) matching is not the focus of this work and (2) there are active research communities working in the area of database schema matching and semantic interoperability [16, 47]. Results arising from those fields can be plugged into our matching procedure without affecting the rest of the procedure.

Other automated matching approaches exist which translate concepts and relations between conceptual models [33]. They require human intervention in the phase of relating the concepts from each conceptual model; while in the actual matching, they are automatic. Nevertheless, the two matched variables may still require a transformation function for the transfer of value. Specifying transformation functions is a manual process that may require information that is not available. Therefore, we use only exact matches in which the variable values are copied.

Too Few Matches. The second potential problem is that the criterion is too restrictive and, potentially, there will be too few matches. A possible weaker criterion would not need the domains of the variables to be the same. This would additionally require a conversion step from domain to domain. It is a viable solution for variables with simple domains, where automatic conversion is possible; nevertheless, undesirable human intervention would be necessary in case of complex domains.

Another weakening of the criterion would allow match of variables represented by similar¹ concepts. This approach would also require manual conversion of variable values, which makes it undesirable.

Counter Criterion. Accepting similar concepts as matching variables may lead to misleading matches which are difficult to identify. In a manual matching process, it seems intuitive to match two goals that have causally related variables. Especially in the

¹ By similar concepts, we mean concepts that in some interpretation may be considered the same but in their current representation this cannot be made evident as their conceptual models are based on incompatible knowledge.

case of an individual determinative relationship, it may be difficult to separate the cause variable from the affected variable. Such matches do not deteriorate the goal model; nevertheless, conceptually this is an extension of the goal model and the transfer of value requires a propagation function. Mismatches such as these occur when the level of abstraction of the goal is one level higher than the one of the capability. Thus:

- *Criterion for mismatch of goals:* two goals do not match if the variables that represent them are causally related.

The criterion for mismatch is not complementary to the criterion for match. The criterion for match includes the criterion for mismatch. The explicit specification of mismatching criterion additionally clarifies when two goals do not match.

5.3 The Matching Step

The Matching step identifies all possible matches between goals and capabilities. It forms design alternatives which are sets of relevant design fragments realizing some of the goals in a particular alternative goal model. The design alternatives are predetermined by the Generation step which produces the alternative goal models.

The alternative goal models are the input for the Matching step. The output is a list of four-tuples, where each tuple contains a goal, a capability, a value and a design fragment. The information in a tuple is: which goal is matched to which capability; what value the goal receives; and which is the design fragment that delivers the value. A design alternative is the set of all design fragments present in a list of four-tuples.

The Matching step has to match every leaf-goal from every alternative goal model with a top-level capability. From optimization point of view, it is worth first matching all goals from the goal model with all top-level capabilities because (1) all goals are leaf-goals in at least one alternative goal model and (2) many alternative goal models contain the same leaf-goals. Therefore, the Matching step has two sub-steps: building a Matching table and forming a design alternative.

Building a Matching Table. In the first sub-step, every goal from the goal model is matched with every top-level capability of every design fragment. The result is a table with rows containing the goals and columns containing the capabilities of particular design fragments. (See Table V-7 in the running example below for an example.) Every cell in the table is marked with 'x' or not marked. The meaning is, respectively, the goal and the capability match or the goal and the capability do not match. We call the table the Matching table. Goals and capabilities are matched using the criterion from Section 5.2. Assuming a common conceptual model, matching goals are discovered by an automatic string matching of variable names.

Forming a Design Alternative. In contrast with the first sub-step, the second sub-step is executed per alternative goal model. All leaf-goals from one model are checked for matches in the Matching table. The matching capabilities are used to form a design alternative and the output of the Matching step, a list of four-tuples.

A design alternative is the set of design fragments corresponding to capabilities that match leaf-goal from the alternative goal model. If more than one capability match a particular goal then for every match a separate design alternative is formed.

For each design alternative, a list of four-tuples is created. The list is the representation of the design alternative with which the consecutive steps of the selection procedure work.

Handling Leaf-Goals without a Match. To be able to evaluate a design alternative, all leaf-goals in the alternative goal model need to have values. This requires that all leaf-goals are matched with capabilities.

The design space of available design fragments is not necessarily complete with respect to the desired system at hand. As a result, not all of the leaf-goals have a matching capability, hence a design fragment. If the system under development implements an innovative business idea, the designer may find that some of the goals are unique to this system and do not have predefined design fragments. To enable the evaluation of partial designs, we have to assume values for the leaf-goals without a match. We assume that the required functionality can be achieved independently of the available design fragments, which results in values for the goal variables called *anticipated* values. The anticipated values are part of the goal model and they can be assigned to any goal. Nevertheless, they are most likely to be assigned to leaf-goals.

Representing Design Alternatives with a List of Four-Tuples. A list of four-tuples includes a tuple for every leaf-goal; i.e., the list contains as many elements as leaf-goals and the first field of every tuple is filled with a concrete leaf-goal. For every leaf-goal that has a match the remaining three fields, namely capability, value and design fragment, are filled in correspondingly. If a leaf-goal does not match any capability but has an anticipated value then only the value field of the tuple is filled in. In case a leaf-goal does not have a matching capability and does not have an anticipated value then the tuple remains unchanged, i.e. empty places for capability, value and design fragment. It is possible to further evaluate a design alternative through the remaining steps of the selection procedure only if there is a value specified in every tuple from the list.

The next section provides an example.

5.4 The Running Example

This section shows the Matching step on behalf of the running example from Section 2. It presents the Matching table containing all eight goals and all three top-level capabilities. It also justifies the data in the table. Moreover, it shows the lists of four-tuples for every alternative goal model.

Table V-5 presents the goals from the example and their operationalization. It is a copy of Table V-1. The bold font in the table highlights the variables that have matching capabilities.

Table V-5. Operationalization of the goals from the running example (a copy of Table V-1)

<i>Goal code</i>	<i>Goal</i>	
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>
G1	To offer online technical support	
v ₁	Quality of service	Real numbers between 0

		and 1
G2	To support troubleshooting	
v ₂	Percentage unanswered question	Percentage
G3	To maintain FAQ	
v ₃	Number of question and answers	Natural numbers
G4	To maintain an active forum for clients	
v ₄	Percentage active one-week-old topics	Percentage
G5	To publish online documentation	
v ₅	Percentage of inaccessible pages	Percentage
G6	To support keyword search	
v ₆	Percentage not indexed pages	Percentage
G7	To perform weekly updates	
v ₇	Number updates	Natural numbers per year
G8	To have 24/7 online instant messaging	
v ₈	Online hours	Natural numbers per week

Table V-6 presents the top-level capabilities, their operationalization, and the values of the goal variables. The table is a compilation of Table V-2, Table V-3, and Table V-4 from which only the rows with the top-level capabilities are taken. Again, the bold font in the table highlights the variables that have matching goals.

Table V-6 Operationalization of some of the capabilities from the running example (compilation of Table V-2, Table V-3, and Table V-4)

<i>Capability code</i>	<i>Capability</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
CM1.C1	Sustain an active forum		
CM1.a ₁	Percentage active one-week-old topics	Percentage	73
CM2.C1	Support up-to-date online help		
CM2.a ₁	Percentage of inaccessible pages	Percentage	7
CM3.C1	VoIP call center service		
CM3.a ₁	Online hours	Natural numbers per week	40

The first sub-step of the Matching step, namely building a Matching table, requires checking for a match every goal from the goal model (goals **G1** to **G8** from Table V-5) and every top-level capability (capabilities **CM1.C1**, **CM2.C1**, and **CM3.C1** from Table V-6).

Without the assumption about common conceptual models (simplifying assumption 5), the reasoning behind the selection of matching goals and capabilities is:

- **G1: To offer online technical support.** This goal is too abstract for the available capabilities; therefore, it has no matches.
- **G2: To support troubleshooting.** This goal has no analogue among the capabilities; therefore, it has no matches.

- **G3: To maintain FAQ.** This goal has no analogue among the capabilities; therefore, it has no matches.
- **G4: To maintain an active forum for clients.** This goal has a direct match. It is operationalized with the same variable as the capability **CM1.C1**.
- **G5: To publish online documentation.** This goal is similar with capability **CM2.C1**. More importantly, they are operationalized with the same variables. Thus, goal **G5** matches capability **CM2.C1**.
- **G6: To support keyword search.** This goal is too detailed for the available capabilities. Its operationalization is close to the operationalization of capability **CM2.C1** but an extra assumption that ‘inaccessible’ is the same as ‘not indexed’ is required. Thus, goal **G6** has no match.
- **G7: To perform weekly updates.** This goal is similar to capability **CM2.C1**. Nevertheless, they have different operationalizations. A possible match would require translation between the variables representing the goals. Therefore, the goal has no match.
- **G8: To have 24/7 online instant messaging.** This goal has the same operationalization as capability **CM3.C1**. Therefore, they match; despite, the meaning of the goals is slightly different.

With the assumption about common conceptual models (simplifying assumption 5), which is also the automatic approach that our Matching activity performs, the matching goals and capabilities are decided based on string matching. This approach leads to the same result. The names of the matching variable names are typed in bold font in Table V-5 and Table V-6.

Applying the matching criterion from Section 5.2, we find the following matching goals and capabilities (see Table V-7):

- Goal **G1** matches capability **C1** of capability model **CM1**;
- Goal **G5** matches capability **C1** of capability model **CM2**; and
- Goal **G8** matches capability **C1** of capability model **CM3**.

Table V-7 presents the Matching table. The rows are goals, the columns are capabilities from particular capability models. The ‘x’ in the sells marks the matching goals and capabilities.

Table V-7. Matching table of the running example

	CM1.C1	CM2.C1	CM3.C1
G1			
G2			
G3			

G4	x		
G5		x	
G6			
G7			
G8			x

The second sub-step of the Matching step, namely forming design alternatives, checks for matching capabilities and leaf-goals in every alternative goal model. Table V-8 presents an aggregated view on the alternative goal models, the leaf-goals of every model and the matching capabilities. The columns represent alternative goal models; the letters c to k in the second row correspond to the letter denoting the alternative goal models in Figure V-18. The rows represent goals from the goal model. A grey cell means that the particular goal is not a leaf-goal in the particular alternative goal model. Correspondingly, a white cell means that a goal is a leaf-goal in an alternative goal model. For example, alternative goal model d has three leaf-goals, namely **G2**, **G5**, and **G8**. If a leaf-goal from a particular alternative goal model and a capability match then the corresponding white cell contains the name of the capability.

Table V-8. Lists of goal-capability pairs

	Alternative goal models								
	c	d	e	f	g	h	i	j	k
G1									
G2									
G3									
G4				CM1.C1	CM1.C1			CM1.C1	CM1.C1
G5		CM2.C1	CM2.C1	CM2.C1	CM2.C1				
G6									
G7									
G8		CM3.C1	CM3.C1	CM3.C1	CM3.C1	CM3.C1	CM3.C1	CM3.C1	CM3.C1

The output of the Matching step is a list of four-tuples. Below, we present the output for every alternative goal model:

- c: <G1, , , >;
- d: <G2, , , >, <G5, C1, 7, CM2>, <G8, C1, 40, CM3>;
- e: <G3, , , >, <G5, C1, 7, CM2>, <G8, C1, 40, CM3>;
- f: <G4, C1, 73, CM1>, <G5, C1, 7, CM2>, <G8, C1, 40, CM3>;
- g: <G3, , , >, <G4, C1, 73, CM1>, <G5, C1, 7, CM2>, <G8, C1, 40, CM3>;
- h: <G2, , , >, <G6, , , >, <G7, , , >, <G8, C1, 40, CM3>;
- i: <G3, , , >, <G6, , , >, <G7, , , >, <G8, C1, 40, CM3>;
- j: <G4, C1, 73, CM1>, <G6, , , >, <G7, , , >, <G8, C1, 40, CM3>;
- k: <G3, , , >, <G4, C1, 73, CM1>, <G6, , , >, <G7, , , >, <G8, C1, 40, CM3>;

The only alternative design resulting from the Matching step that can be further evaluated is the one originating from alternative goal model f. More alternative designs would have been possible if the goal model contained anticipated values. The example we use in this chapter to illustrate the selection procedure does not contain anticipated values.

6 Integrating Goal and Capability Models

The purpose of the Integration step is to compensate for the simplifying assumption¹ 1 and for the simplifying assumption² 4, and the many-to-one matches between goals and capabilities (see Section 5.1.)

The drawback of the assumptions is that according to them, one capability can match only one goal and can set the value of only this one goal. In reality, one capability influences the value of potentially many goals from the goal model. Furthermore, not only top-level capabilities influence goal values in goal models; all capabilities from a capability model can affect the goals in a goal model.

The drawback of the work-around solution to the many-to-one relation between goals and a capability presented in Section 5.1 is that it duplicates non-scalable design fragments. In case of a scalable design fragment (see Section 5.1), several capability models may be reduced to one and, in this way, remove duplicated design fragments.

To offset the drawbacks, we need to add information to the goal model. This is realized by adding goal relationships between capabilities and goals. Adding a relationship requires adding a new or overwriting an existing propagation function. In essence, the integration is goal modelling.

6.1 The Integration Step

The Integration step takes as an input a list of four-tuples. Implicitly, it also receives the alternative goal model that originated this list. The Integration step merges the alternative goal model with the capability models of the design fragments present in the list of four-tuples. The output is an integrated alternative goal model.

Merging the goal models includes two types of actions: replacing goals with capabilities and adding goal relationships.

Replacing Goals with Capabilities. Every capability model has at least one top-level capability that matches with a goal from the goal model. This top-level capability replaces the goal it matches, while keeping goal relationships. The relationships of the capability in the capability model are also kept. By replacing a goal with a capability, (1) the value of the capability is made part of the goal model and (2) the capability model is linked with the goal model.

¹*Simplifying assumption 1:* The top-level capabilities capture all the information about a design fragment (see Section 5.1)

²*Simplifying assumption 4:* A capability that matches a number of goals from a goal model always matches another more abstract or aggregated goal in the goal model (see Section 5.1)

Adding Goal Relationships. The newly added capabilities may affect the values of goals in the goal model not matched by the top-level capability of the capability model. This happens in case the variables of the goal and capability are causally related. The causal relationship is accounted for by adding a new goal relationship from the capability to the goal. Depending on the type of relationships the goal has, the new relationship adds a new propagation function (in case the goal is only tail in its relationships) or replaces the existing propagation function (in case the goal is a head in a relationship.)

6.2 Handling Feature Interaction

By feature interaction, we mean the occurrence of properties of a composition of two designs that were not foreseen. Every design fragment has certain properties which may not hold in a combination with other design fragments in a design alternative. Moreover, new properties may emerge because of the composition of several design fragments. This problem is one of the primary concern of Chapter VI Synthesis of Value and Process Patterns; nevertheless, the consequences of the feature interaction can be accounted for in the Integration step.

Two design fragments that are interacting in a certain way are expected also to interact in the same way at capability models level. For example, if two design fragments combined have a synergy, it is intuitive to expect that the achieved variable values are closer to satisfaction. In the contrary, two design fragments can be conflicting, which, intuitively, results in variable values that are further from satisfaction.

The effects of the feature interaction cannot be added directly as a relationship between capabilities because the capabilities do not change their values. To account for a potential interaction between capabilities from different capability models, we have to add goal relationships to goals from the goal model.

6.3 The Running Example

The Matching step produced the following list of four-tuples that can be further evaluated:

- f: <G4, C1, 73, CM1>, <G5, C1, 7, CM2>, <G8, C1, 40, CM3> (see Section 5.4.)

The alternative goal model in which goals **G4**, **G5** and **G8** are the only leaf-goals is shown in Figure V-19. The alternative goal model is derived by removing goals **G3**, **G6** and **G7** from the goal model of the running example shown in Figure V-1.

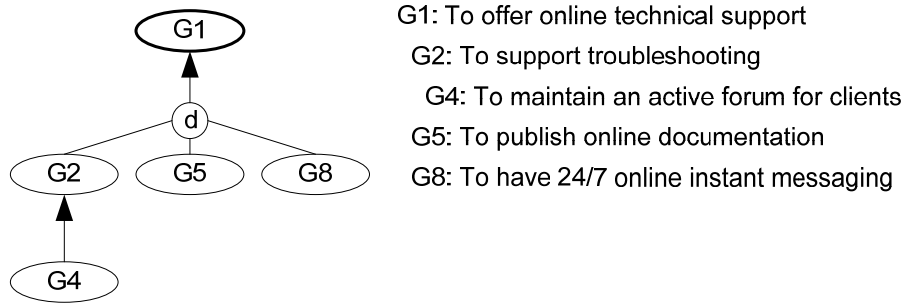


Figure V-19. Alternative goal model number 4 (see Table V-8)

The Integration step replaces goals **G4**, **G5** and **G8** respectively with capabilities **CM1.C1** (see Figure V-2), **CM2.C1** (see Figure V-3), and **CM3.C1** (see Figure V-4.) The capability models **CM1** and **CM2** of, respectively, capabilities **CM1.C1** and **CM2.C1** contain each three capabilities. These capabilities are added to the alternative goal model. The result of the integration step is the integrated alternative goal model shown in Figure V-20.

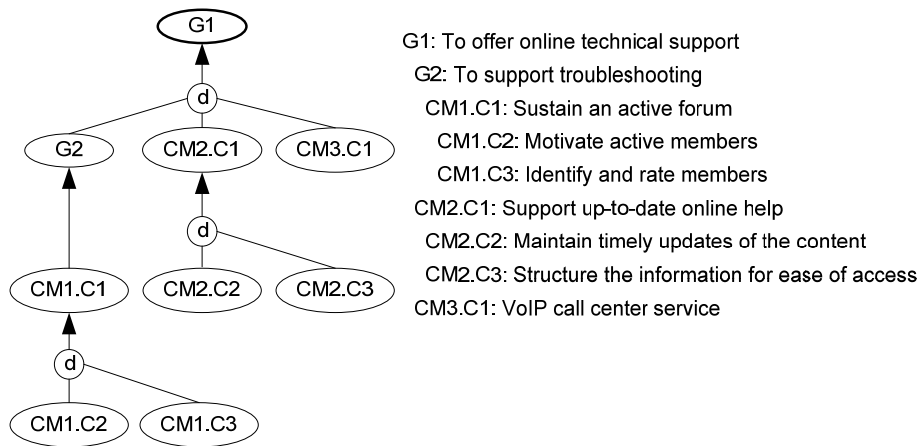


Figure V-20. Integrated alternative goal model derived from alternative goal model 4 and capability models 1, 2, and 3

Figure V-20 contains no additional goal relationships because the Integration step does not add any. The added capabilities do not interact among each other and do not affect more goals than the ones they match. Therefore, there is no need for additional goal relationships.

7 Propagation and Evaluation Steps

This section briefly describes the last two activities, Propagation and Evaluation, from the selection procedure from Figure V-5.

Propagation Step. The step receives as an input an integrated alternative goal model which has values¹ assigned to leaf-goals. By applying the satisfaction calculation described in Chapter IV Goal-modelling, Section 5 Satisfaction Calculation, we determine the values of the top-level goals in the integrated alternative goal model. The output of the step is an alternative solution capability model which is a goal model with calculated values for the top-level goals.

Evaluation Step. For a description of the Evaluation step, we refer to Section 3. The step receives as an input a list of alternative solution capability models. Each alternative solution capability model is seen as a solution with multiple properties, where the properties (also called attributes) are the top-level goals in the model. The Evaluation step assigns a number (a rating) to each alternative solution capability model by solving a multiattribute decision problem based on the stakeholders' criteria. The output is a list of alternative solution capability models, where every model has a rating.

8 Improved Selection Procedure: the Pre-evaluation Step

The main drawback of the selection procedure from Figure V-5 is that the Integration step is executed for every design alternative. This is expensive, in terms of time and effort, because the integration is a manual goal modelling activity which has to be executed potentially many times. To counter this disadvantage, we postpone the execution of the Integration step to a point in the selection procedure when a pre-selection of the most promising alternative designs is made. To justify the change in the order of activities, we need to assume that:

- *Simplifying assumption 6:* The added information during the Integration step does not change significantly the value of the top-level goals.

The relationships added during the Integration step represent causal relations which were not considered important in the first place and therefore not added in the goal model, or they represent a weaker form of a match between a capability and a goal. This supports our simplifying assumption.

The need of the Integration step is to account for incompatible design fragments (feature interaction.) In this case, the values of the top-level goals are changed significantly but usually negatively; i.e., because of the integration, the design alternative scores lower.

Postponing the Integration step, we do not miss a good solution. In the worst-case scenario, we can initially nominate a bad solution as good one but this will be noticed.

¹ The value is a capability value if the leaf-goal is a capability or an anticipated value if the leaf-goal is an unmatched goal.

In the improved selection procedure (see Figure V-6), the Integration step is preceded by a Pre-Evaluation step. This step is the same as the Evaluation step and its function is to reduce the number of design alternatives further considered by the selection procedure. A limited number of alternative solution capability models are nominated as best alternative solution capability models and passed further to the Integration step. After the integration and before the final evaluation, a second execution of the Propagation step is needed to account for the newly added relationships.

All steps except the Integration are automatic. The improvement of the selection procedure from Figure V-6 in comparison with Figure V-5 is in the reduced number of executions of the Integration step.

9 Related work

The work of Gross, D. and Yu, E. [28] triggered the elaboration of the selection procedure to its current specification. The common idea is to use requirements goal models to evaluate design patterns by scoring the contribution of each pattern to the satisfaction of particular goals in a goal model. In addition, our method prescribes how to select patterns, how to combine the contribution of multiple patterns, how to propagate quantitative values throughout the model, and how to rate alternative designs.

The remaining discussion applies to the matching step only, not the selection procedure as a whole.

In our design method, patterns are selected as relevant for a particular system under development if a capability goal model matches a requirements goal model. In Section 5, we make a number of assumptions to simplify the problem of matching. The Database Schema Matching and Integration research [16, 47] addresses the problem in its full complexity. The difference of the matched structures is in the semantics of what they represent. Therefore, any approach from the area Database Schema Matching and Integration that does not assume the matched structure to be a conceptual model (database schema) is reusable in our framework. As we state in the future work discussion of this thesis, additional research is needed to evaluate the approach that best suits the purposes of selecting relevant patterns. All of the proposed solutions involve manual intervention to a various degree.

Results from the Ontology mapping research [33, 52] can be also used to match goals. Ontology mapping is finding concepts and relations in two ontologies that refer to the same real-life entities¹. This is analogous to our problem of finding goals and capabilities that refer to the same phenomenon. A goal model, correspondingly a capability model, is not an ontology. Nevertheless, the expressiveness of some ontology covers goal models. The type of ontology called *populated* ontology contains instances of concepts and these instances may be goals.

Ontology mapping varies in precision and coverage [12]. Precision is the degree of preserving the intended meaning of the mapped concepts and relations when translating

¹ This is a simplified definition. In the general case, ontology mapping includes more than mapping concepts and relations. For more information, refer to [33].

from one ontology to the other. Coverage is the part of concepts and relations intended to be mapped that are actually mapped.

An ontology mapping approach is applicable to goal matching under the following condition: there exists an ontology mapping with a perfect¹ precision and full² coverage between the populated ontology representing the goal model and any populated ontology representing a capability model. Such an approach leads to the same results as following our matching criterion and assumption number 5 for common conceptual model from Section 5.2 but without making the assumption. The drawback of such an approach is the expense of building the ontology mapping, which is a manual process [33], and the strict requirements for precision and coverage. The advantage is that once the mapping is developed the matching of goals is automatic.

The ontology matching approach can be used without the restriction of perfect precision. But the consequence would be that a goal and a capability that do not refer to the same phenomenon will be matched, which requires an additional value transformation function between the goal and capability.

10 Summary

This chapter completes the answer of research question **Q2.2.1: How to identify the relevant design patterns given a set of requirements?** and answers our research question **Q2.2: What are the relevant design patterns for a particular design?** It defines a selection procedure to be used to select the relevant design patterns using a goal model of the requirements.

In the process of defining the selection procedure several simplifying assumptions were made. Below, we summarize the assumption and their consequences:

- *Simplifying assumption 1:* The top-level capabilities capture all the information about a design fragment.
- *Consequence:* A capability model is matched only at the level of its top-level capabilities.

- *Simplifying assumption 2:* The top-level capabilities in one capability model are always independent of each other.
- *Consequence:* A design fragment annotated with a capability model having several top-level capabilities is regarded as several design fragments with capability models having a single top-level capability.

- *Simplifying assumption 3:* The levels of abstraction or aggregation in the goal model always include the level of abstraction or aggregation of any top-level capability.

¹ By perfect precision, we mean that the two concepts or relations have the same meaning in the two ontologies. In the particular case, we mean that the mapped goal and capability instances refer to the same phenomena.

² By full coverage, we mean that all concepts and relations intended to be mapped are actually mapped. In the particular case, we mean that every instance of a goal or capability is known to map or not to another instance.

- *Consequence*: A capability cannot match several top-level goals.
- *Simplifying assumption 4*: A capability that matches a number of goals from a goal model always matches another more abstract or aggregated goal in the goal model.
- *Consequence*: A capability always matches one goal in the goal model.
- *Simplifying assumption 5*: The goal and capability models use the same concepts to refer to identical phenomena, indicators, variables, and variable domains.
- *Consequence*: Matching of variables can be reduced to string-matching of their names.
- *Simplifying assumption 6*: The added information during the Integration step does not change significantly the value of the top-level goals.
- *Consequence*: A Pre-evaluation step can be performed before the Integration step.

Chapter VI

Synthesis of Value and Process Patterns

The chapter answers research question **Q2.3**. It presents the synthesis procedures for value and process patterns. Both procedures are described in the following structure: first, an example set of patterns is given; second, the synthesis procedure is defined; and third, the procedure is applied on the example set of patterns¹.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	Chapter V Selection Procedure
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

1 Introduction

To synthesize patterns means to build one model from the solution fragments of each one of them. The resulting model is more than the parts (the fragments) that make it up. The gains from the synthesis are the synergies and emerging properties of the system.

Our synthesis approach is an automated procedure used by people to design a system with desired properties. The properties of the synthesized model are not formally

¹ This chapter is based on previous work by Zlatev, Z., Eck van, P., Wieringa, R. and Gordijn, J. [76]

derivable from the properties of the fragments. The synthesis procedure does not guarantee that the synthesized model will satisfy the top-level goals. The synthesis procedure is the best effort to design a system with the available information. It involves creativity and experience to spot synergies and avoid negative interaction of pattern properties leading to dysergy.

The Integration step from the Selection procedure (Chapter V Selection Procedure, Section 6 Integrating Goal and Capability Models, on page 112) is connected with the synthesis of patterns. It 'synthesizes' the goal model of the patterns. That is, it accounts for interacting pattern properties reflected in the capability models.

The proposed value and process synthesis procedures support the designer by (1) prescribing a sequence of steps, (2) providing a number of automatic steps, and (3) validating the syntax of the synthesized model. Below, we define them using two examples.

2 Synthesis of Value Fragments

2.1 Example

Let us assume that we have the following example case. A company wants to form a business network in which it is the intermediary of certain type of transactions. To be concrete, the company wants to create a market for exchange of overcapacity memory integrated circuits. We call the company MemEx. It is a global market with properties of a perfect market: i.e., low trust, anonymous participants, and no follow-up exchanges. To be profitable, MemEx has to create a market with critical mass of participants. To do so, MemEx sets the following objectives: (i) increase the trust in the market and in the participants; (ii) discourage opportunistic behaviour; and (iii) remain neutral in creating incentives for future transactions. The analysis of the objectives (we assume them our top-level goals) leads to lower level goals such as:

- **G1: Require supplier of memory integrated circuits to file credit information prior to making an offer;**
- **G2: Require buyers of memory integrated circuits to register upfront purchase; and**
- **G3: Ask suppliers to evaluate buyer.**

We do not show the complete goal model here because it is outside the scope of this chapter.

Following the selection procedure (Chapter V Selection Procedure) for finding patterns that can be part of a possible design of the desired business network, we select the following three patterns as matching the three goals above. The patterns are listed with their matching capability and the role benefiting from the matching capability.

- Pattern Screening (SCR) with matching capability:
 - **C1: Screen businesses**

and benefiting role Intermediary. (The full description is in Appendix E Library of Value Patterns on page 319;)

- Pattern Registration (REG) with matching capability:
 - **C1: Collect client information**

and benefiting role **Intermediary**. (The full description is in Appendix E Library of Value Patterns on page 313;) and

- Pattern Rating (RAT) with matching capability:
 - **C1: Reduce anonymity**

and benefiting role **Intermediary**. (The full description is in Appendix E Library of Value Patterns on page 311.)

In this chapter, we use the three patterns to illustrate the synthesis procedure for value models. Figure VI-1 presents the value models of the patterns, respectively: Figure VI-1 (a) is the Screening value model; Figure VI-1 (b) is the Registration value model, and Figure VI-1 (c) is the Rating value model.

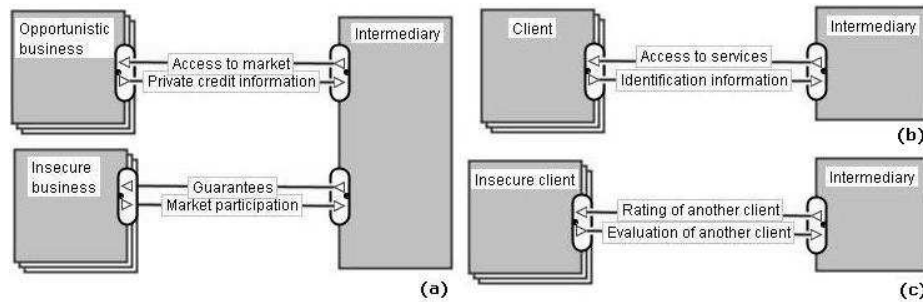


Figure VI-1. Value models: (a) – Screening pattern, (b) –Registration pattern, and (c) – Rating pattern

2.2 Synthesis Procedure for Value Patterns

Below, we present the synthesis procedure in pseudo code (see Figure VI-2.) It has two parts: Instantiate patterns and Synthesize patterns (indicated with comments). The first part is a preparation for the actual synthesis in which the value model of every pattern is modified such that the actors and value exchanges reflect the particular business at hand. In the second part, value models of patterns are linked such that they make one value model. They are synthesized one by one regardless of any order.

```

1: #comment: Part one - Instantiate patterns
2: FOR EACH pattern
3:     Map role of matching capability to business. #1a
4:     Resolve remaining roles in pattern. #2m
5:     Modify value objects. #3m
6: END FOR EACH
7: #comment: Part two - Synthesize patterns
8: WHILE pattern
9:     Find identical actors. #4a
10:    Resolve cardinality between identical actors. #5a
11:    Add the pattern to the value model. #6a

```

```
12:      Merge identical actors.                               #7a
13: END WHILE
```

Figure VI-2. Pseudo code of the synthesis procedure for value patterns. (Legend: the commented number at the end of the line is the step number. The suffix ‘a’ means automatic step; the suffix ‘m’ means manual step.)

To synthesize the selected patterns means to link their instantiated value models in a single value model. For clarity of presentation, we refer throughout this sub-section to the instantiated value models of patterns as patterns or pattern models. Correspondingly, the synthesized value model is referred to as the value model or the design.

We now discuss in detail statements 3, 4, 5, 9, 10, 11, and 12 in the pseudo code in Figure VI-2. These statements are referred to as steps: e.g., step 4 is statement number 9.

2.2.1 Part one – Instantiate Patterns

Instantiation of patterns is necessary because the value models of patterns are template-like structures which represent abstract relation between entities. Actors and value objects within value models of patterns need to be instantiated such that they represent the actual businesses and objects exchanged in the business at hand.

Actors¹ in pattern value models represent roles, where a role is associated behaviour with a business (see Chapter II Design Knowledge in Existing e-Business Models, Section 4 Documenting Patterns.) They are instantiated by finding the businesses in the concrete system under development playing the roles in the pattern.

Roles in a pattern and actors in an instantiated value model must be unique but not necessarily related with one-to-one mapping. In the description of every pattern, there is a statement of the form: “the pattern contains a *particular number of roles played by a range of possibilities for actors.*” In every pattern in the library, the number of actors is less than or equal to the number of roles: i.e., an actor can play two roles but no role can be played by two actors. Thus, the instantiation of roles (1) must not introduce more actors than roles there are and (2) must guarantee unique actors.

A pattern value model contains abstract value objects such as **product** or **service**. In the concrete system under development, the product or service are specific instances such as **book** or **transportation**.

Value objects, before and after instantiation, do not need to be unique: i.e., one object may occur several times in a model. The only constraints on the instantiation of value objects are that it does not change (1) the direction of exchange, (2) the exchange participants and (3) the number of value objects.

A. Step 1: Map the Role of the Matching Capability to a Business

A pattern was selected for synthesis because one of its top-level capabilities matched a goal from the goal model of the system under development. From the pattern description, the top-level capability benefits a particular role in the pattern. Correspondingly, the goal in the goal model belongs to a certain business. Because the capability represents something in the design that makes the goal happen, we conclude that the role that

¹ The *e³-value* modeling notation lacks primitives to express roles in the sense meant by the description of a pattern. We use instead the concept of an actor in the scope of a pattern description.

benefits from capability is played by the business having this goal. In other words, the first role from the pattern to instantiate is the role associated with the matched capability and the instance is the actor having the goal.

This step offers an automatic mapping of roles and businesses playing these roles.

The mapping of roles and business actors is captured in a correspondence table, in which the first column lists the roles as named in the pattern and the second column contains the names of actor as in the business at hand. The table is filled in partially within this step: only mappings resulting from the matching capabilities and goals are added. These are shown as rows in bold font. For an example, see further on Table VI-1 of the example of this section.

The above approach of mapping roles and actors requires that no two roles benefit from one capability. In such a way, we guarantee that within this step, *Map roles of matching goals and capabilities*, the mapping of roles and actors is one-to-one. The possibility of an actor to play more than one role (the one-to-many relation between a role and an actor) as discussed before is shown in the next step, *Resolve the remaining roles in the pattern*.

All patterns that are currently available in the library comply with the restriction on benefiting roles from a single capability. The restriction was not imposed during the identification or description of the patterns. Patterns that are newly added to the library need to meet that constrain.

Example Step 1. We exemplify step 1 of the synthesis procedure (see Figure VI-2, statement 3) with the Screening pattern. The pattern is selected for synthesis because its capability **C2** matches goal **G1**, as given in Section 2.1. Because (i) **G1** is a goal of MemEx and (ii) capability **C2** benefits the Intermediary role, we establish a direct instantiation relationship between the MemEx actor and the Intermediary role. Table VI-1 shows this.

Table VI-1. Correspondence table for roles in the Screening pattern, partially filled in

<i>Roles in the Screening pattern</i>	<i>Business actors in the example</i>
Opportunistic business	
Intermediary	MemEx
Insecure business	

Following the same line of reasoning, we find the following direct correspondence relationships in the remaining two patterns:

- MemEx and the Intermediary role in the Registration pattern, as capability **C1** matches goal **G2** (Section 2.1) and
- MemEx and the Intermediary role in the Rating pattern, as capability **C1** matches goal **G3** (Section 2.1.)

The mapping results are presented in bold font in the correspondence tables for the Registration and Rating patterns, respectively in Table VI-3 and Table VI-4.

B. Step 2: Resolve the Remaining Roles in the Pattern

Usually the direct mappings between roles and actors determined in the previous step do not cover all roles in a pattern. This means that patterns add new roles that need to be

instantiated. The information to do this is not available from the selection procedure. It needs to be uncovered by a person interpreting the textual descriptions of the matching goal and pattern, including: goal description and pattern context, description, problem, solution, and capabilities. The result of this manual operation is a mapping between a role and the most probable business actor to execute the role.

The step may result in one actor assigned to play a second role. The decision to map more than one role to a business is taken by a business analyst who has knowledge not present in the goal model and the patterns. In this way, one-to-many relationships are introduced in the correspondence table which are later resolved by unifying actors with same names.

This step completes the correspondence table between roles and actors. (For an example, see further on Table VI-2 of the example of this section.)

Example Step 2. Step 2 is also illustrated with the Screening pattern. The synthesis procedure (see Figure VI-2, statement 4) requires mapping the remaining two roles: Opportunistic business and Insecure business (see Table VI-1.) From the specification of goal **G1: Require supplier of memory integrated circuits to file credit information prior to making an offer**, we deduce that the role that gives private credit information in the value model of the Screening pattern is played by the suppliers of memory integrated circuits. Thus, we map the Opportunistic business role to the Supplier actor. Furthermore, the Insecure business role remains to be played by the buyers of the memory circuits. We assert this by mapping the Insecure business role to the Buyer actor. See Table VI-2 for the result. (The mapping in bold font is from the previous step.)

Table VI-2. Correspondence table for roles in the Screening pattern

<i>Roles in the Screening pattern</i>	<i>Business actors in the example</i>
Opportunistic business	Supplier
Intermediary	MemEx
Insecure business	Buyer

The remaining unmapped roles from the Registration and Rating patterns are determined with reasoning from the formulation of goals **G2** and **G3** (Section 2.1), and the description of the Registration and Rating patterns. The derived mappings are shown in Table VI-3 and Table VI-4. (The mappings in bold font are from the previous step.)

Table VI-3. Correspondence table for roles in the Registration pattern

<i>Roles in the Registration pattern</i>	<i>Business actors in the example</i>
Intermediary	MemEx
Client	Buyer

Table VI-4. Correspondence table for roles in the Rating pattern

<i>Roles in the Rating pattern</i>	<i>Business actors in the example</i>
Intermediary	MemEx
Insecure client	Supplier

C. Step 3: Modify Value Objects

Similarly to roles, value objects within value models of patterns need to be instantiated to represent the actual objects exchanged in the business at hand. Not every value object needs to be modified. For example, a value object **Fee** will most probably remain unchanged in every instance of the pattern. The step is meant for abstractly modelled value objects such as **Product**, **Service**, and **Information**. These need to be replaced with particular value objects such as **Book**, **Transportation**, and **Stock quote**.

The necessary information for the modification of value objects needs to be uncovered by a person interpreting the textual descriptions of the matching goal and pattern. The modification includes only changing the value object (the name of the value object): the direction of exchange, the participants and the number of objects does not change.

The mapping of abstract and concrete value objects is captured in a correspondence table similar to the one for roles and actors. The first column of the table contains the value objects as in the pattern and the second column the values objects in the example. For an example, see further on Table VI-5 of the example of this section.

Example Step 3. Step 3 of the synthesis procedure (see Figure VI-2, statement 5) instantiates the abstract value objects in the patterns. We exemplify this with the Rating pattern depicted in Figure VI-1 (c.) The two value objects, **Rating of another client** and **Evaluation of another client**, need to be made concrete by specifying who exactly is the ‘another client’. From the goal that the Rating pattern matches, namely goal **G3: Ask suppliers to evaluate buyer**, we infer that the client in question is the buyer. Thus, the value objects change to **Rating of a particular buyer** and **Evaluation of a buyer**. Table VI-5 shows the result of the mapping.

Table VI-5. Correspondence table for value objects in the Rating pattern

<i>Value objects as in the Rating pattern</i>	<i>Value objects as in the example</i>
Rating of another client	Rating of a particular buyer
Evaluation of another client	Evaluation of a buyer

Using a manual process again, we find the concrete value object for patterns Registration and Rating. In particular, the Screening pattern (see Figure VI-1 (a)) does not change any of the value objects. Therefore, the correspondence table contained in Table VI-6 repeats the abstract value objects as concrete value objects.

Table VI-6. Correspondence table for value objects in the Screening pattern

<i>Value objects as in the Rating pattern</i>	<i>Value objects as in the example</i>
Access to market	Access to market
Private credit information	Private credit information
Guarantees	Guarantees

Market participation	Market participation
----------------------	----------------------

The Registration pattern from Figure VI-1 (b) needs a change only in one of the value objects, namely Access to services. The particular service that the intermediary gives access to is the ability to transact with other businesses. Thus, the concrete value object is Enable transactions with suppliers. Table VI-7 gives the mapping between value objects.

Table VI-7. Correspondence table for value objects in the Registration pattern

<i>Value objects as in the Rating pattern</i>	<i>Value objects as in the example</i>
Access to services	Enable transactions with suppliers
Identification information	Identification information

D. Result: Instantiated pattern value models

The instantiation of value models is (1) substituting the roles with the corresponding actors and (2) substituting the abstract value objects with the concrete value objects for the business at hand. The instantiation of the Screening, Registration and Rating patterns is shown in Figure VI-3. The structure of the models remains the same; only the labels of actors and objects change.

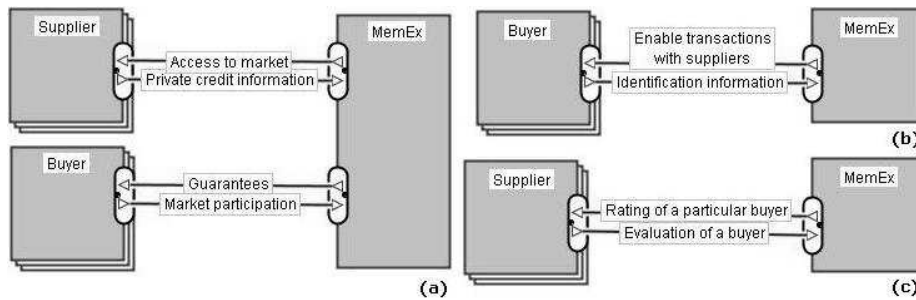


Figure VI-3. Instantiated value models: (a) – Screening pattern, (b) – Registration pattern, and (c) – Rating pattern

2.2.2 Part two – Synthesize Patterns

The synthesis begins with an empty design. We pick indifferently of the order a pattern and the value model of this pattern becomes the design. Then again indifferently of the order, we pick another pattern and add it to the design. The result is an incrementally synthesized new design to which we iteratively add all remaining patterns.

Below, we describe how the actual synthesis works. This is an automatic process.

A. Step 4: Find Identical Actors

For every pattern in a process of adding to the design, we have to find the identical actors in the pattern and in the design. The actors in the pattern value model and the actors in

the synthesized value model are unique: i.e., they have unique names. Thus, we find the identical actors in the two models by string-matching of their names. The result of this step is a list of names each of which refers to two actors, respectively: one in the instantiated pattern value model and one in the synthesized value model.

B. Step 5: Resolve Cardinality between Identical Actors

Identical actors may have different cardinalities in the pattern and in the value model. That is, an actor may be modelled as a single entity in the value model of the pattern but represented as a group of actors in the synthesized model. It is also possible that in both models the actors are represented as groups of actors and each group includes a different number of participants. This requires one of the models to be migrated to the cardinality of the other. We assume that if a certain constellation of value exchanges holds for one actor then it will also hold for many actors. Thus, we change the single actor into a group of actors. In case of two groups of actors with different number of participants, we take the greater number.

The example in this chapter does not have matching actors with different cardinalities. For an example, we refer to Chapter VIII Validation of the Design Framework, Section 4.2 Synthesis of Value Patterns, where we synthesize a real-life example.

C. Step 6: Add the Pattern to the Value Model

In this step (step 6, statement 11 in Figure VI-2), the instantiated value model is added to the design: i.e., the pattern is copied into the design. By doing this, all actors and exchanges are kept. As a result, the value model contains redundant information, namely the identical actors discovered in step 4 *Find identical actors* (statement 11 in Figure VI-2.) The redundant actors are removed in the next step.

D. Step 7: Merge Identical Actors

In this step (step 7, statement 12 in Figure VI-2), we merge the duplicated actors. The identical actors are replaced with one of them which takes over the exchange interfaces of the others. The actors whose exchanges are taken over are removed.

E. Example

The synthesis of the value model takes three iterations as there are three patterns. Because the design is initially empty, the addition of the first pattern results in a model identical with this pattern. We choose to start with the Screening pattern. (We note that there are no arguments to selecting this one. Patterns are chosen indifferently of the order.) Thus, the intermediate result for the value model looks such as the pattern itself, see Figure VI-4

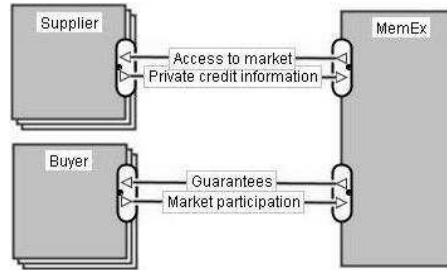


Figure VI-4. Intermediate result for the synthesized value model after the first iteration. Copy of Figure VI-3 (a)

The next pattern to add is the **Registration** pattern. The instantiated value model is presented in Figure VI-3 (b). As we can see from the intermediate value model and the value model of the pattern, there are two identical actors in the models: **MemEx** and **Buyer**. After finding the identical actors, which is step 4 in our procedure, we proceed with the next step, which ensured that the cardinalities of the identical actors are equal. We skip this step because the cardinalities match. In step 6, *Add the pattern to the value model*, we copy the pattern into the value model, which results into the model shown in Figure VI-5.

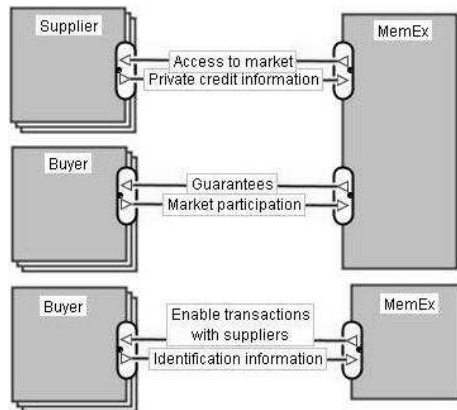


Figure VI-5. Intermediate result for the synthesized value model after step 6 in the second iteration

In step 7, we merge the redundant actors, which results in the model shown in Figure VI-6.

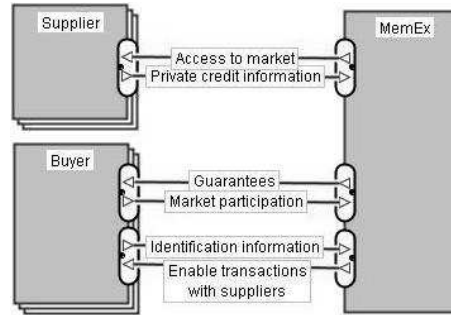


Figure VI-6. Intermediate result for the synthesized value model after the second iteration

In the third iteration, we repeat the cycle with the Rating pattern, Figure VI-3 (c.) We find again that the MemEx actor occurs in the value model and in the pattern. Moreover, the Supplier actor has also two identical occurrences. In both matches of actors, the cardinalities are equal. This leads to merging of the actors, see Figure VI-7.

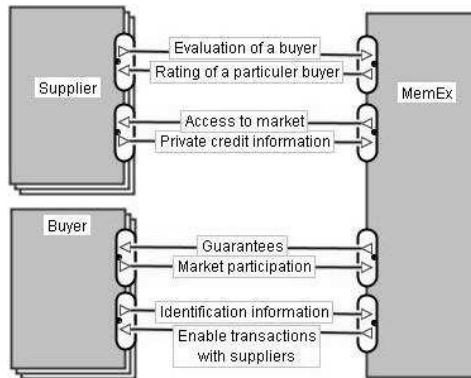


Figure VI-7. Synthesized value model

Figure VI-7 presents the final synthesized value model. We do not add more actors or exchanges because all goals from the goal model were matched with patterns. This is the model to be checked for consistency with the synthesized process model.

3 Synthesis of Process Fragments

3.1 Example

Let us assume that we have the following example case. A company provides a service and it wants to implement a business process that handles customers' requests for the service. Additionally, we assume that this process takes care of the payment for the service. For concreteness, we name the company providing the service DeliSer and, respectively, the company using the service is called ConSer. DeliSer has a market power

and can determine the terms of the contracts it signs with its clients. Therefore, it wants to realize a simple process with fixed offers which cannot be negotiated. For the same reason, DeliSer requires the payment be done immediately after the agreement is reached. The actual delivery of the product is not modelled in the process model. The analysis of the objectives (we assume them our top-level goals) leads to lower level goals such as:

- **G1: Determine the terms of the contracts and**
- **G2: Require immediate payment.**

We do not show the complete goal model here because it is outside the scope of this chapter.

Executing the selection procedure (Chapter V Selection Procedure) with the goals above, we select the following two patterns which constitute a possible design of the desired business process. The patterns are listed with their matching capability and the role benefiting from the matching capability.

- Pattern Take it or leave it (TOL) with matching capability:

- **C1: Dictate contract conditions**

and benefiting role **Supplier** (The full description is in Appendix G Library of Process Patterns on page 375) and

- Payment (PAY) with matching capability

- **C1: Receive payment**

and benefiting role **Provider** (The full description is in Appendix G Library of Process Patterns on page 367.)

Figure VI-8 presents the process models of the patterns, respectively: Figure VI-8 (a) is the Take it or leave it process model and Figure VI-8 (b) is the Payment process model.

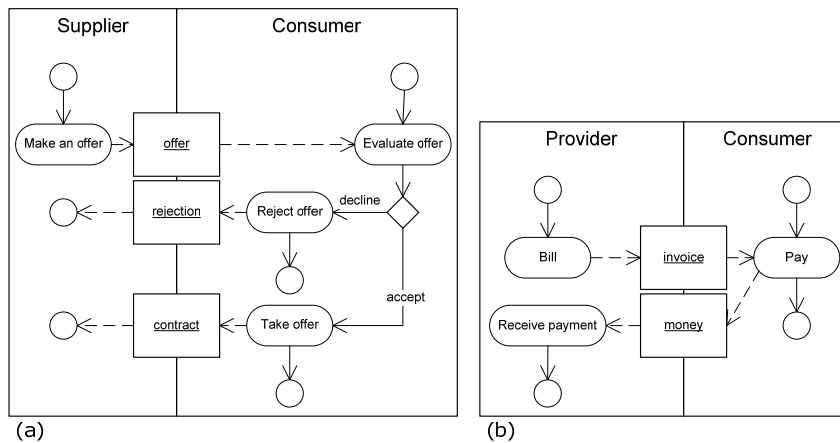


Figure VI-8. Process models: (a) – Take it or leave it pattern and (b) – Payment pattern

In this chapter, we use the two patterns to illustrate the synthesis procedure for process models.

3.2 Synthesis Procedure for Process Patterns

Below, we present the synthesis procedure in pseudo code (see Figure VI-9.) It has two parts: Instantiate patterns and Synthesize patterns (indicated with comments.) The first part is a preparation for the actual synthesis. Therein, the process model of every pattern is modified such that the swimlanes, activities and messages reflect the particular business at hand. In the second part, process models of patterns are linked such that they make one process model.

```

1: #comment: part one - Instantiate patterns
2: FOR EACH pattern
3:     Map role of matching capability to business. #1a
4:     Resolve remaining roles in pattern. #2m
5:     Modify activities. #3m
6:     Modify messages. #4m
7: END FOR EACH
8: #comment: part two - Synthesize patterns
9:     Determine swimlanes. #5a
10:    Arrange activities. #6a
11:    Add missing activities and messages. #7m
12:    Link connectors. #8m

```

Figure VI-9. Pseudo code of the synthesis procedure for process models. (Legend: the commented number at the end of the line is the step number. The suffix 'a' means automatic step; the suffix 'm' means manual step.)

To synthesize the selected patterns means to link their instantiated process models in a single process model. For clarity, we refer throughout this sub-section to the instantiated process models of patterns as patterns or pattern models. Correspondingly, the synthesized process model is referred to as the process model or the design.

Further, we discuss in detail statements 3, 4, 5, 6, 9, 10, 11, and 12 in the pseudo code in Figure VI-9. These statements are referred to as steps: e.g., step 4 is statement 6.

3.2.1 Part one – Instantiate Patterns

Instantiation of patterns is necessary because the process models of patterns are abstract sequences of activities coordinated by messages which need to be made concrete for the particular case at hand. Furthermore, each activity has to be assigned to a concrete business that executes it.

Pattern descriptions contain a notion of a role, where a role is the associated behaviour with an interaction participant. Roles are modelled as swimlanes and the relationship is one-to-one. Swimlanes group activities such that if executed by one participant then the participant exhibits certain behaviour. In the process model of the system, swimlanes also model grouping of activities based on performing businesses. Thus, the instantiation of a pattern swimlane is finding a business that play the role modelled with the swimlane.

Although one pattern swimlane always represents one role, one business can play several roles. Thus, two swimlanes can be instantiated with one performing business, which leads to a single swimlane in the instantiated model.

Activities within process models of patterns need to be instantiated such that they represent the actual activities performed by businesses. A pattern process model contains abstract activities such as **Deliver service**. In the concrete system under development, the delivery of the service is a specific instance, such as, e.g., **Return keyword search result**. Activities, before and after instantiation, need to be unique: i.e., one activity must occur only once in a model. Furthermore, the instantiation does not change the swimlanes, sequence, or number of activities.

Messages within process models of patterns need also to be instantiated such that they represent the actual messages exchanged by the businesses at hand. A pattern process model contains abstract messages, such as **Request**. In the concrete system under development, the message is specific, such as, e.g., **Search keywords**. Messages, before and after instantiation, need to be unique: i.e., one message must occur only once in a model. Furthermore, the instantiation does not change the sender, receiver, sequence, or number of messages.

A. Step 1: Map the Role of the Matching Capability to a Business

A pattern was selected for synthesis because one of its top-level capabilities matched a goal from the goal model of the system under development. From the pattern description, the top-level capability benefits a particular role in the pattern. Correspondingly, the goal in the goal model belongs to a certain business. Because the capability represents something in the design that makes the goal happen, we conclude that the role that benefits from capability is played by the business having the goal. In other words, the first role from the pattern to instantiate is the role associated with the matched capability and the instance is the business having the goal.

This step offers an automatic mapping of roles and businesses playing the roles.

The mapping of roles (swimlanes) and businesses is captured in a correspondence table, in which the first column lists the swimlanes as named in the pattern and the second column contains the names of businesses (swimlanes) as in the business at hand. The table is filled in partially within this step: only mappings resulting from the matching capabilities and goals are added. These are shown as rows in bold font. For an example, see further on Table VI-8 of the example of this section.

Example Step 1. We exemplify step 1 in the synthesis procedure for process patterns (see Figure VI-9, statement 3) with the **Take it or leave it** pattern. The pattern is selected for synthesis because its capability **C1: Dictate contract conditions** allows DeliSer to dictate the term of the contracts it closes. Capability **C1** benefits the **Supplier** role; therefore, we establish a direct instantiation relationship between the DeliSer business and the **Supplier** role. Table VI-8 shows this.

Table VI-8. Correspondence table for roles in the **Take it or leave it** pattern, partially filled in

<i>Roles (swimlanes) as in the Take it or leave it pattern</i>	<i>Businesses (swimlanes) as in the example</i>
Supplier	DeliSer
Consumer	

The **Payment** pattern is selected for synthesis because of its capability **C1: Receive payment** which allows DeliSer to bill its clients. Capability **C1** benefits the **Provider** role; therefore, we establish a direct instantiation relationship between the DeliSer business and the **Provider** role. Table VI-9 shows this.

Table VI-9. Correspondence table for roles in the **Payment** pattern, partially filled in

<i>Roles (swimlanes) as in the Payment pattern</i>	<i>Businesses (swimlanes) as in the example</i>
Provider	DeliSer
Consumer	

B. Step 2: Resolve the Remaining Roles in the Pattern

This step is the same as Step 2 from the synthesis procedure for value pattern. Applying manually knowledge from textual descriptions of goals and patterns, the result is a mapping between roles and most probable businesses to execute the role.

The step may result in one business assigned to play a second role. This may lead to one-to-many relationships in the correspondence table, which is later resolved by unifying swimlanes with same names.

This step completes the correspondence table between roles and businesses. (For an example, see further on Table VI-10 of the example of this section.)

Example Step 2. Step 2 of the synthesis procedure (see Figure VI-9, statement 4), applied on the **Take it or leave it** pattern, requires mapping of the remaining role **Consumer** (see Table VI-8.) From the description of the business at hand, we deduce that the company called ConSer plays the role. Thus, we map the **Consumer** role to the ConSer business. Table VI-10 indicates the result. (The mappings in bold font are from the previous step.)

Table VI-10. Correspondence table for roles in the **Take it or leave it** pattern

<i>Roles (swimlanes) as in the Take it or leave it pattern</i>	<i>Businesses (swimlanes) as in the example</i>
Supplier	DeliSer
Consumer	ConSer

In the **Payment** pattern, the remaining role that requires mapping is **Consumer** (see Table VI-9.) From the description of the business at hand, we deduce that the role is played by the company called ConSer. Thus, we map the **Consumer** role to the ConSer business. Table VI-11 shows the result. (The mappings in bold font are from the previous step.)

Table VI-11. Correspondence table for roles in the **Payment** pattern

<i>Roles (swimlanes) as in the Payment pattern</i>	<i>Businesses (swimlanes) as in the example</i>
Provider	DeliSer
Consumer	ConSer

C. Step 3: Modify Activities

Similarly to roles, activities within process models of patterns need to be instantiated to represent the actual activities performed by businesses. Not every activity needs to be modified. For example, an activity **Send invoice** will most probably remain unchanged in every instance of the pattern. The step is meant for abstractly modelled activities such as **Deliver service**. This needs to be replaced with a particular activity such as **Return keyword search result**.

The necessary information for the modification of activities needs to be uncovered by a person interpreting the textual descriptions of the matching goal and pattern. The modification includes only changing the name of the activity: the sequence, the swimlane and the number of activities do not change.

The mapping of abstract and concrete activities is captured in a correspondence table similar to the one for swimlanes. The first column of the table contains the activities as in the pattern and the second column the activities as in the business. For an example, see further on Table VI-12 of the example of this section.

Example Step 3. Step 3 of the synthesis procedure (see Figure VI-9, statement 5) instantiates the abstract activities in the patterns. Activity **Make an offer** from the **Take it or leave it** pattern needs to be made concrete by specifying that it is a response to a request from a client. Thus, the activity changes to **Respond to request for quotation**. The remaining activities from the same pattern are informative enough in their abstract form and, therefore, we do not change them. Table VI-12 shows the result of the mapping.

Table VI-12. Correspondence table for activities in the **Take it or leave it** pattern

<i>Activities as in the Take it or leave it pattern</i>	<i>Activities as in the example</i>
Make an offer	Respond to request for quotation
Evaluate offer	Evaluate offer
Reject offer	Reject offer
Take offer	Take offer

In the **Payment** pattern, activity **Bill** needs to be made concrete by specifying that it is preparation and sending of an invoice. Thus, the activity changes to **Send invoice**. The remaining activities are informative enough in their abstract form and, therefore, we do not change them. Table VI-13 shows the result of the mapping.

Table VI-13. Correspondence table for activities in the **Payment** pattern

<i>Activities as in the Payment pattern</i>	<i>Activities as in the example</i>
Bill	Send invoice
Receive payment	Receive payment
Pay	Pay

D. Step 4: Modify Messages

Messages within process models of patterns need to be instantiated to represent the actual messages exchanged by businesses. Not every message needs to be modified. For example, a message *invoice* will most probably remain unchanged in every instance of the pattern. The step is meant for abstractly modelled messages such as *request*. This needs to be replaced with a particular message such as *search keywords*.

The necessary information for the modification of messages needs to be uncovered by a person interpreting the textual descriptions of the matching goal and pattern. The modification includes changing the name of the message only. Note that the sender, the receiver, and the number of messages do not change.

The mapping of abstract and concrete messages is captured in a correspondence table similar to the one for swimlanes. The first column of the table contains the messages as in the pattern and the second column the messages as in the business. For an example, see further on Table VI-14 of the example of this section.

Example Step 4. Step 4 of the synthesis procedure (see Figure VI-9, statement 6) instantiates the abstract messages in the patterns. Message *contract* from the *Take it or leave it* pattern needs to be made concrete by specifying that it is a purchase order submitted by a client. Thus, the message changes to *purchase order*. The remaining messages are informative enough in their abstract form and, therefore, we do not change them. Table VI-14 shows the result of the mapping.

Table VI-14. Correspondence table for messages in the *Take it or leave it* pattern

<i>Messages as in the Take it or leave it pattern</i>	<i>Messages as in the example</i>
offer	offer
rejection	rejection
contract	purchase order

For the *Payment* pattern, all messages are informative enough in their abstract form and, therefore, we do not change them. Table VI-15 shows the result of the mapping.

Table VI-15. Correspondence table for messages in the *Payment* pattern

<i>Messages as in the Payment pattern</i>	<i>Messages as in the example</i>
invoice	invoice
money	money

E. Result: Instantiated pattern process models

The instantiation of process models is (1) substituting the roles with the corresponding businesses; (2) substituting the abstract activities with concrete activities executed by the businesses; and (3) substituting the abstract messages with concrete messages exchanged by the businesses. The instantiation of the *Take it or leave it* and *Payment* patterns is shown in Figure VI-10. The structure of the models remains the same; only the labels of swimlanes, activities and messages changes.

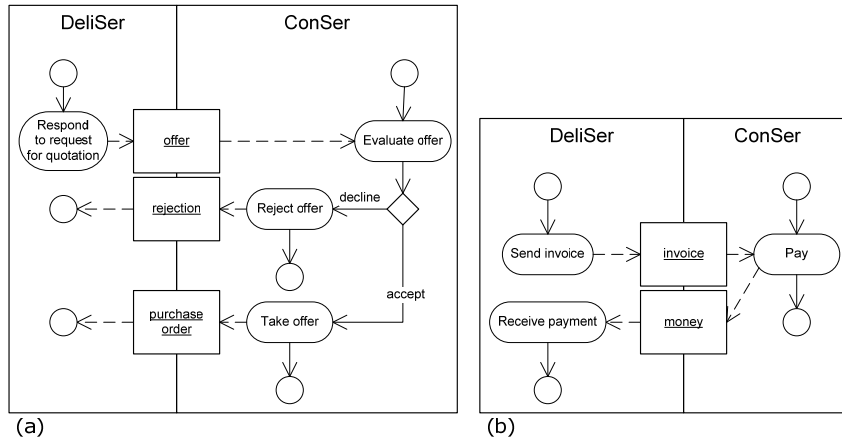


Figure VI-10. Instantiated process models: (a) – Take it or leave it pattern and (b) – Payment pattern

3.2.2 Part two – Synthesize Patterns

The synthesis begins with a design that contains only a start symbol followed by a parallel split. We pick all patterns participating in the process model and add them into the design. The synthesis is a semi-automatic process during which the sequence of activities from different patterns is decided and, eventually, missing activities or messages are added.

Below, we describe how the actual synthesis works.

A. Step 5: Determine Swimlanes

In this step (see Figure VI-9, statement 9), we determine the number and names of swimlanes in the design. This step is automatic.

The activities of every pattern are grouped in two or more swimlanes. Swimlanes are identified with their names. Bringing all patterns together in one design may lead to two or more swimlanes with the same name. Because identical swimlanes model one business, we represent only one swimlane in the process model and put all activities from the various patterns into this swimlane.

The number of unique businesses playing the roles from all patterns determines the number of swimlanes in the design and the cardinality of the parallel split in the initial design. After this step, the design contains the start symbol, a particular number of swimlanes, and one connector¹, linked with the parallel split, in each swimlane. For an example, see further on Figure VI-11 of the example of this section.

Example Step 5. Our starting position is an initial design which contains a start symbol followed by a parallel split and the two instantiated process patterns in Figure VI-10. Each of the two patterns that we want to synthesize has two swimlanes called DeliSer

¹ We introduce the connector concept for the description of process patterns, see Chapter II Design Knowledge in Existing e-Business Models. A connector models the beginning or end of a control flow.

and ConSer. Therefore, we determine that the process model will have two swimlanes called DeliSer and ConSer, and that the parallel split will have two connectors. The intermediate result of step 5 is shown in Figure VI-11.

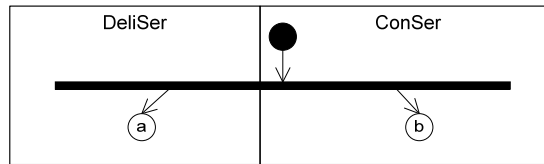


Figure VI-11. Intermediate result for the synthesized process model after step 5

B. Step 6: Arrange Activities

In this step (see Figure VI-9, statement 10), the process fragments of the patterns are put together in the design. This step is automatic.

The process fragment of every pattern contains activities sequenced in a certain order with control flow and messages. When a process fragment is added to the design, the order of execution of activities does not change. The only manipulation of the pattern process model is allocating the activities to the right swimlanes; i.e., every activity is placed in a swimlane with the same name as in the instantiated pattern. This may cause the pattern to visually look different but with respect to sequence of activities and messages nothing will change.

All pattern process models are added at once. Assuming an intuitive execution order from top-to-bottom, the patterns are positioned indifferently of this order on the page. Later in Step 8, the correct execution order of patterns is determined by linking their connectors. To enable this, the current step assigns unique names to connectors.

Example Step 6. In this step, the process fragments from the two patterns are copied into the process model. The activities are placed in the appropriate swimlanes and the connectors are given unique identifiers. The intermediate result of step 6 is shown in Figure VI-12.

In this step (see Figure VI-9, statement 11), we add process specific activities and messages which are not present in any pattern. This step is manual.

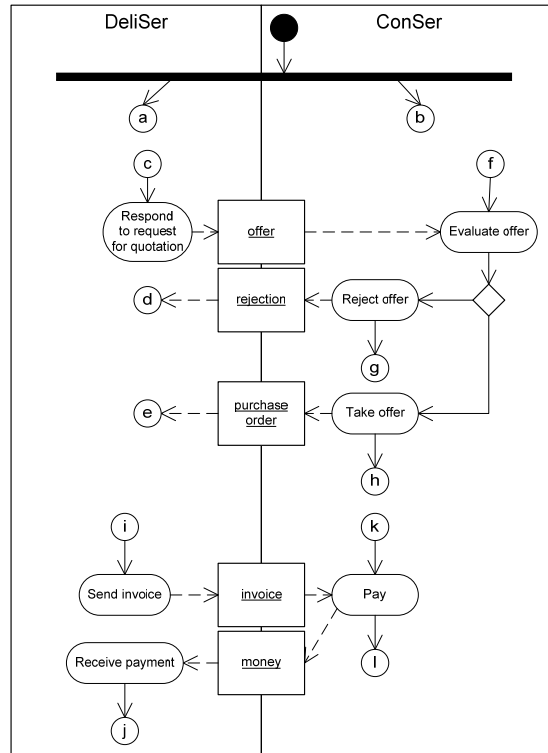


Figure VI-12. Intermediate result for the synthesized process model after step 6

Not every process model can be designed completely out of patterns. Businesses want to differentiate and, therefore, they need customized processes. Furthermore, a specific behaviour may not be present in the library. Then, this has to be added manually into the design. In this step, the designer developing the process perspective can add missing fragments.

C. Step 7: Add Missing Activities and Messages

Example Step 7. In this step, the need of additional activities is foreseen by the designer of the perspective. In our example, we add four fragments. The first fragment, the one with connector **m** in Figure VI-13, is an activity which concludes the interaction and terminates the process. The remaining three fragments, the ones with connectors **n**, **o**, and **p**, contain only the end symbol to mark the end of the process model. The intermediate result of step 7 is shown in Figure VI-13.

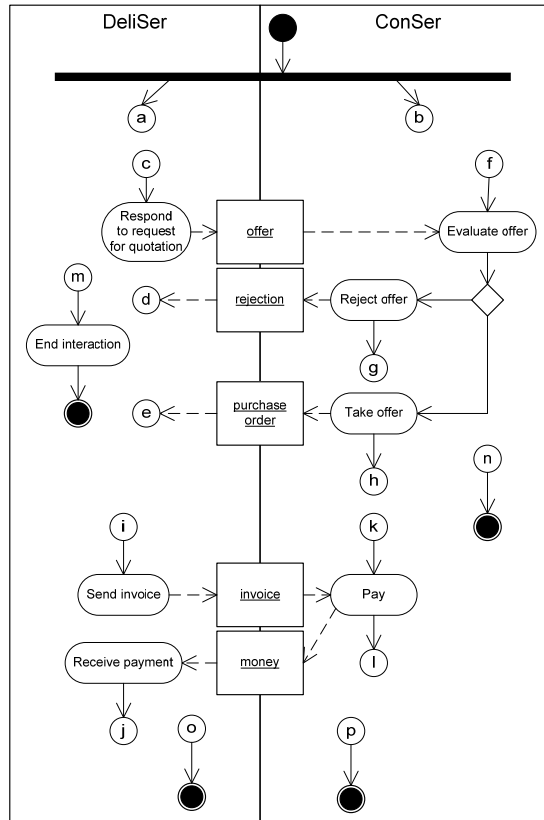


Figure VI-13. Intermediate result for the synthesized process model after step 7

D. Step 8: Link Connectors

In this step (see Figure VI-9, statement 12), we link the fragments such that they make one process model. This step is manual.

Prior to this step, the design contains disconnected fragments of a process. The boundary of every fragment is defined by the connectors this fragment has. We link connectors by pairing them. Once paired, the connectors are removed from the model and the fragments are linked with control flow.

We illustrate the linking with Figure VI-14. There we have three fragments, Figure VI-14 (a), Figure VI-14 (b), and Figure VI-14 (c). The designer decides that connector **a** from the first fragment has to be linked with connector **d** from the third fragment. Additionally, connector **b** from the first fragment has to be linked with connector **c** from the third fragment. The intermediate result of the unification of connectors is shown in Figure VI-14 (d). Finally, the connectors are removed and the synthesised model looks such as the representation in Figure VI-14 (e).

The linking of connectors is a manual process. The designer decides on what is the overall sequence of execution of patterns in the final design.

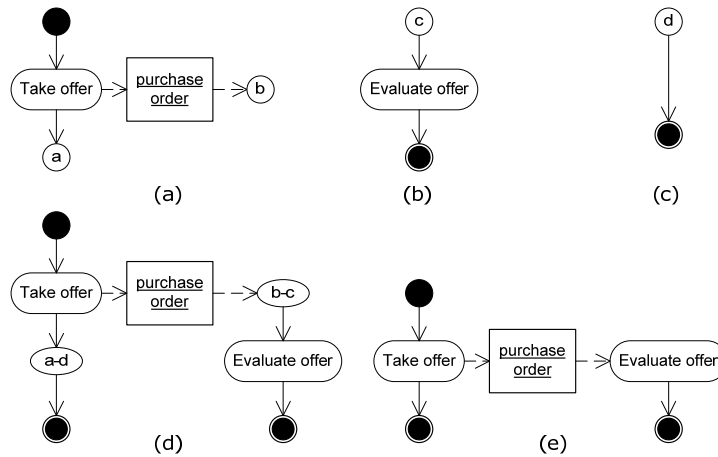


Figure VI-14. Example of linking connectors. (a) – first fragment; (b) – second fragment; (c) – third fragment; (d) – intermediate result; and (e) – final result

Example Step 8. In this step, we decide which connectors to unify and, in that matter, determine the execution sequence of the activities from the patterns. We decide that the following pairs of connectors are formed: **a – c**, **m – d**, **e – i**, **j – o**, **b – f**, **g – n**, **h – k**, and **l – p**. Some of the pairs, namely **a – c**, **e – i**, **b – f**, and **h – k**, are formed to satisfy the requirement of the **DeliSer** business of dictating the market. **DeliSer** demands payment immediately after the acceptance of its offer, see the description of the example in Section 3.1. Therefore, the **Payment** pattern follows right after the **Take it or leave it** pattern.

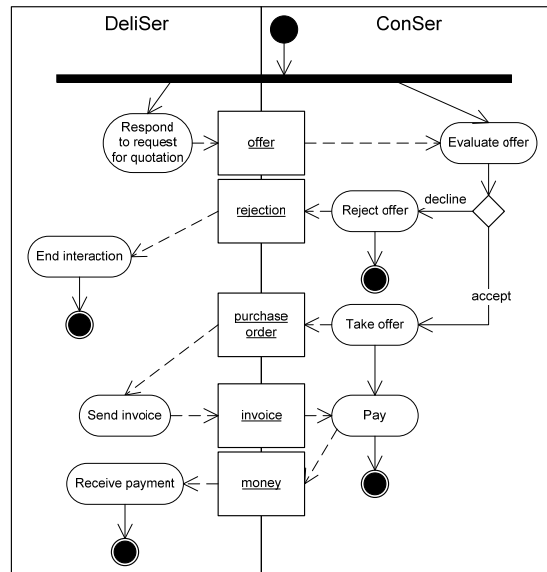


Figure VI-15. Synthesized value model

We link the paired connectors; remove the connectors; and arrive at the model shown in Figure VI-15.

Figure VI-15 presents the final synthesized process model. We do not add more swimlanes, activities or messages because all goals from the goal model were matched with patterns. This is the model to be checked for consistency with the synthesized value model.

3.3 Readability of Synthesized Process Model

Usually the process model of a system is too big to fit on one page. A standard technique to deal with the problem is to show the process in several parts. The division of the process into smaller processes is decided based on presentation needs; i.e., parts of the process are shown on one page because in that combination they are easy to follow and understand. This approach allows part of the process to appear on several pages.

A real-life process model synthesized from patterns also does not fit on one page. Because it has to be synthesized before it can be split in parts and presented to people, we need a different approach to deal with complexity. We need to divide the process model before synthesizing it. We call a part of the complete process a sub-process.

The decision which patterns to form a sub-process is taken by the designer of the perspective. One possibility is to pick patterns that match goals with a common head-goal or goals from the same branch in the goal model. We use this approach in the real-life example with which we validate the complete framework, see Chapter VIII Validation of the Design Framework.

The difference with the presentation approach is that the parts of the model are not driven by communication needs but based on requirements captured in goals. Because of that, patterns cannot take part in more than one sub-process.

4 Related work

OOram [49] is a framework for creating custom methodologies for software specifications. In its technology dimension which describes concepts, notations and tools, the framework prescribes a generic method for synthesis of design fragments. The approach is based on object-oriented modelling of software systems, where the specification contains static and behavioural models of the system under development. The synthesis idea is based on the premises that a particular object plays a specific role and that one object can play many roles. A design fragment, called also pattern in the OOram terminology, captures a number of interacting roles. Each role is played by an object. When two fragments are synthesized, the identical objects are unified and the interactions kept. The approach distinguishes two types of synthesis: safe and unsafe. The unsafe synthesis cannot guarantee the dynamic properties of the fragments in the synthesized model and, therefore, its recommended use is limited.

In comparison with the OOram synthesis approach, we apply the same idea of synthesis on the basis of roles played by objects. In the case of value pattern synthesis, the roles are business roles and the objects are business actors. Where in the case of process pattern synthesis, the roles are swimlanes and the objects are people, departments

or software systems responsible for the execution of the activities in a swimlane. The difference is that our synthesis is manual and the resulting designs are unsafe in the sense of OOram. Our synthesis procedure is meant to support the designers rather than to automatically synthesize model fragments.

Yacoub, S. and Ammar, H. [70] describe an approach to pattern-oriented analysis and design (POAD) for software engineering patterns (the type of patterns described by Gamma, E. et al. [22].) The approach includes three phases, namely: analysis, design, and design refinement. The first two correspond, respectively, to our selection of relevant patterns and synthesis of patterns steps; where the third is left outside our design method. The POAD design method is prescription of manual steps to be performed by the designers. For example, the design phase (understand synthesis step) prescribes a process including instantiation of (possibly) multiple instance of a pattern, identification of pattern relationships and construction of diagram. Although the steps resemble our approach, there two main differences: (1) the POAD method is more a guideline for manual execution of the process and (2) the construction of designs uses information about pattern dependencies from the analysis phase. These two limit the applicability of POAD to software engineering patterns.

In previous work [76], we proposed to select the building block for the new business model from the library of patterns. The selected fragments need additional work to become a coherent design for which we employ heuristics to connect the fragments and achieve simple consistency. Despite that, we did not require the composed patterns to form an internally consistent and completely developed view. We intentionally left design freedom in order to accommodate changes due to integration with other views.

5 Summary

This chapter answers research question **Q2.3: How to synthesize value and process models from design patterns?** It defines synthesis procedures for value and process patterns. These are finite iterations over a number of automatic and manual steps that guide the designer thought the process of synthesis.

Chapter VII

Consistency between Value and Process Models

The chapter answers research question **Q2.1.1**. It discusses the consistency among the models developed as part of the perspectives in our design framework. In particular, it describes an approach to consistency between value and process models. The chapter presents the approach to checking consistency between e^3 -value models and Activity diagrams; namely, a common notation to which all other models translate. It gives the transformation procedures from the e^3 -value notation and from the Activity Diagrams to the common notation. Additionally, the chapter defines consistency between value and process models and provides a consistency criterion in the common notation. The two transformation procedures and the consistency check are illustrated with an example¹.

The bold font in the table below positions the contents of the chapter with respect to the research questions and results.

<i>Research questions</i>	<i>Research results</i>	<i>Thesis chapters</i>
Q1: What is the design knowledge in existing e-business models?	R1: two libraries of value design patterns and process design pattern, correspondingly	Chapter II Design Knowledge in Existing e-Business Models
Q2: How to reuse design knowledge in the development process of an e-business specification?		Chapter III Design Framework for e-Business Models
Q2.1: What makes an e-business specification?		
Q2.1.1: How to check consistency between value and process models?	R5: a definition of consistency between value and process models, and a procedure to check the consistency criteria	Chapter VII Consistency between Value and Process Models
Q2.2: What are the relevant design patterns for a particular design?	R2: a goal-modelling technique, including propagation of satisfaction values	Chapter IV Goal-modelling
Q2.2.1: How to identify the relevant design patterns given a set of requirements?	R3: a selection procedure that identifies the most relevant patterns for a particular design given a set of requirements	Chapter V Selection Procedure
Q2.3: How to synthesize value and process models from design patterns?	R4: a synthesis approach to derive complete models from patterns	Chapter VI Synthesis of Value and Process Patterns

¹ This chapter is based on a paper published at the 2005 International Conference on Cooperative Information Systems (CoopIS) by Zlatev, Z. and Wombacher, A. [73]

1 Introduction

Our design framework (Chapter III Design Framework for e-Business Models) initially had three perspectives. Despite that the perspectives were later reduced to two, the framework remains multi-perspective. Each perspective results in a separate model of the desired system. The models are produced within one framework but each is developed independently of the other. Although, there is only one system under development, the understanding of the system may differ from perspective to perspective due to differences in the methodologies followed, conflicting knowledge about the domain, or opposing stakeholders' goals.

Adopting a multi-perspective approach to system development presupposes a decentralized development process. Models are developed independently from each other and, usually, they result in a distributed design¹. The benefits of reducing the complexity by analyzing certain requirements isolated from the others come at a price of potentially inconsistent specifications. A set of models is inconsistent if at least two models contain statements about the future system that mutually contradict each other so that the system cannot be implemented. Therefore, specifications need to be checked for consistency to identify required changes in the final design.

We define consistency as a relation between two (or more) models such that the specification does not contain contradicting statements, thereby preventing from implementing the system. Consequently, two (or more) models are consistent, if an implementation of the specification can be built. Moreover, two (or more) models are inconsistent if it is not possible to build a single system that correctly implements each model.

Consistency of specifications is a necessary condition for the implementation of the desired system. It guarantees that the system can exist. Nevertheless, it does not guarantee that it will be the desired system. For that, additional conditions and relations among the models must hold, such as, e.g., correctness. In this chapter (and the proposed design approach), we limit ourselves to consistency.

In this dissertation, we restrict ourselves to detecting inconsistencies only and we leave the question of management [17] or immediate removal of inconsistencies out of our research scope.

In our design framework, each model is developed using the same development method in each perspective (see for an overview Chapter III Design Framework for e-Business Models.) All perspectives use a common shared goal model that represents the requirements of the system. The use of a shared goal model potentially reduces the number of conflicting statements. However, because the differences in the development methods are not the only cause (see above) of inconsistencies, we cannot assume that the models developed in our framework are consistent.

¹ The opposite of a distributed design would be the plan of a building, where the electric-wire perspective and the water-pipe perspective are independently developed but centrally coordinated through space.

2 Checking Consistency between an e^3 -value Model and an Activity Diagram

Our framework has two perspectives which result in two models: the first prescribes the system under development from an economic value point of view and the second from a business process point of view. Each of the models is expressed in a particular modelling notation: e^3 -value [25] (Appendix A e^3 -value Modelling Notation) for the economic value perspective and UML Activity diagram [45 pages 3-155—3-169] (Appendix B UML Activity Diagram Modelling Notation) for the business process perspective.

For simplicity, we refer to the economic value model as value model and to the business process model as process model. To show that we mean a model in a concrete notation, we refer to the value model as e^3 -value model or e^3 -value model and to the process model as Activity diagram.

In our approach to consistency, we define a common semantic model, which we call Reduced model. Both models, the e^3 -value and the Activity diagram, are transformed to reduced models. The consistency relation between the value and process models is, then, defined and checked in terms of reduced models. Below, we describe the approach.

Our starting position is two models: one representing the economic value perspective and one representing the business process perspective. Figure VII-1 shows these in the middle two boxes. Our objective is to check if the two models are consistent. This is not a straightforward task because the languages used to model the perspectives have concepts and relations with distinctive meanings. A way to check consistency is to interpret the specification with respect to the effects these can cause in the real world if implemented. If the two models prescribe two systems that have the same effect then the specification is consistent. Such defined consistency can be checked only by people. We call it informal consistency and Figure VII-1 shows it at the top.

We operationalize the informal consistency definition by (1) transforming the models to reduced models and (2) defining the consistency as an equivalence relationship between reduced models. Figure VII-1 shows the result of the transformation at the bottom: to the left is the reduced model derived from the value model and to the right is the reduced model derived from the process model. The equivalence relationship is defined such that the reduced models are equivalent in case the value and process models are informally consistent. (In Section 8, we show that our approach is plausible because always when two reduced models are equivalent then the value and process models are informally consistent.)

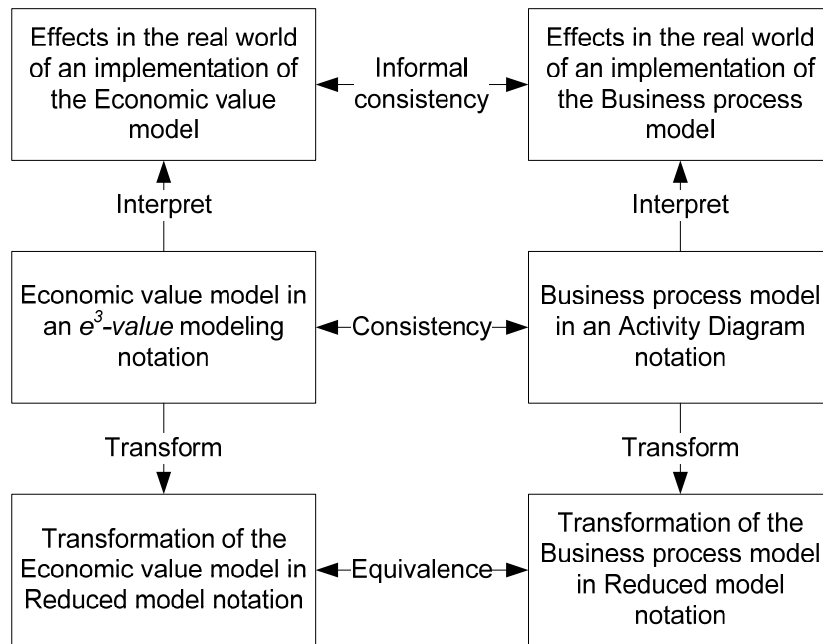


Figure VII-1. Operationalization of consistency

The approach illustrated in Figure VII-1 works with arbitrary notations for the value and process models. The particular selection influences the particular common concepts but is invariant to the proposed approach based on reduced models.

Not affecting the general applicability of the approach, we make use of the specific semantics of the concrete used notations. We modify the transformation and equivalence relations such that we make maximum use of similar concepts in the e^3 -value notation and Activity diagram. The details follow in Section 5 Reduced Model.

3 Example

We consider an example with the following businesses taking part: a buyer, a seller, and a shipping company. The seller has a shop and a warehouse at two different locations. It can directly sell products to customers only from the shop. If a product is purchased that is not present in the shop then a delivery from the warehouse must be made. A shipping company is paid to arrange the logistics to the buyer's home. The seller supports two payment methods: (1) in case of an off-the-shelf product, the seller requires immediate payment in cash; (2) in case of a purchase from the warehouse, the seller allows late payment by, e.g., a bank transfer.

3.1 Economic Value Perspective

We use the e^3 -value modelling notation [25] (see Appendix A e^3 -value Modelling Notation for details) to represent the value aspect of a business model. Figure VII-2 shows a value model of the example business described above. It contains the three businesses connected by value exchanges. The buyer is willing to give money in return for a product. The two potential exchanges are with the seller, in the middle of the figure, who has an interest in the same value objects. The dependency path, which starts in the buyer, is split at the seller. This is an OR-fork which exemplifies that some products are handed to the buyer immediately while others must be transported by a shipping company. The seller and shipper exchange the value objects Fee and Transport which are paired together reciprocally.



Figure VII-2. Value model of the example business, using e^3 -value modelling notation

3.2 Business Processes Perspective

The e^3 -value model focuses on the pairing of objects that have economic value for businesses. We now discuss the coordination of activities performed by each business to achieve the exchange of value objects. We use an UML Activity diagram [45 pages 3-155—3-169] (see Appendix B UML Activity Diagram Modelling Notation for details) to represent the business processes perspective of our example.

Figure VII-3 shows the sequence of actions performed during a purchase of a product. The process starts with the buyer requesting a product. Her order is processed and two outcomes are possible: either the desired product is present in the shop; or the product must be reserved and shipped from the warehouse. These options are represented in Figure VII-3 as a choice in the seller's swimlane. In case the path to the left (marked with 1 in the figure) is followed then the product is handed directly and payment in cash is received in return. In the second case (marked with 2 in the figure), a reservation is made. This is followed by two parallel branches which represent the payment of an invoice and the transportation of the product. The latter requires coordination with the logistics provider, which is shown as message exchanges between the seller and shipper swimlanes. The actual delivery of the product is represented in the bottom of Figure VII-3 as a message from the shipper swimlane to the buyer.

5 Reduced Model

The e^3 -value model and the activity diagram are not directly comparable. The e^3 -value model is based on value exchanges disregarding the order in which they are performed. The activity diagram is based on sequences of object flows disregarding relationships of reciprocal economic value among objects. Thus, we construct a reduced model containing the common concepts and relations of the e^3 -value model and the activity diagram to make the two models comparable.

The reduced model is used to compare abstractions of the e^3 -value model and the activity diagram. To avoid confusion of terminology as both notations use the concept *object*, we refer to objects and object flows in the activity diagram as messages and message exchanges, respectively.

In particular, the reduced model used for consistency checking consists of business units, common value objects, and common value exchanges, where:

- a *business unit* (called *unit* for short) corresponds to an actor from the e^3 -value model and a swimlane from the activity diagram. It represents organizational units (grouping of responsibilities) within a business, which are profit and loss responsible but not necessarily legal entities;
- a *common value object* (called *common object* for short) corresponds to a value object from the e^3 -value model and a message from the activity diagram. It represents an object of economic value in the e^3 -value-model sense which is used for coordination of business activities;
- a *common value exchange* (called *common exchange* for short) corresponds to a value exchange in the e^3 -value model and a message exchange in the activity diagram. It is a bilateral exchange of coordinating value objects between profit and loss responsible entities disregarding order, reciprocity and bundling.

Figure VII-4 shows the visual notation of the reduced model concepts, where (a) represents a business unit, (b) represents a common value object, and (c) represents a common value exchange.

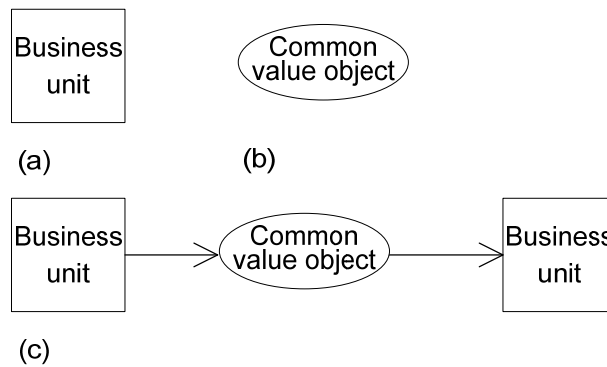


Figure VII-4. Visual notation of the reduced model: (a) – business unit, (b) – common value object, and (c) – common value exchange

A reduced model is an explicit representation of a single alternative dependency path in an e^3 -value model and of a single execution sequence in an activity diagram. It contains the value exchanges executed in one possible scenario in a business.

Common value exchanges are closely related to value exchanges because they are more generic than message exchanges which explicitly represent message ordering. Reciprocity, as contained in the value model, is not considered in the reduced model because there is no corresponding concept in activity diagrams. Alternatives are represented explicitly because OR-forks in e^3 -value models and choices in activity diagrams are not always comparable. This approach of comparing alternatives independently of each other is well known, e.g., from deciding properties of workflow models, which is often based on occurrence graphs derived from Petri nets.

Value objects can be divided into three sub-types, namely goods, services and experiences. The term product refers to both goods and services. We require that products are represented in the reduced model with common objects; whereas, we omit experiences from the reduced model as it is unlikely that these are modelled as message exchanges in an activity diagram.

An exchange of a value object in the e^3 -value-model sense corresponds to a sequence of messages exchanged between two swimlanes in an activity diagram. We will call such a sequence a transaction. Since a sequence of message exchanges does not provide a direction as a value exchange does, we select a single message exchange as a representative of the sequence. The direction of the selected message exchange reflects the direction of the value object exchange. Correspondingly, we map the message of the selected message exchange to a common object.

6 Transformations to Reduced Models

6.1 Semantic Relationships between Instances

Besides the conceptual transformation from a value and a process model to a reduced model, the instances of the concepts also have to be semantically correlated. The semantic relationship between instances can be one-to-one, one-to-many, and many-to-many. The first two relationship types can be observed in the example described in Section 3 and are covered in this section, while the many-to-many relationship is discussed in Section 9.

The transformation of an e^3 -value model or an activity diagram results in reduced models. In particular, the reduced models must be based on the same set of semantic instances of units and common objects. The existing semantic relationship between instances of actors and swimlanes is represented by two relationships: between an actor and a unit, and between the same unit and a swimlane. Correspondingly, the semantic relationship between instances of value objects and messages is represented by two relationships: between a value object and a common object, and between the same common object and a message. To restrict the relationships between actors (and value objects) and swimlanes (and messages) to one-to-one and one-to-many, we allow a unit (and a common object) to take part in at most a single one-to-many relationship.

The instances of a business unit and a common value object in the reduced model are determined by an expert who has knowledge about the instances of the corresponding concepts in the e^3 -value model and the activity diagram. The expert also determines the mapping between the instances, which is captured in transformation tables.

For the business in our example (Section 3), the mappings of actors from the e^3 -value model and activity diagram to reduced models are listed in Table VII-1 (a) and Table VII-1 (b), respectively. Due to the construction of the example, the rows in Table VII-1 contain the same actor names representing one-to-one relationships. Table VII-2 (a) lists the mapping between value objects in the e^3 -value model and the common objects in the reduced model. Again, due to the construction of the example this mapping represents a one-to-one relationship. Table VII-2 (b) lists the mapping between selected messages in the activity diagram and common objects of the reduced model. The mapping in Table VII-2 (b) contains two one-to-many relationships; i.e., the Money and Cash messages of the activity diagram map to the Money common object of the reduced model, and the Product and Off-the-self product messages of the activity diagram map to the Product common object of the reduced model.

Table VII-1. Mapping of: (a) – actors of the e^3 -value model of Figure VII-2 to business units of the reduced model and (b) – business units of the reduced model to swimlanes of the activity diagram of Figure VII-3

(a)		(b)	
e^3 -value model	Reduced model	Reduced model	Activity diagram
Buyer	Buyer	Buyer	Buyer
Seller	Seller	Seller	Seller
Shipper	Shipper	Shipper	Shipper

Table VII-2. Mapping of: (a) – value objects of the e^3 -value model of Figure VII-2 to common value objects of the reduced model and (b) – common value objects of the reduced model to messages of the activity diagram of Figure VII-3

(a)		(b)	
e^3 -value model	Reduced model	Reduced model	Activity diagram
Money	Money	Money	Money Cash
Product	Product	Product	Product Off-the-shelf product
Fee	Fee	Fee	Fee
Transport	Transport		

In the following, we describe the transformation of an arbitrary e^3 -value model and activity diagram to their underlying reduced models by means of the example from Section 3. Further in Section 7, a notion of consistency based on equivalence of reduced models will be introduced.

6.2 Transformation from an e^3 -value Model to Reduced Models

This section describes a transformation from an e^3 -value model to reduced models. The transformation has three steps. The first step separates possible alternatives in a scenario. The second step selects the actors and value objects to be represented in the reduced model as units and common objects, and builds the transformation tables. The third step transforms actors and value objects to units and common objects.

Step 1: Separate Alternatives. This step deals with OR-forks in the dependency path in e^3 -value models. When we encounter an OR-fork, we duplicate the e^3 -value model in accordance to the number of alternatives in the OR-fork. In each copy of the original model, we substitute the OR-fork with a dependency connection to form a single alternative. The other alternatives remain disconnected. This transformation step generates from a single e^3 -value model potentially many e^3 -value models.

The OR-forks are treated, beginning from the start stimuli, consequently in the order of their occurrence along the dependency path. This guarantees that all possible scenarios of value exchanges are captured in individual reduced models. At the end of the step, the exchanges that are not connected in a dependency path are removed from the models. Then, the dependency paths are also removed. The final result of step 1 of the transformation is shown in Figure VII-5.

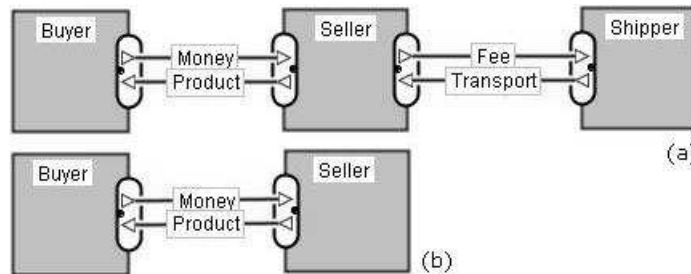


Figure VII-5. Final result of transformation Step 1: (a) – value model of the first alternative dependency path and (b) – value model of the second alternative dependency path

Step 2: Transformation Tables. This step classifies the value objects into product and experience types. Product type value objects are entitled for transformation, where the experience type value objects are removed from the e^3 -value model. As a result, actors that exchange only experience type of value objects are isolated and are, therefore, also removed from the model. In the e^3 -value model of our example, all value objects are eligible for translation.

Remaining actors and product type value objects are mapped to business units and common value objects, respectively. With regard to our example, the mappings are represented in the transformation tables Table VII-1 (a) and Table VII-2 (a).

Step 3: Generate Reduced Models. This step transforms each e^3 -value model representing an alternative into a reduced model. In particular, actors and value objects are transformed into business units and common value objects as specified in the mapping tables (see Table VII-1 (a) and Table VII-2 (a)). As a result of this

transformation the specific information on reciprocity of value exchanges and on bundling of value objects is omitted. The reduced models, derived from the e^3 -value model (see Figure VII-2), are depicted in Figure VII-6.

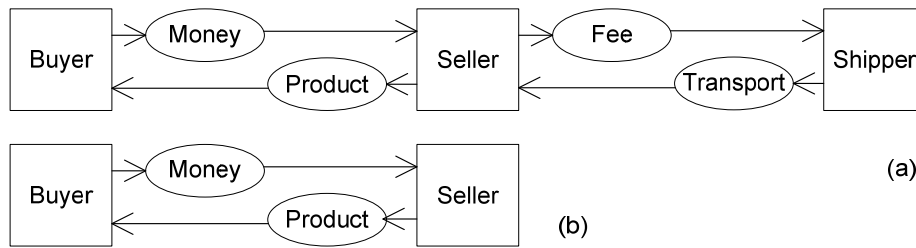


Figure VII-6. Reduced models corresponding to the e^3 -value model: (a) – first reduced model and (b) – second reduced model

6.3 Transformation from an Activity Diagram to Reduced Models

The transformation of an activity diagram to reduced models is performed in three steps. The first step resolves choices in the control flow. The second step identifies sequences of messages, marks single messages as corresponding to value exchanges in the e^3 -value-model sense, and builds transformation tables. The third step transforms swimlanes to units and messages to common objects.

Step 1: Remove Choices. This step transforms the activity diagram to a number of models which do not contain choices; i.e. the resulting models do not have conditional branches of execution flow. The transformation works in a similar way as the transformation of OR-forks in the e^3 -value model. We begin from the start stimuli and we follow the execution flow. Each time we encounter a choice, we duplicate the model and substitute the choice with a direct transition to one of the alternatives. We cut the disconnected branches from the execution tree. This transformation step generates potentially many activity diagrams from a single activity diagram. Figure VII-7 shows the result of transformation step 1 applied on the activity diagram from Figure VII-3.

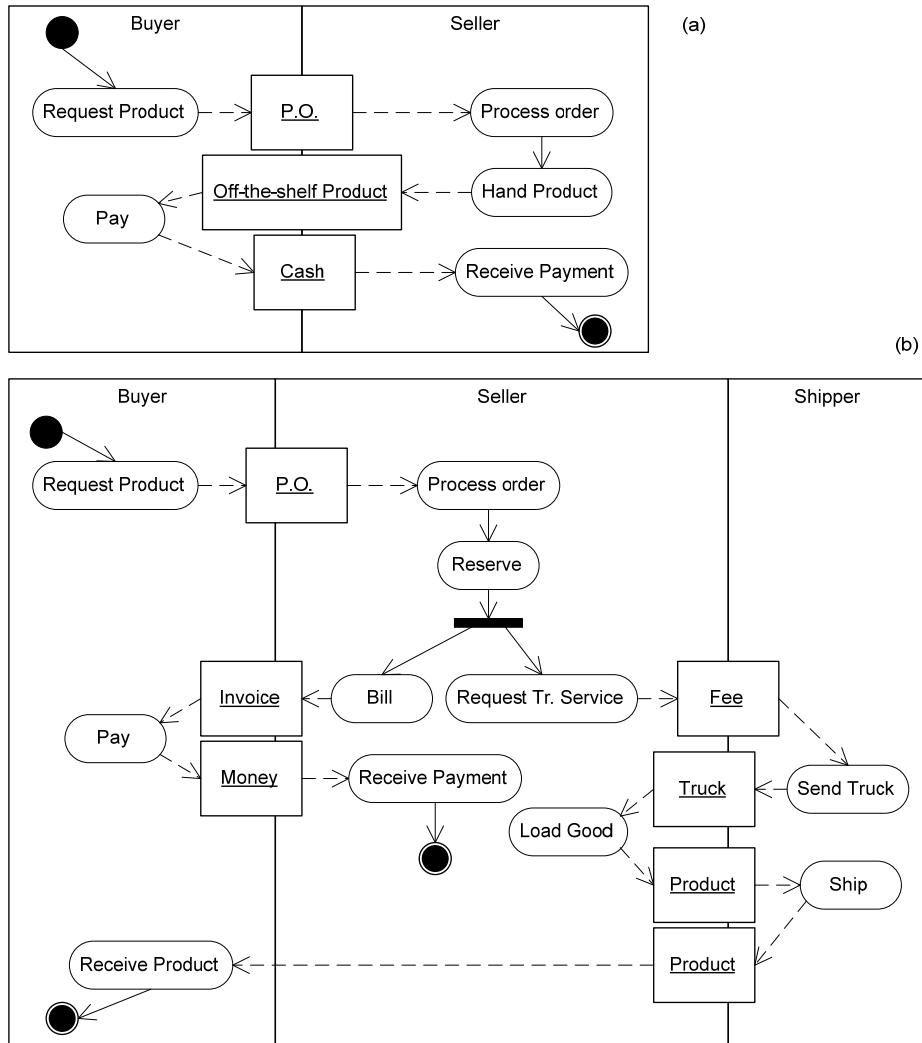


Figure VII-7. Result of transformation Step 1: (a) first activity diagram without conditional branches and (b) – second activity diagram without conditional branches

Step 2: Transformation Tables. This step identifies the flow of messages between two swimlanes that result in an exchange of a value object in the e^3 -value-model sense. Additionally in each sequence, a single message is selected to be further transformed to a common object.

The selected messages and their sending and receiving swimlanes are mapped to common value objects and business units, respectively. The mappings are represented in transformation tables which for our example are Table VII-1 (b) and Table VII-2 (b).

Step 3: Generate Reduced Models. This step transforms each activity diagram representing an alternative into a reduced model. In particular, swimlanes and messages are transformed into business units and common value objects as specified in the mapping tables Table VII-1 (b) and Table VII-2 (b). As a result of this transformation the specific information about sequence of message exchanges is omitted. The final reduced models, derived from the activity diagram (see Figure VII-3), are depicted in Figure VII-8.

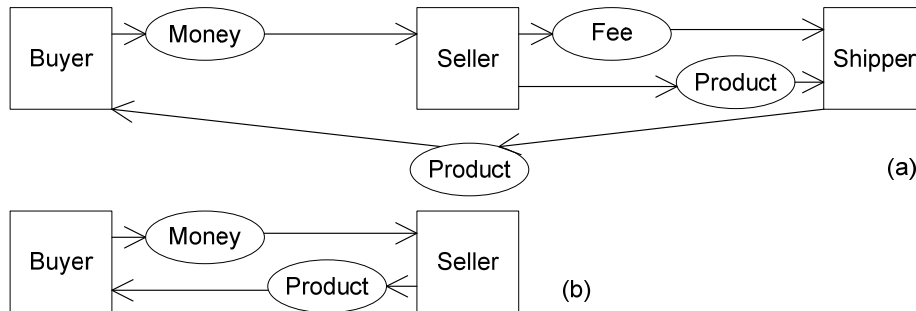


Figure VII-8. Reduced models corresponding to the activity diagram: (a) – first reduced model and (b) – second reduced model

For clarity of presentation, we refer further to a reduced model derived from an activity diagram as a reduced process model. Correspondingly, a reduced model derived from a value model is referred to as a reduced value model. The origin of a reduced model is undistinguishable from the model itself; we name them differently for explanation purposes only.

7 Equivalent Reduced Models

Two reduced models are equivalent if each contains the same common value exchanges. This means that:

- each reduced model contains the same business units;
- each reduced model contains the same common value objects;
- in each reduced model, the sending and receiving business units of a particular common value object are the same.

In our example, we can determine that the reduced models of Figure VII-6(b) and Figure VII-8(b) are equivalent. In contrast, the models of Figure VII-6(a) and Figure VII-8(a) are not because (i) the *Transport* common object is not present in the reduced process model and (ii) the *Product* common object is exchanged between different units in the two models.

7.1 Transitivity

The reduced models in Figure VII-6(a) and Figure VII-8(a) can be made equivalent by applying transitivity on the *Product* common object in the reduced process model (see Figure VII-9). Transitivity removes intermediary units from a chain of common exchanges by directly representing the common exchange between the units in the beginning and the end of the chain. Thus, the *Product* common exchanges between Seller and Shipper, and between Shipper and Buyer (see Figure VII-9) is represented as a direct *Product* common exchange between Seller and Buyer. See for illustration the left part of Figure VII-10.

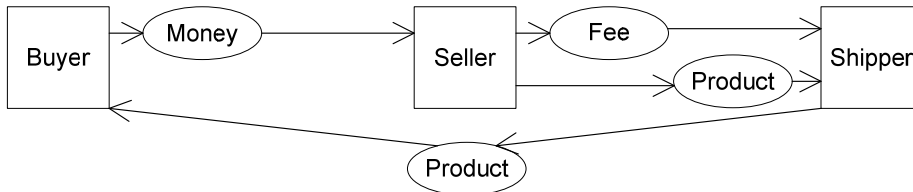


Figure VII-9. The reduced process model without an equivalent reduced value model. This figure is a copy of Figure VII-8(a)

The reason for the unit in the beginning of the chain to involve additional units must be beneficial to the unit itself. Thus, the benefit must be provided by the intermediary unit because otherwise it would not have been involved. The benefit introduced by the intermediary unit, in our case this is Shipper, to Seller is represented by an unspecified common exchange. See for illustration the right part of Figure VII-10.

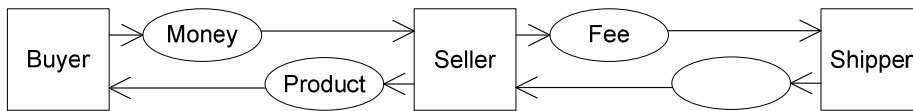


Figure VII-10. Reduced model after transitivity transformation

The unspecified common exchange can be instantiated by any reduced model common object. This introduces several additional options to be checked by the equivalence testing. With regard to the example, the instantiation with the *Transport* common object, as depicted in Figure VII-11, results in a reduced model equivalent with the reduced value model from Figure VII-6 (a).

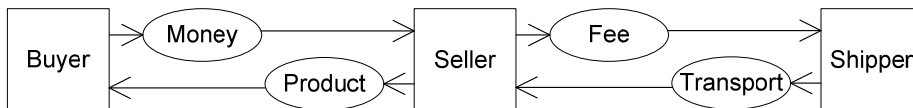


Figure VII-11. Reduced model after transitivity transformation and after instantiation of the unspecified common exchange

Due to the equivalence of the reduced models, we consider the *e³-value* model and the activity diagram to be consistent, as we informally do in Section 4.

7.2 Formal Consistency Definition

An e^3 -value model and an activity diagram are consistent if there exists at least one non-trivial mapping under which the corresponding sets of reduced models are equivalent.

A non-trivial mapping is one for which holds that:

(1) every product value exchange in the e^3 -value model is mapped to one common value exchange. This includes that (i) every product value object is represented in the reduced model and (ii) sending and receiving actors of a product value object are not mapped to a single business unit in the reduced model;

(2) every transaction in the activity diagram is mapped to one common value exchange. This includes that (i) every transaction is represented in the reduced model and (ii) sending and receiving swimlane of the message representing the transaction are not mapped to a single business unit in the reduced model.

The restrictions listed above preserve the product value exchanges in the e^3 -value model and the transactions in the activity diagram during the transformation; i.e., these are all represented in the reduced model. In case the granularity of the e^3 -value model and the activity diagram is similar, the relationships between actors and swimlanes, and between value objects and messages are usually one-to-one or one-to-many. Nevertheless, the relationships are many-to-many in the general case. The consequences of which, we discuss in the next section.

8 Plausibility of the Consistency Check

The proposed consistency check is plausible with respect to the informal consistency definition if all model pairs considered to be informally consistent are consistent with regard to our consistency definition and vice versa. To argue that this is the case, we will decompose the informal consistency definition from Section 4 and compare it with the building blocks of our consistency definition.

For the informal consistency, we make the following observations:

1. It is based on relations between separate alternative dependency paths and separate execution sequences;
2. The relation between an alternative dependency path and an execution sequence is based on a single set of product value exchanges happening in both models.

Our transformation procedures represent the original model as several reduced models, one per alternative, which is based on alternative dependency paths and execution sequences. That is, one alternative dependency path (execution sequence) results in a single reduced model. Thus, the granularity of the performed consistency check is the same as in the informal one.

The second observation says that an alternative dependency path and an execution sequence result in the same product value exchanges. Our definition of equivalent reduced models requires identical common value exchanges in the two models. This shows that both consistency definitions require a relationship between models based on *the same set of* product value exchanges and *on the same set of* common value exchanges.

As we describe in Section 5 the relationship between value exchanges and common exchanges is one where every product value exchange is represented in the reduced

model. Similarly, transactions in the activity diagram are identified as such if they result in a product value exchange. Thus in case of a non-trivial mapping, every product exchange is transformed to a common exchange.

We conclude that the proposed consistency definition is plausible with respect to the informal consistency definition.

9 Granularity of Models: Many-to-Many Relationships between Instances

In the previous section, we show how our consistency check works between models with comparable granularity. In particular, the relationships between actors and swimlanes, and value objects and messages are only one-to-one and one-to-many; where, the latter one breaks down to one-to-one relationships when the alternatives are taken separately. However, there is no guarantee that the granularity of e^3 -value models and activity diagrams is the same. Thus, the relationships between actors and swimlanes, and value objects and messages are in the general case many-to-many. In this section, we discuss when and, if so, how consistency of models with different granularity can be checked.

Our consistency definition is based on equivalence of reduced models, which requires equivalence of units and common objects. From (i) the equivalence of units, respectively common objects, and (ii) the way the mapping tables are constructed (see Section 6) follows that the semantic relationships between instances in the e^3 -value model and the activity diagram are one-to-one. To guarantee a proper result of our consistency check, we have to ensure that the checked models have similar granularity.

There are two strategies for adapting granularity: either aggregation is performed on the more fine-grained model or division of the more coarse-grained model. We have explored both approaches and discovered the following drawbacks.

The *aggregation approach* may lead to a single actor and a single swimlane. This is because, a many-to-many relationship between actors and swimlanes can result in an aggregation of two swimlanes which may trigger an aggregation of two actors and so forth. Due to the aggregation, the exchanges of product value objects between aggregated actors in the e^3 -value model are lost; the same holds for messages between swimlanes in the activity diagram. This loss of information makes the consistency decision less precise.

The *division approach*, on the other hand, may lead to the finest-grained granularity in both models, where a single actor exchanges a single value object or a single swimlane sends a single message. This is again a loss of information comparable to the aggregation case. In particular, the relation information between different value objects, correspondingly messages, is lost. Thus to limit the loss of information, we can constrain ourselves to division of only value objects and swimlanes or only actors and messages. From the two, we select the second because actors are intuitively more coarse-grained than swimlanes and a single message may represent more than a single economic value.

Below, we illustrate how granularity of models is equalized by division of actors and messages. By means of an example, we show how we resolve one-to-many relationships.

9.1 Example

We consider a business where a client is interested in the mortgage and insurance products of a bank. Figure VII-12 represents the e^3 -value model of the example. The client (to the left in the figure) is interested in exchanging monthly fees for a period of time in case it gets a loan in a form of a mortgage. Additionally to that for some economic reasons, the client is interested in insurance from the same bank. The dependency path includes an AND-fork, shown within the Client actor, which denotes that the client wants both products.

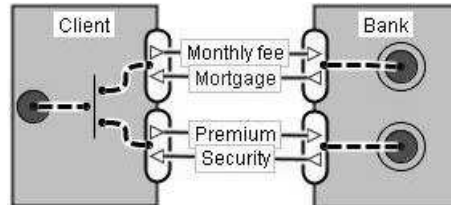


Figure VII-12. Value model of the example

Figure VII-13 represents an activity diagram for the above example case, where the bank is modelled as two swimlanes representing its Mortgage and Insurance departments separately. The client (the first swimlane from left to right) requests simultaneously a mortgage and insurance. The handling of the requests is performed in a similar way: e.g., the mortgage request (in the top of the figure) is processed by the Mortgage swimlane which after processing the request grants mortgage to the client. Once given, the insurance and the mortgage are utilized by the client, which is shown in the client swimlane as a Consume activity. The bottom of Figure VII-13 shows the monthly payments performed by the client: two separate payments to the Mortgage and Insurance departments are represented by the Monthly fee and Premium messages.

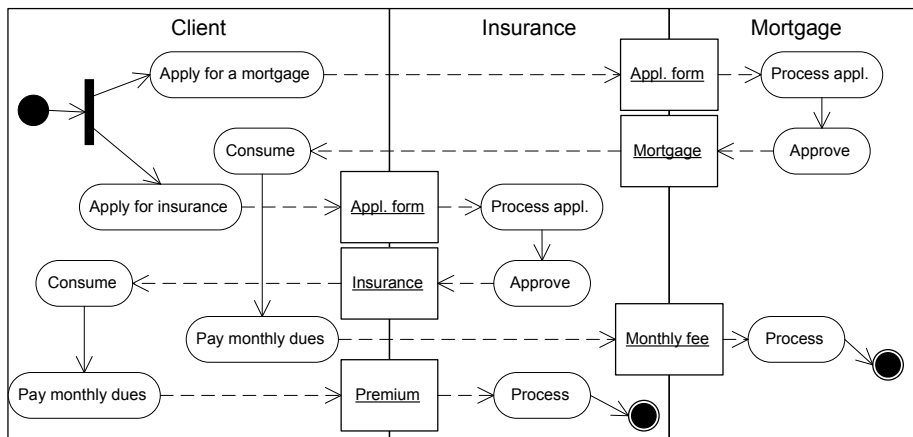


Figure VII-13. Process model of the example with two swimlanes representing the bank

9.2 Type of Relationships

We distinguish two types of relationships: *individual* and *aggregation*. An individual one-to-many relationship means that one entity is mapped to several independent entities each of which represents the entity as a whole. This is the type of relationship in the first example in Section 5, where **Money** maps to **Money** and **Cash**. An aggregation one-to-many relationship means that one entity is mapped to several independent entities which together represent the entity as a whole. This is the type of relationship in the second example above, where **Bank** maps to **Insurance** and **Mortgage**.

The individual type of relationship is not relevant for this section because the actors and swimlanes or objects and messages are of the same granularity. Further in this section, we assume always an aggregation relationship.

9.3 Splitting of Actors

The models in Figure VII-12 and Figure VII-13 differ in granularity: the bank from the e^3 -value model is represented as two individual swimlanes in the activity diagram. To resolve the one-to-two relationship between actor and swimlanes, we split the bank actor in the e^3 -value model. The newly appeared actors, named **BankI** and **BankM**, need to distribute the value exchanges of the original actor **Bank**. In our example case, there are four exchanges and we generate all possible combinations with the new actors. Figure VII-14 shows these. We check consistency with all combinations.

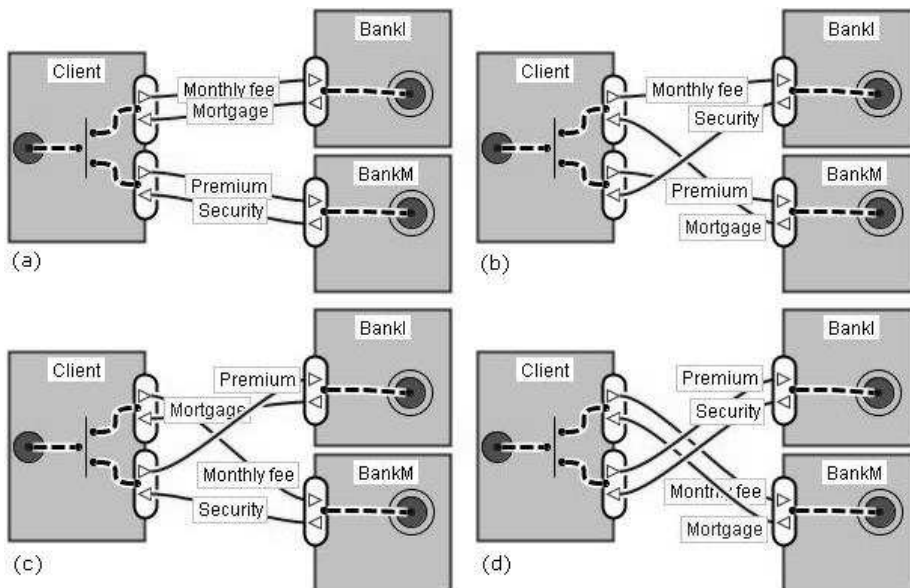


Figure VII-14. All combinations of value object exchanges after the splitting of **Bank** actor: (a) – combination one, (b) – combination two, (c) – combination three, and (d) – combination four

The choice of splitting the **Bank** actor is derived from the mapping tables, where the one-to-many relationship is observed. (The particular mapping table is not shown.) We

split an actor by splitting the corresponding unit in the reduced model; therefore, all combinations of exchanges of value objects have to be considered.

Based on the splitting of the units in the value reduced model it turns out that the reduced models are equivalent, which fits to the informal consistency. Depending on the particular mapping of actors and swimlanes, model Figure VII-14 (a) and (d) result in equivalent reduced models with the reduced model of Figure VII-13.

9.4 Splitting of Messages

Figure VII-15 shows a second activity diagram for the example case. The diagram differs in two points from Figure VII-13. First, the bank is represented as one swimlane and correspondingly the activities and messages belonging to the Mortgage and Insurance swimlane are in the Bank swimlane. The second difference is in the way payment of monthly dues is modelled. In the bottom of Figure VII-15, payment by the client is represented as a single message.

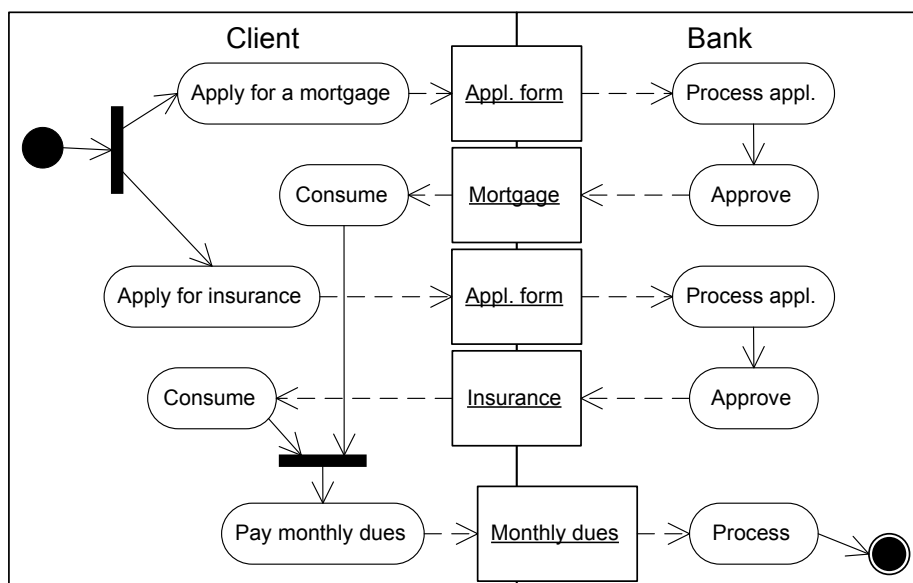


Figure VII-15. Process model of the example with one message representing the monthly dues

The models in Figure VII-12 and Figure VII-15 differ in granularity: the monthly payments are represented as two distinctive value objects in the e^3 -value model while in the activity diagram they are modelled as one message. To resolve the two-to-one relationship between value objects and a message, we split the Monthly dues message into IMonthly dues and MMonthly dues. The new messages share the same sender and receiver as the original message. Figure VII-16 illustrates this at the bottom.

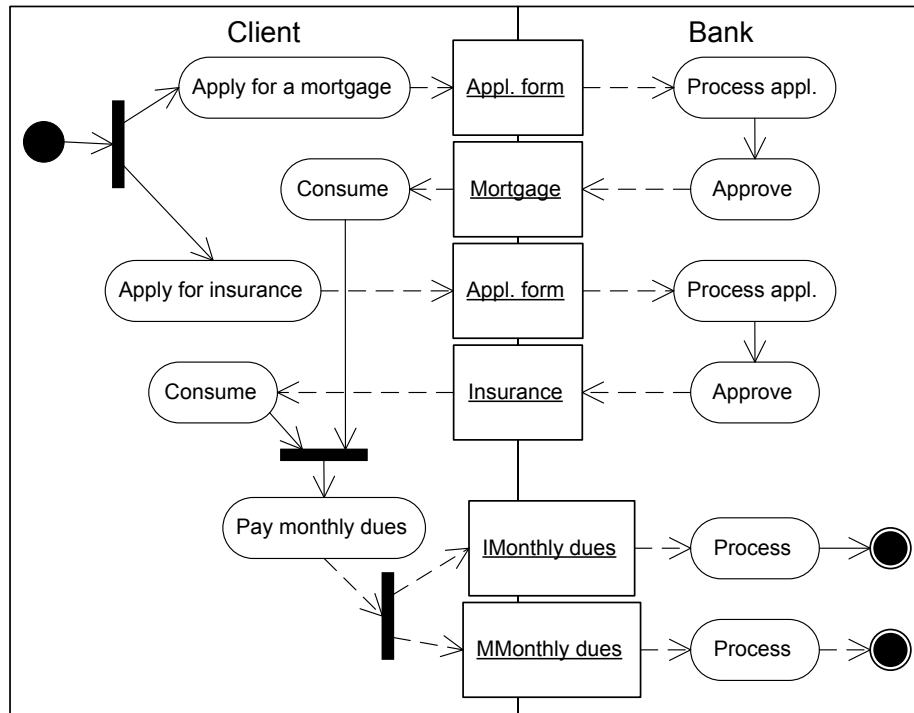


Figure VII-16. Process model after the splitting of Monthly dues message

The choice of splitting the `Monthly dues` message is derived from the mapping tables, where the one-to-many relationship is observed. (The particular mapping table is not shown.) We split a message by splitting the corresponding common object in the reduced model.

Based on the splitting of the common objects in the process reduced model it turns out that the reduced models are equivalent, which fits to the informal consistency.

9.5 Many-to-many Relationships

We analyzed a number of examples to justify our approach of splitting actors and messages. We classified these based on:

- Cardinality of the relationship, where we consider one-to-many and many-to-many relationships;
- Direction of the relationship, where we consider one instance in the e^3 -value model related to many instances in the activity diagram and vice versa;
- Arguments of the relationship, where we considered relationships between actors and swimlanes, and value objects and messages.

Our analysis shows that it is possible to adapt the granularity of models applying the approach of splitting actors and messages. Namely, it is possible to reduce the one-to-many and many-to-many relationships to one-to-one relationships in all cases except one

where we have one-to-many relationship between a swimlane and actors. Although such a relationship is possible, we think it rarely occurs because, intuitively, an e^3 -value model is at a higher level of granularity than an activity diagram.

Up to this point, we have discussed many-to-many relationships between actors and swimlanes independently of many-to-many relationships between value objects and messages. The splitting of a many-to-many relationship between actors and swimlanes does not affect the splitting of many-to-many relationships between value objects and messages. Nevertheless, certain dependencies may occur when the actor to be split exchanges a value object that maps to a message to be split. We consider all combinations.

10 Related Work

Consistency is checked in several ways. An approach is *syntactic translation* (also called direct translation) [9]. It is based on mapping of concepts and relations of the modelling notations. Then, one model is translated to the modelling language of the other. This means that the instances of concepts and relations in one notation are translated into instances of the corresponding concepts and relation in the second notation. This allows checking for inconsistency between two models in different notations by checking for inconsistency in two models in the same notation. This approach requires a consistency check procedure to be available in one of the notations. It is applicable only to bilateral consistency checks.

Another approach is *common semantic model* (also called canonical representation) [10]. It selects a single modelling notation (not necessarily one already in use) and transforms all models into that notation. The transformation works the same way as in the syntactic translation approach above: the instances of concepts and relations in the two notations are translated into instances of the corresponding concepts and relations in the common notation. The inconsistencies are checked in the common model, which require definition of a consistency check procedure for the common model. The approach is applicable to consistency check among more than two specifications.

A third approach is *meta-representation* [54, 55]. It maps modelling and meta-modelling concepts from one modelling notation to another and does not require transformation between models. Without transformation of instances, the approach is able to check only if the constraints on relations between concepts in one model hold in another. The approach does not need to define a transformation procedure and it is applicable to consistency check among more than two notations.

Our approach draws on the common semantic model approach. We define a common semantic model, which we call reduced model, in a pair-wise fashion. This gives us richer reduced models compared to a single reduced model for all perspectives. Additionally, our approach introduces a consistency check based on alternatives; i.e. models are decomposed into smaller models and checked individually for consistency. This provides a consistency check that matches with the informal consistency definition.

Our work is an extension of Gordijn's [27] requirement engineering approach to innovative e-commerce ideas. He specifies a method for exploration of business opportunities based on the distribution of value in business networks. Additionally to the

value perspective, the approach includes two more: a business process and an information systems viewpoints. The three viewpoints match closely with our perspectives. However, we explicitly check for consistency as we assume independent development of models; whereas Gordijn's approach is based on a common set of scenarios represented in each model.

Our approach requires a semantic mapping between concepts in the value and process models. The work of Gordijn, Akkermans and Vliet [26] elaborates on the differences between business and process modelling by showing semantic differences between concepts. We use this information to define our transformation tables. While Gordijn, Akkermans, and Vliet specify differences between concepts of the different models, our approach specifies semantic relationships between instances.

Wieringa and Gordijn [62] define a correctness relationship between an e^3 -value model and a process model. We use this to define our informal consistency. In addition, we provide an operationalization of this informal consistency definition based on transformations to reduced models.

The work of Dijkman et al. [15] is also based on the common semantic model approach. It relates perspectives by means of a basic perspective which contains pre-defined concepts and relations. Every perspective from a design framework need to be mapped to basic concepts and relations from the basic perspective. Our approach differs in the way how the reduced model (the basic perspective in Dijkman et al.'s terms) is defined. We do not require a pre-defined reduced model with abstract basic constructs, but we determine the reduced model after the modelling notations are selected. This allows defining richer reduced models in terms of common concepts and relations.

Consistency of a workflow model can usually be defined based on the set of potential execution sequences, a straightforward approach to check consistency is on a single workflow model. This approach has been applied to several workflow models, such as for example by van der Aalst and Weske [1] to Workflow Nets (WF-Nets), by Fu et.al. to guarded Finite State Automata [21], by Yi and Kochut [71] to Coloured Place/Transition Nets, or by Wodtke and Weikum [66] to statecharts. In either case it is checked whether the execution of the workflow results in a deadlock, that is, no further action is possible although a final state has not been reached yet. However, there exists also approaches on checking consistency between several workflows represented in the same modelling approach, such as e.g. [1, 68, 35]. In our work, consistency between different modelling approaches is defined.

11 Summary

This chapter answered our research question **Q2.1.1: How to check consistency between value and process models?** It defines consistency between an e^3 -value model and an Activity diagram. It provides an operationalization of the consistency check by defining a reduced model that contains the common concepts from the two notations. Further with the help of mapping tables, the e^3 -value model and the activity diagram are transformed to reduced models. Finally, the equivalence of reduced models is checked as a criterion for consistency. We argue that the consistency definition through equivalence of reduced models is valid with respect to a beforehand-defined informal consistency.

Chapter VIII

Validation of the Design Framework

Application of the Design Method and Evaluation of Exhibited Properties

This chapter provides arguments in support of the usefulness of the proposed design framework. It starts with a list of desired properties that our design method should exhibit. For each property, a falsifiable claim is provided: if a claim holds then the property is present. The validation of the framework is demonstrated with a real-life example to which the method is applied. After a detailed description of the development steps, the chapter concludes with an analysis of the validation claims and the method properties.

1 Introduction

The main contributions of this thesis are the two libraries of design patterns, and the design framework and method to reuse design patterns. In this chapter, we validate them by demonstrating the usefulness of the design method, where the method uses the framework and libraries.

The validation of a method for system specification by means of a comparison with another method is a difficult task. Such a validation would require the existence of a reference method¹. In the particular research project, it is not feasible to make a comparative evaluation of the proposed method. The first reason for this is that the existing methods serve different purposes and, consequently, there are no commonly agreed criteria for comparison. Second, none of the existing methods is universally accepted as a reference method or considered as a standard approach to system development. Finally, there is no commonly-agreed upon, complete and minimal specification of a particular existing system that we compare the resulting specifications against.

Because of the above reasons, we limit our validation to an existence proof. We demonstrate that the framework exists, the design patterns are reused, and the method generates valuable system specifications. Moreover, we show that the method exhibits certain properties. The achievement of these properties is the purpose for developing the method. Thus, we claim that *our method is a valid method if it achieves its purpose*.

¹ By a reference method, we mean a thoroughly investigated method, in which exact and clear descriptions of the development steps are given for the accurate determination of one or more design decisions; the documented accuracy and precision of the method are commensurate with the method's use for assessing the accuracy of other methods for measuring the same property values (adapted from [23])

By *property*, we mean qualities of the method that are desired and make the method distinct. Examples are: (1) develops a consistent set of models and (2) provides an explicit justification of design decisions (see for more Section 1.1.)

By *claim*, we mean an assertion which, if true, is the evidence that the method has a certain property.

By *argument*, we mean a firmly established fact or an intuition which supports our belief that a certain claim is true.

1.1 Properties of the Design Method

The properties of our method below are derived from Chapter III Design Framework for e-Business Models, Section 5 Objective and Available Resources, on page 35. A method that has these properties is considered to achieve the purpose of efficiently and with explicit design decisions, develop a consistent, high-quality system specification.

Method Property 1 (MP1): The Method Develops a Consistent Specification. This is a property of the result of the method. It includes a functional and a non-functional aspect. The former is the fact that it is possible to develop a set of models each of which in a different perspective of the framework. The latter is concerned with the qualitative aspect of the specifications: in particular, the fact that the set of models is consistent.

Method Property 2 (MP2): The Method Is Effective. This is a property of the method. It holds if better quality of the specifications is achieved. In comparison to a method that investigates a limited number of design alternatives on the basis of heuristics or rules of thumb, our method considers more alternatives. This leads to a result with better quality. From the qualitative statement about better quality of the result of the method, we derive a qualitative property of the design process, namely the design process is effective.

Method Property 3 (MP3): The Method Is Efficient. This is a qualitative property of the method. The property holds if less effort is required during design in comparison to a method that does not reuse existing design knowledge. Our method requires less knowledge and time because it reused design patterns.

Method Property 4 (MP4): The Method Provides Explicit Justification of Design Decisions. This is a qualitative property of the design process. It states that the process is able to explain why certain decisions are taken and which parts of the design are affected by these decisions.

1.2 The Claims

To validate that the method exhibits the four properties listed in the previous section, we make a number of claims and we give arguments to believe that these claims hold true. Our claims are:

Claim 1 (C1): Correct Models Can Be Developed. This claim is in support of property MP1. It demonstrates that the method produces models that correct representation of the reality. We verify claim C1 by asking experts in the domain of the particular modelling

notation and in the domain of the particular system under development to review the final specification. If the experts agree that every single model is a correct representation of the intended system then the claim holds.

Claim 2 (C2): Consistent Specification Can Be Developed. This claim is also in support of property MP1. It demonstrates that the produced models are consistent. We verify claim C2 by asking the same experts as in claim C1 to review the final specifications. Whereas, in claim C1, the experts examine the models independently of each other, in claim C2, the experts examine the specification together. If the experts agree that the specification is consistent¹ then the claim holds.

Claim 3 (C3): The Exploration of the Design Space Is Extensive and Automated. This claim is in support of property MP2. It demonstrates that the method considers more alternative design choices than a method which investigates a limited number of design alternatives and in which subsequent design decisions depend on the path of previously taken decisions. Claim C3 holds if the method checks on every step every viable alternative rather than adding new design fragments relying that the composition is still the best design. The process of exploration is automatic in the repetitive steps.

Claim 4 (C4): Design Knowledge Is Reused. This claim is in support of property MP3. It demonstrates that the method reuses design knowledge in a form of patterns. Including available partial solutions reduces the effort of development. Claim C4 holds if patterns are (re)used in the process of development.

Claim 5 (C5): The Library Is Always the First Source of Design Fragments. This claim is in support of property MP3. It demonstrates that the non-automatically generated fragments in the final specification are not present in the library. Claim C5 holds if solutions to system requirements are first searched for in the library of patterns.

Claim 6 (C6): Fragments in the Final Specification Can Be Traced Back to Requirements. This claim is in support of property MP4. It demonstrates that every automatically generated fragment can be traced back to a requirement. Claim C6 holds if there is a sequence of design decisions that links the fragments in the final specification with the initial requirements.

Claim 7 (C7): Design Decisions Are Explicit. This claim is in support of property MP4. It demonstrates that every design decision is explicitly justified; i.e., every decision is motivated with objective arguments and can be explained. Claim C7 holds if there is a procedure for making decisions and a concrete reject or accept decision for every design pattern.

Table VIII-1 presents the correspondence between properties and claims.

Table VIII-1. Correspondence between properties and claims

<i>Method Property</i>	<i>Claim</i>
------------------------	--------------

¹ A set of models is consistent if it does not contain contradicting statements that make the system under development impossible to build (Chapter VII Consistency between Value and Process Models)

MP1: the method develops a consistent specification	C1: correct models can be developed
	C2: consistent specification can be developed
MP2: the method is effective	C3: the exploration of the design space is extensive and automated
MP3: the method is efficient	C4: design knowledge is reused
	C5: the library is the first source of design fragments
MP4: the method provides explicit justification of design decisions	C6: fragments in the final specification can be traced back to requirements
	C7: design decisions are explicit

1.3 Validation Methodology

For a demonstration of our validity arguments, we select a non-trivial real-life example. We apply the design method and the result is the required existence proof of usability of the method.

2 Real-life Example: Clearing Rights to Make Music Content Public

We validate our method with a real-life example from the domain of making music content public. This involves a number of businesses, societies, and individuals. In particular, the business network consists of the following main participants: music users (e.g.: radio station, bar, disco, shop, etc.), which broadcast music content with intellectual property rights; rights owners (singers, text writers, associated composers, producers, etc.), who create music content; and rights societies, which intermediates the transfer of rights to make music content public from rights owners to music user. The example includes other entities such as advertising companies, banks and radio listeners, branch organizations.

Despite the variety of businesses forming the network, we focus on a particular type of music users, namely a commercial radio station, and we focus on a rights society managing the rights to make music content public. The prototypes of the business actors in the example are the Dutch rights society SENA (www.sena.nl) and radio stations such as Radio 538 (www.radio538.nl) or Slam FM (www.slamfm.nl). Below, we describe the goals of a top radio station and a typical rights society.

2.1 Goals and Goal Model of Radio Station

2.1.1 Goals of Radio Station

Music users are businesses that make music public: i.e. they play music to a number of listeners because they have some commercial interest of doing that. Examples are: radio stations, bars, discos, and shops. In our real-life example, we restrict our-selves to radio stations. Further, we use the terms music user, radio, and radio station interchangeably.

We mean any commercial radio station which does not require subscription from its listeners but makes money by broadcasting commercials.

A commercial radio station has one main goal: to make money. Thus, we start our analysis from this goal:

- **G1: Maximize profitability.**

We measure goal **G1** with the net profit of the radio per, e.g., year. The net profit is measured in euros and the goal is considered achieved if the money is above a certain value.

We operationalize goal **G1** with:

- **v₁: Net profit per year [Euro].**

Variable v_1 is the amount of money left in the radio after the costs to operate the radio are subtracted from the revenue.

To maximize the net profit, a commercial company needs to maximize its revenue and to minimize its operation costs. From this relationship follows the decomposition of the main goal in three new goals. The first two goals concern the maximization of revenue; the third goal minimizes the operation costs.

Following the line of maximization of revenue, a radio station sells time slots during which advertising companies broadcast their commercials. There are two ways to increase the income flow from selling slots: (1) sell more slots and (2) sell slots at a higher price. The goals that reflect that are goals **G2** and **G5**, respectively. Thus, the next goal in our model is:

- **G2: Maximize broadcasted commercials.**

We measure goal **G2** with the average number of commercial breaks in the radio program per hour. We assume that a break for commercial lasts on average 4 minutes and that a single commercial is 30 seconds long. Assuming this, we can calculate that every timeslot fits 8 commercials.

We operationalize goal **G2** with:

- **v₂: Breaks for commercials per hour [Natural numbers plus 0].**

Hence, the number of commercials per hour is 8 times the number breaks.

Commercials, in general, do not attract listeners. People listen to a radio station because they are interested in the content of the radio program. Therefore to succeed in broadcasting commercials to its audience, a radio has to intertwine the commercials within its program. Goal **G2** is decomposed in two goals, the first one of which is:

- **G3: Bundle commercials with music content.**

We measure goal **G3** with the presence or absence of commercials within the regular program of the radio station.

We operationalize goal **G3** with:

- **v₃: Bundled product and advertisement [Boolean].**

People listen to a radio for various reasons. One of the major ones is that they want to be entertained. Again, many things can be entertaining but listening to music is very often the first choice. Here, we restrict ourselves to the so-called music channels; i.e., radio stations that offer exclusively music content. Consequently, the only thing that the listeners are interested in is the songs broadcasted by the radio station.

- **G4: Play music to entertain.**

Entertaining the audience by playing songs can be done in various ways. In our real-life example, we focus on the presentation format of the songs. From all the properties of a particular format of a radio program, we focus on the sequence of songs uninterrupted by

a break for commercials. For example, a '8-songs-non-stop' format means that 8 songs are broadcasted in a sequence without a commercial in-between. We measure goal **G4** with the number of songs that fit in the particular format.

We operationalize goal **G4** with:

- **v₄: Songs per hour [Real numbers].**

The link between the format of a radio program and the number of songs played in an hour is as follows. It is assumed that the average length of a song in minutes is 4 and a break for commercials lasts 4 minutes. Thus, the value that v_4 can take are calculated with $(4*v_4 + 1)*k = 60$, where $k = 1..15$. The values are real numbers but this does not mean that only part of a song is played, although this may be a valid format.

The second goal that maximizes the cash-in is:

- **G5: Maximize the price per commercial.**

We measure goal **G5** with the price per second broadcast time. Following the assumption we made above that the duration of an average commercial is 30 seconds, we calculate that the price for a commercial is 30 times the price for 1 second broadcast time. We assume that the time slots are equally priced within the 24-hour interval.

We operationalize goal **G5** with:

- **v₅: Price per second of broadcast time [Euro].**

The price per second of broadcast time depends of the number of listeners. Advertising companies are willing to pay more money if their message is heard by more people. In order to increase the price of broadcast time, the radio station must attract more listeners. Thus, the next goal is:

- **G6: Maximize number of listeners.**

We measure goal **G6** with the average number of listeners per hour. For simplicity, we assume that the number of listeners remains constant within the 24-hour interval.

We operationalize goal **G6** with:

- **v₆: Listeners per hour [Natural numbers plus 0].**

Variable v_6 is the average number of listeners per hour calculated over a period of time, e.g. a day, a week or a month. The number is assumed invariant of the hour of the day; i.e. audience remains the same throughout the day and the night, as said earlier.

For the reason that we consider only the format of the presentation of the music content as relevant for the number of listeners, a goal that influences the attracting of listeners is goal **G4 (G4: Play music to entertain.)** Additionally, goal **G2 (G2: Maximize broadcasted commercials)** negatively influences the attraction of listeners because of the general assumption that listeners do not like commercials.

The third goal (see above) that determines the satisfaction of goal **G1** minimizes operation costs. Thus, we specify goal **G7** as:

- **G7: Minimize operation costs.**

We measure goal **G7** with the money spent per year to operate the radio station. This includes, e.g., the rights to broadcast a song, the costs to broadcast a song, but also one-time costs.

We operationalize goal **G7** with:

- **v₇: Costs per year [Euro].**

Variable v_7 is the total amount of money spent to run a radio station for one year.

2.1.2 Goal Model of Radio Station Music User

The goals of the radio station, listed in the previous subsection, and their relations are shown in Figure VIII-1. This is the goal model of a commercial radio station.

The goals in the goal model are denoted only with their abbreviation and number, corresponding to the goal descriptions in the previous subsection. To avoid further name conflicts as other goals may be abbreviated in a similar way, we uniquely identify the goals by prefixing their names with a code that refers to the goal model to which the goals belong. The code for the commercial radio station goal model is **RAD**. Thus, the full name of goal, e.g., **G1** is **RAD.G1**.

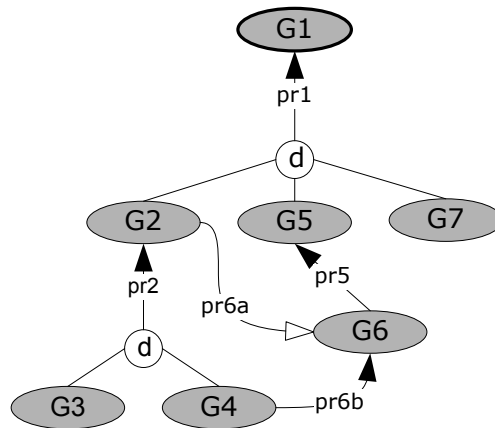


Figure VIII-1. Goal model of a commercial radio station

Table VIII-2 summarizes the goals of a commercial radio station and their operationalization.

Table VIII-2. Goals of a commercial radio station and their operationalization

Goal code	Goal	
Variable code	Variable	Variable domain
G1	Maximize profitability	
v ₁	Net profit per year	Euro currency
G2	Maximize broadcasted commercials	
v ₂	Breaks for commercials per hour	Natural numbers plus 0
G3	Bundle commercials with music content	
v ₃	Bundled product and advertisement	Boolean
G4	Play music to entertain	
v ₄	Songs per hour	Non-negative real numbers
G5	Maximize the price per commercial	
v ₅	Price per second of broadcast time	Euro currency
G6	Maximize number of listeners	
v ₆	Listeners per hour	Natural numbers plus 0
G7	Minimize operation costs	

v ₇	Costs per year	Euro currency
----------------	----------------	---------------

2.1.3 Propagation Functions

The goal model of a commercial radio station in Figure VIII-1 contains 7 goals which are related among each other with 4 goal relationships, 2 decomposition relationship and 2 dependency relationships. Each of the relationships is associated with a propagation function which captures the effect goals are causing one another. The number of propagation functions equals the number of head-goals: in the particular case, this is 4.

All propagation functions are a common sense approximation of what a real-life relation between the variables could be. The assumptions for the shape of the functions come in addition to the simplifications made while identifying the goals of a commercial radio station. The assumed propagation functions demonstrate the use of our method; the introduced imprecision does not invalidate the method.

The first propagation function **pr1** is associated with the decomposition relationship between goal **G1** (the head of the relationship) and goals **G2**, **G5**, and **G7** (the tails of the relationship.) Propagation function **pr1** calculates the value of goal variable **v₁** out of the values of goal variables **v₂**, **v₅**, and **v₇**. In other words, the function calculates the net profit per year out of the breaks for commercials per hour, the price per second of broadcast time, and the costs per year. The net profit is measured per year; therefore, the breaks for commercials have to be converted to the same scale. That means converting from hours to years, which translates to multiplying by 24 (number of hours per day) and then by 365 (number of days per year). A second conversion is needed to equalize the scale of price per second and duration of break for commercials: the price per minute is 60 times the price for second.

The conversions allow us to calculate the revenue per year. It is the number of breaks for commercials per year, multiplied by 4 min per break, and multiplied by the price of a broadcast minute. Finally to calculate the net profit, we need to subtract the costs per year from the revenue. Thus, the propagation function is:

$$\bullet \quad v_1 = \text{pr1}(v_2, v_5, v_7) = 24 * 365 * v_2 * 4 * v_5 * 60 - v_7.$$

The second propagation function **pr2** is associated with the decomposition relationship between goal **G2** (the head of the relationship) and goals **G3** and **G4** (the tails of the relationship.) Propagation function **pr2** calculates the value of goal variable **v₂** out of the values of goal variables **v₃** and **v₄**. In other words, the function calculates the number of breaks for commercials per hour out of the fact that music content and commercials are bundled and the number of songs per hour. Due to the fact that commercials and songs are bundled together, more songs results in less commercials. The following constrain holds: the breaks per hour plus the songs together multiplied by 4 minutes must equal 60 minutes $(v_2 + v_4) * 4 = 60$.

Thus, the propagation function is:

$$\bullet \quad v_2 = \text{pr2}(v_3, v_4) = v_3 * (60 - 4 * v_4) / 4.$$

The third propagation function **pr5** is associated with the dependency relationship between goal **G5** (the head of the relationship) and goal **G6** (the tail of the relationship.) Propagation function **pr5** calculates the value of goal variable **v₅** out of the value of goal variable **v₆**. In other words, the function calculates the price per second of broadcast time

out of the number of listeners per hour. The general trend is that more listeners are of greater value to advertisers, which leads to high prices.

We assume, the marketing department forecasts the following dependency between price and listeners, which is also the propagation function:

- $v_5 = \text{pr5}(v_6) = (\text{sqrt}(v_6/8) - 10)/(\text{sqrt}(20) - 1)$.

Two points of this function are:

- **800 listeners lead to 0 euros per second.**
- **1 020 800 listeners lead to 100 euros per second.**

The last propagation function pr6 is associated with the dependency relationship between goal **G6** (the head of the relationship) and goals **G2** and **G4** (the tails of the relationship.) Propagation function pr6^1 calculates the value of goal variable v_6 out of the values of goal variables v_2 and v_4 . In other words, the function calculates the listeners per hour out of the breaks for commercials per hour and the songs per hour. The two variables have opposite effect: the more the songs the more the listeners, but the more the breaks the less the listeners. Due to the particular simplifications, variables v_2 and v_5 are not independent: both measure time intervals in one hour. The following constrain holds between them: the breaks per hour plus the songs together multiplied by 4 minutes must equal 60 minutes $(v_2 + v_4)*4 = 60$.

The propagation function as a function of two and, respectively, one variable is:

- $v_6 = \text{pr6}(v_2, v_4) = 20*v_2^2 - 600*v_2 + 4000*\text{sqrt}(v_4) + 4500$ and
- $v_6 = \text{pr6}(v_4) = 20*v_4^2 + 4000*\text{sqrt}(v_4)$.

Two points of this function are:

- **14 songs (1 break) to 18 887 listeners.**
- **9 songs (6 break) to 13 620 listeners.**

2.1.4 Anticipated Values for Goals in the Goal Model of a Radio Station

The goal model of a radio station includes goals that:

1. are already achieved,
2. capture a known problem with a ready to reuse solution, and
3. represent an innovation with only a vague idea for implementation.

The first type does not need to be matched with patterns. In this case, the values for the goal variables do not come from matching capabilities but they are set in the model itself. The second type of goals receives their value from matching capabilities; whereas, the third type needs an estimation of the variable value as the goal does not have a known solution.

To enable the comparison of old and new designs and the evaluation of partial designs, we have to consider the satisfied goals and assume values for the goals without a match. This results in assigning values to goal variables which we called anticipated values. The anticipated values are part of the goal model.

In particular for the goal model of a radio station, we have anticipated to achieve values for two goals: goal **G4** and goal **G7**. Variable v_4 can take value from 0 to 15, where: 0 would mean no songs, only commercials and 15 would be only songs. We take a common sense value of 12. The estimation of the costs per year to operate a radio station, variable v_7 , are roughly estimated to 20 000 000 euros based on a common sense assumption about number of people and expenses. Table VIII-3 summarizes these.

¹ In Figure VIII-1, propagation function pr6 is denoted with pr6a and pr6b .

Table VIII-3. Anticipated values for goals in the goal model of a commercial radio station

<i>Capability code</i>		<i>Goal</i>	
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Anticipated value</i>
G4	Play music to entertain		
v ₄	Songs per hour	Non-negative real numbers	12
G7	Minimize operation costs		
v ₇	Costs per year	Euro currency	20 000 000

2.2 Goals and Goal Model of Rights Society

2.2.1 Goals of Rights Society

Rights societies are non-for-profit organizations that manage intellectual property rights on content produced by various artists. In our real-life example, the content is a song; the artists are the people creating the song, e.g. text writers, composers, singers, and producers. Right societies collect permission fees from users of content: i.e., every person or business that make music content public has to pay permission fees to a rights society.

Our prototype of a rights society is SENA (www.sena.nl), a Dutch society that collects performance fees for both performing artists¹ and producers². We believe that SENA is a representative society and its goals can be easily generalized to any other society that collects fees for music performance.

The three main goals of a rights society are to sign an agreement with all music users, to pay the collected permission fees to the right rights owners, and to be efficient³. Below, we elaborate on the three main goals and analyze each one of them.

The first main goal is:

- **G1: Maximize the number of charged music users.**

We measure goal **G1** with the collected permission fees from music users. Counting directly number of music users is less informative because of big difference in audiences. The permission fees are measured in euro and the goal is considered achieved if the collected money is above a certain value.

We operationalize goal **G1** with:

- **v₁: Collected permission fees [Euro/Year].**

Variable v₁ is the amount of money paid to the rights society by the various music users.

The variety of music users is large. As in the case of analysis of the goals of a music user (see Section 2.1), we restrict our-selves to commercial radio stations. Therefore, goal **G1** is decomposed into two sub-goals. The first one focuses on the collection of permission fees from radio stations; whereas, the second covers the remaining types of music users, such as bars, discos, shops. Thus:

- **G2: Maximize contracted radio stations.**

¹ A performing artist is everyone who makes a creative contribution to the performance which was recorded. His or her personal talents, experience and ideas make this performance unlike any other.

² A producer is the natural or legal person who creates the basic conditions for the recording. The producer takes care of the organization, bears the financial responsibility, etc.

³ By efficient we mean that the rights society must keep operation costs minimal.

One way to measure goal **G2** is by measuring the number of radio stations with contract with the rights society. Nevertheless, the interpretation of this measure is hindered by differences in the audience of radio stations. To offset this drawback, we measure the number of rights, bought by a radio station, to play a song to a single listener. The operationalization of goal **G2** is:

- **v₂: Rights to play a song to a listener per year [Natural numbers plus 0].**

In case a radio station has potentially an audience of 1000 listener and it plays 10 songs per hour then the radio station must buy $1000 \cdot 10 \cdot 24 = 240\,000$ rights to legitimately broadcast music content one day. Such operationalization allows us to better measure the satisfaction of goal **G2**. Usually, a few radio stations cover the most of the radio audience. Thus, a comparison between the sold rights and the potential number of listeners is more informative than a comparison between the number of contracted and of all operating radio stations.

The types of music users different from radio stations are covered by goal **G3**:

- **G3: Outsource the contracting of the remaining music users to branch organizations.**

This goal is measured with the presence or absence of an agreement with a branch organization for managing the rights for music users that are not radio stations. We operationalize **G3** with a Boolean variable that shows if the service of contracting music users is delivered by a third party or not.

- **v₃: Outsourced product delivery [Boolean].**

The second main goal aims at spending the collected permission fees as payments to rights owners. It is:

- **G4: Distribute collected permission fees among rights owners.**

The goal is measured with the money per year paid out to performing artists and producers. The percentage of the paid off money out of all collected permission fees can be an evaluation criterion for the goal. We operationalize the goal with:

- **v₄: Paid off money [Euro/Year].**

The information needed to distribute the collected money is: (1) play-list data – how many times is each song played; (2) the artists performing each song; and (3) the bank account of each artist. (1) and (2) are needed for repartitioning; (3) is needed for the actual transfer.

To fulfil goal **G4**, two other goal must be achieved. First, the collected permission fees have to be repartitioned¹ and second the due amounts have to be transferred to the rights owners. The first of the two decomposition sub-goals of goal **G4** is:

- **G5: Repartition the collected permission fees.**

Goal **G5** is satisfied if at the end of the accounting period the rights society can calculate the amounts due to every rights owner. In all other cases, the goal is not satisfied. Therefore, we operationalize goal **G5** with a Boolean variable.

- **v₅: Repartitioned [Boolean].**

To achieve goal **G5**, the rights society must have data about the broadcasted songs by every radio station. This is a valid sub-goal but may result in very high operation costs because contracting small radio stations does not add new information for the

¹ By repartition, we mean to distribute the amount of money among all rights owners proportionally to the broadcasted music content.

repartitioning but incurs costs. Therefore, we relax the goal to collecting play-lists statistics only from the top¹ radio stations.

- **G6: Get play-lists from top radio stations.**

Goal **G6** is based on the assumption that the top radio stations represent the preferences of the audience and can be used to measure which song is listened to how many times and by how many people. The assumption is justified despite that small radio stations offer non-mainstream music content: the number of listeners is not enough to change significantly the statistics from the big radio stations.

We operationalize goal **G6** with:

- **v₆: Song-listener coverage [Percentage].**

Variable v_6 represents the percentage of all broadcasted songs for which we have statistical data. For example, there are two radio stations. The first one broadcasts 100 songs to 80 listeners. The second one broadcasts 50 songs to 20 listeners. The total amount of songs broadcasted to a single listener is $100 \cdot 80 + 50 \cdot 20 = 9000$. If we have statistical data about the specific tracks played only from the first radio station, then we have a song-listener coverage of $(100 \cdot 80 / 9000) \cdot 100 = 88.89\%$. This means that we know the name of the song and the number of listeners of that song for 88.89% if all songs were played to a single listener.

The second decomposition sub-goal of goal **G4** (**G4: Distribute collected permission fees among rights owners**) is a goal that aims at transferring of due amounts the rights owners. This is goal **G7**:

- **G7: Transfer money to rights owners.**

We measure goal **G7** with the amount of money per year paid to performing artists and producers. The operationalization is:

- **v₇: Transferred money [Euro/Year].**

To achieve goal **G7**, we need to collect information about the rights owners. From the statistics provided by the top-radio stations, we know what the songs are but we do not know to whom and where to transfer the money. Thus, a sub-goal of goal **G7** is a goal that aims at collecting information about rights owners.

- **G8: Register rights owners.**

The measurement of **G8** would require counting the number of rights owners for whom we do not have the necessary information. Similarly to goal **G6**, this would not be informative enough because one unknown, e.g., producer with a big share of the collected permission fees would impact the money distribution greater than, e.g., 10 unknown singers performing only one song. Therefore, we measure goal **G8** with the percentage of all collected permission fees that cannot be transferred because of lack of information.

We operationalize **G8** with:

- **v₈: Non-transferable money [Percentage].**

The non-transferable money is the total sum of shares produced during repartitioning for which we do not have owners' back transfer information. Variable v_8 is percentage of the non-transferable money out of all collected permission fees.

The third, and last, main goal of a rights society is to be efficient: i.e., to spend minimum money on operation costs. Thus, goal **G9** is:

- **G9: Minimize operation costs.**

¹ Top radio stations are the ones with the biggest audience.

We measure goal **G9** with the amount of money per year spent to cover operations expenses. The goal is considered achieved if these expenses are less than 11%¹ of total amount of collected permission fees. We operationalize the goal with:

- **v₉: Operations costs [Euro/Year].**

A rights society has two main activities and their execution results in making expenses. The two activities are clearing of rights for making content public and repartitioning collected permission fees. Consequently, we decompose goal **G9** to two sub-goals: minimize clearing costs and minimize repartitioning costs.

Further, we analyze the first sub-goal:

- **G10: Minimize clearing costs.**

Similarly to goal **G9**, we measure the goal with the money per year spent to execute the clearing of rights. We operationalize the goal with:

- **v₁₀: Clearance costs [Euro/Year].**

To clear the rights to make music content public, a rights society has to find the music users and sign a contract with them. Similarly to goals **G2** and **G3**, the music users are divided into radio stations and remaining music users². Goals **G11** and **G14** reflect the clearing costs from the two types of music users, respectively.

The motivation for goal **G11** is as follows. The costs associated with contracting music users can be a significant. Additionally, radio stations vary in size (number of listeners) and, correspondingly, in importance for the rights society. A contract with a big radio station is worth the money and effort; while a small radio station does not justify the money spent on, e.g., negotiating an agreement. To be efficient, a rights society has to fragment the radio stations; i.e., to use different approaches to radio station depending on their size. Thus, the goal of reducing the clearing costs of radio stations is:

- **G11: Fragment radio stations.**

We measure goal **G11** with the presence of market segments of radio stations. A segment is a group of radio stations with preferences towards a particular type of contract. That is, we count the number of music users that opt taking particular arrangement of rights clearance with the rights society.

We operationalize goal **G11** with a Boolean variable that captured the presence or absence of segmentation.

- **v₁₁: Fragmented market [Boolean].**

The rights society has to offer customized term and conditions to radio stations; e.g., to offer several-rounds negotiation for the top radio stations and fixed contracts for small radios. Therefore, goal **G11** is decomposed to **G12** and **G13**, which, respectively, are:

- **G12: Negotiate with big radio stations.**

Goal **G12** is measured with the fact that the rights society offers a customizable contract to the top radio stations. The contract can be negotiated. We operationalize the goal with:

- **v₁₂: Offered customizable contract [Boolean].**

- **G13: Offer fixed contracts to small radio stations.**

Goal **G13** is measured with the fact that the rights society offers a simple contract to the small radio stations. The contract is fixed (non-negotiable) and includes the basic terms and condition of rights transfer. We operationalize the goal with:

- **v₁₃: Offered simple contract [Boolean].**

The second goal addressing the clearing costs of the remaining music users is:

¹ The exact percentage is a norm imposed by regulatory body to a rights society like SENA.

² Remaining music users includes bars, discos, shops etc.

- **G14: Minimize search costs with the remaining music users.**

We measure goal **G14** with the money spent per year on searching and contacting music users. We operationalize the goal with:

- **v₁₄: Search costs [Euro/Year].**

Above, we decomposed goal **G9** to two sub-goals: **G10: Minimize clearing costs** and **G15: Minimize repartitioning costs**. Below, we analyze this second sub-goal:

- **G15: Minimize repartitioning costs.**

We measure the goal **G15** with the money spent per year to repartition the collected permission fees. We operationalize the goal with:

- **v₁₅: Repartitioning costs [Euro/Year].**

For a rights society to be able to repartition the collected permission fees, it needs information about broadcasted songs, number of listeners and rights owners. Additionally, the rights society has to transfer the permission fees to the rights owners, which also incurs costs. When achieved, goal **G6 (G6: Get play-lists from top radio stations)** guarantees the necessary information about songs and listeners. The missing piece of information is the identification information of rights owners. The costs to acquire this are captured within goal **G16**, which is one of the decomposition sub-goals of goal **G9**. The second decomposition sub-goal is goal **G18** and this account for the costs of transferring permission fees.

- **G16: Minimize rights owners search costs.**

Similarly to all other goals that aim at keeping costs low, we measure the goal **G18** with the money spent per year on registering rights owners. We operationalize the goal with:

- **v₁₆: Rights owners search costs [Euro/Year].**

The costs for searching and registering of rights owners can be reduced by outsourcing part of the activities to the entities (rights owners) with a stake in them. Namely, provide means for self-registering. This is a goal that affects goal **G18**, thus:

- **G17: Provide self-registering.**

We measure goal **G17** with the percentage of all artists that used the interface for self-registering. Therefore, we operationalize **G17** with:

- **v₁₇: Registered clients [Percentage].**
- **G18: Minimize payment costs.**

Similarly to all other goals that aim at keeping costs low, we measure the goal **G18** with the money spent per year on transferring permission fees to the rights owners. We operationalize the goal with:

- **v₁₈: Payment costs [Euro/Year].**

The payment costs can be reduced by outsourcing the payment to a third party. This is a goal that affects goal **G18**, thus:

- **G19: Outsource payment.**

We measure goal **G19** with the presence or absence of third party to execute the transactions of money transfer. Therefore, we operationalize **G19** with a Boolean variable:

- **v₁₉: Outsourced product delivery [Boolean].**

2.2.2 Goal Model of Rights Society

The goals of the rights society, listed in the previous subsection, and their relations are shown in Figure VIII-2. This is the goal model of a rights society.

The goals in the goal model are denoted only with their abbreviation and number, corresponding to the goal descriptions in the previous subsection. To avoid further name

conflicts as other goals may be abbreviated in a similar way, we uniquely identify the goals by prefixing their names with a code that refers to the goal model to which the goals belong. The code for the right society goal model is **SOC**. Thus, the full name of goal, e.g., **G1** is **SOC.G1**.

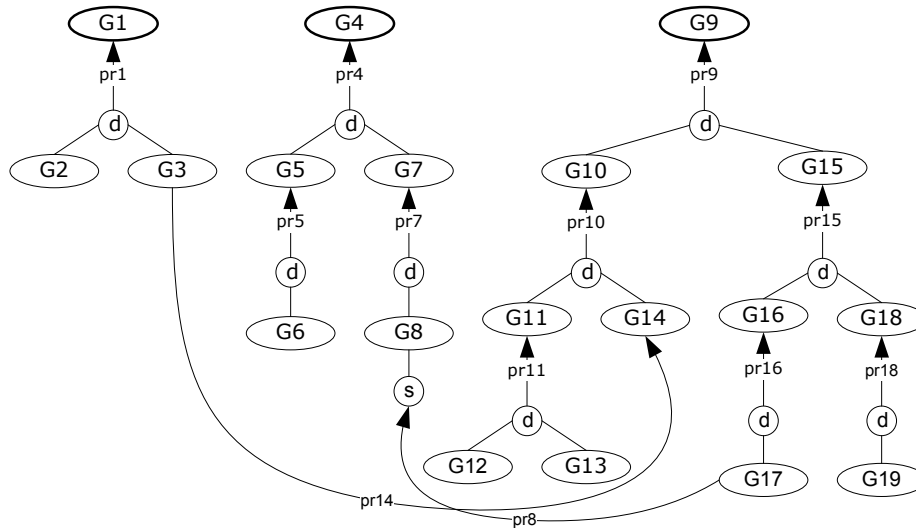


Figure VIII-2. Goal model of a rights society

Table VIII-4 summarizes the goals of a rights society and their operationalization.

Table VIII-4. Goals of a rights society and their operationalization

Goal code	Goal	
Variable code	Variable	Variable domain
G1	Maximize the number of charged music users	
v ₁	Collected permission fees	Euro currency per year
G2	Maximize contracted radio stations	
v ₂	Rights to play a song to a listener per year	Natural numbers plus 0
G3	Outsource the contracting of the remaining music users to branch organizations	
v ₃	Outsourced product delivery	Boolean
G4	Distribute collected permission fees among rights owners	
v ₄	Paid off money	Euro currency per year
G5	Repartition the collected permission fees	
v ₅	Repartitioned	Boolean
G6	Get play-lists from top radio stations	
v ₆	Song-listener coverage	Percentage

G7	Transfer money to rights owners	
v ₇	Transferred money	Euro currency per year
G8	Register rights owners	
v ₈	Non-transferable money	Percentage
G9	Minimize operation costs	
v ₉	Operations costs	Euro currency per year
G10	Minimize clearing costs	
v ₁₀	Clearance costs	Euro currency per year
G11	Fragment radio stations	
v ₁₁	Fragmented market	Boolean
G12	Negotiate with big radio stations	
v ₁₂	Offered customizable contract	Boolean
G13	Offer fixed contract to small radio stations	
v ₁₃	Offered simple contract	Boolean
G14	Minimize search costs with the remaining music users	
v ₁₄	Search costs	Euro currency per year
G15	Minimize repartitioning costs	
v ₁₅	Repartitioning costs	Euro currency per year
G16	Minimize rights owners search costs	
v ₁₆	Rights owners search costs	Euro currency per year
G17	Provide self-registering	
v ₁₇	Registered clients	Percentage
G18	Minimize payment costs	
v ₁₈	Payment costs	Euro currency per year
G19	Outsource payment	
v ₁₉	Outsourced product delivery	Boolean

2.2.3 Propagation Functions within the Goal Model of a Rights Society

The goal model of a rights society in Figure VIII-2 contains 19 goals which are related among each other with 13 goal relationships: 10 decomposition, 1 substitution and 2 dependency relationships. Each of the relationships is associated with a propagation function which captures the effect goals are causing one another. The number of propagation functions equals the number of head-goals: in our particular case, this is 13.

All propagation functions are a common sense approximation of what a real-life relation between the variables could be. The assumptions for the shape of the functions come in addition to the simplifications made while identifying the goals of a rights society. The assumed propagation functions demonstrate the use of our method; the introduced imprecision does not invalidate the method.

Below, we describe the 13 functions:

Propagation function **pr1** is associated with the decomposition relationship between goal **G1** (the head of the relationship) and goals **G2** and **G3** (the tails of the relationship.) Propagation function **pr1** calculates the value of goal variable v_1 out of the values of goal variables v_2 and v_3 . In other words, the function calculates the collected permission fees out of the sold rights to play a song to a listener and a customer. The rights to play a single song to a single listener are negotiated individually between the rights society and the radio station. Here, we assume that the price for a radio is on average 0.00007 euros and for the remaining music users the price is 0.00001 euros¹. Further, we assume that the branch organization represent a potential market of 45000000 song-listeners per day. We calculate the collected permission fees by summing up the permission fees from all music users. Thus, the propagation function is:

$$\circ \quad v_1 = \text{pr1}(v_2, v_3) = 0.00007 * v_2 + 0.00001 * 45000000 * 365 * v_3.$$

Propagation function **pr4** is associated with the decomposition relationship between goal **G4** (the head of the relationship) and goals **G5** and **G7** (the tails of the relationship.) Propagation function **pr4** calculates the value of goal variable v_4 out of the values of goal variables v_5 and v_7 . In other words, the function calculates the money paid off to rights owners out of the fact that the money was repartitioned and the amount actually transferred. The paid off money is the transferred money but only in case it was repartitioned. Thus, the propagation function is:

$$\circ \quad v_4 = \text{pr4}(v_5, v_7) = v_5 * v_7.$$

Propagation function **pr5** is associated with the decomposition relationship between goal **G5** (the head of the relationship) and goal **G6** (the tail of the relationship.) Propagation function **pr5** calculates the value of goal variable v_5 out of the value of goal variable v_6 . In other words, the function determines when is possible to do a repartitioning: how much data about songs and audience is needed to distribute the collected money. We assume that if the song-listener coverage is greater or equal to 85% then it is possible to repartition the collected money. Thus, the propagation function is:

$$\circ \quad v_5 = \text{pr5}(v_6) = v_6 \text{ div } 85.$$

Propagation function **pr7** is associated with the decomposition relationship between goal **G7** (the head of the relationship) and goal **G8** (the tail of the relationship.) Propagation function **pr7** calculates the value of goal variable v_7 out of the value of goal variable v_8 . In other words, the function calculates the amount of transferred money out of the calculation of non-transferable money. We assume that the total amount of collected permission fees is 30000000 euros². The transferred money is all the collected money minus the percentage of non-transferable. The propagation function is:

$$\circ \quad v_7 = \text{pr7}(v_8) = 30000000 - 30000000 * v_8 / 100.$$

Propagation function **pr8** is associated with the dependency relationship between goal **G8** (the head of the relationship) and goal **G17** (the tail of the relationship.) Propagation function **pr8** calculates the value of goal variable v_8 out of the value of goal variable

¹ The prices of 0.00007 euros per song per listener and 0.00001 per song per customer are examples. The accurate price per song per listener can be found on the Web site of Recording Industry Association of America (RIAA), http://www.riaa.com/issues/licensing/webcasting_faq.asp#doespay. Shows are not considered core music users; therefore, the price per song per customer is more attractive.

² The amount of collected permission fees is different compared to the calculated by propagation function **pr1** because here the operation costs are subtracted.

v_{17} . In other words, the function calculates the percentage of money that cannot be transferred due to unknown recipient out of the percentage of registered rights owners. Because not every rights owners gets an equal share of the permission fees, the function between non-transferable money and registered rights owners is not linear. Nevertheless, it follows the trend that the more the registered rights owners the less the non-transferable money. We assume that the percentage of non-registered rights owners consists of individuals with small stakes in the collected permission fees, which means that they all have comparably the same effect of the percentage of non-transferable money. We approximate: the percentage of money that cannot be transferred is half of the percentage of unregistered rights owners. Thus, we use the following linear function to represent the propagation function **pr8**:

- $v_8 = \text{pr8}(v_{17}) = (100 - v_{17})/2$.

Propagation function **pr9** is associated with the decomposition relationship between goal **G9** (the head of the relationship) and goals **G10** and **G15** (the tails of the relationship.) Propagation function **pr9** calculates the value of goal variable v_9 out of the values of goal variables v_{10} and v_{15} . In other words, the function calculates the operations costs out of the clearance and the repartitioning costs. The operations costs are the total sum of the costs from clearance and repartitioning. The propagation function is:

- $v_9 = \text{pr9}(v_{10}, v_{15}) = v_{10} + v_{15}$.

Propagation function **pr10** is associated with the decomposition relationship between goal **G10** (the head of the relationship) and goals **G11** and **G14** (the tails of the relationship.) Propagation function **pr10** calculates the value of goal variable v_{10} out of the values of goal variables v_{11} and v_{14} . In other words, the function calculates the clearance costs out of the search, negotiation and contracting costs per radio stations and the remaining music users. The clearance costs are the total sum of the costs from search, negotiation and contracting. To be able to calculate the clearance costs, we need the following data: clearance costs for radio stations in case of segmented and non-segmented markets, and the clearance costs of the remaining music users. We assume that the clearing costs per radio stations are 200000 euros per year and eventual segmentation reduces them by 75%. Consequently, the propagation function is:

- $v_{10} = \text{pr10}(v_{11}, v_{14}) = 200000*(1 - v_{11}*0.75) + v_{14}$.

Propagation function **pr11** is associated with the decomposition relationship between goal **G11** (the head of the relationship) and goals **G12** and **G13** (the tails of the relationship.) Propagation function **pr11** calculates the value of goal variable v_{11} out of the values of goal variables v_{12} and v_{13} . In other words, the function calculates the segmentation of radio stations out of the offered simple and customizable contracts. If there are two types of offered contracts then the market is segmented. The propagation function is:

- $v_{11} = \text{pr11}(v_{12}, v_{13}) = v_{12} \text{ and } v_{13}$.

Propagation function **pr14** is associated with the dependency relationship between goal **G14** (the head of the relationship) and goal **G3** (the tail of the relationship.) Propagation function **pr14** calculates the value of goal variable v_{14} out of the value of goal variable v_3 . In other words, the function calculates the search costs for music users that are not radio stations out of the type of contracting mechanism adopted by the rights society. To be able to do the calculation, we need to know the contracting costs of the music users in case of outsourcing and in-society execution. We assume that the costs are

100000 euros per year and the decrease in case of outsourcing is 20000 euros per year. The propagation function is:

$$\circ \quad \mathbf{v_{14}} = \mathbf{pr14(v_3)} = \mathbf{100000 - 20000*v_3}.$$

Propagation function **pr15** is associated with the decomposition relationship between goal **G15** (the head of the relationship) and goals **G16** and **G18** (the tails of the relationship.) Propagation function **pr15** calculates the value of goal variable v_{10} out of the values of goal variables v_{16} and v_{18} . In other words, the function calculates the repartitioning costs out of the costs to search for rights owners and the cost to transfer payments. The propagation function is:

$$\circ \quad \mathbf{v_{15}} = \mathbf{pr15(v_{16}, v_{18})} = \mathbf{v_{16} + v_{18}}.$$

Propagation function **pr16** is associated with the decomposition relationship between goal **G16** (the head of the relationship) and goal **G17** (the tail of the relationship.) Propagation function **pr16** calculates the value of goal variable v_{16} out of the value of goal variable v_{17} . In other words, the function calculates the rights owners search costs out of the percentage of registered clients. To be able to make the calculation, we need information about the costs associated with the search of a single rights owner and the number of all rights owners. We assume that the search costs are 1000 euros per rights owner and that the rights owners are 100 new individuals per year. The rights owners search costs per year are the percentage of unregistered right owners times the costs of registration. Then, the propagation function is:

$$\circ \quad \mathbf{v_{16}} = \mathbf{pr16(v_{17})} = \mathbf{1000*100*(100 - v_{17})/100}.$$

Propagation function **pr18** is associated with the decomposition relationship between goal **G18** (the head of the relationship) and goal **G19** (the tail of the relationship.) Propagation function **pr18** calculates the value of goal variable v_{18} out of the value of goal variable v_{19} . In other words, the function calculates the payment costs out of the type of payment mechanism adopted by the rights society. To be able to do the calculation, we need to know the payment costs in case of outsourcing and in-society execution. We assume that the costs are 10000 euros and the decrease in case of outsourcing is 2000 euros. The propagation function is:

$$\circ \quad \mathbf{v_{18}} = \mathbf{pr18(v_{19})} = \mathbf{10000 - 2000*v_{19}}.$$

2.2.4 Anticipated Values for Goals in the Goal Model of a Rights Society

The goal model of a rights society includes goals that: (1) are already achieved, (2) capture a known problem with a ready to reuse solution, and (3) represent an innovation with only a vague idea for implementation. To enable the comparison of old and new designs and the evaluation of partial designs, we have to consider the satisfied goals and assume values for the goals without a match. Thus as part of the goals model, we assign anticipated values to goal variables.

In particular for the goal model of a rights society, we have anticipated to achieve values for eight goals: goals **G2**, **G6**, **G9**, **G10**, **G14**, **G15**, **G16**, and **G18**. Respectively, the values these get are:

- v_2 is assigned a value of 560 640 000 000. The value is derived from the following assumptions. There are 8 big radio station, each has constant audience of 1 000 000 listeners, and each plays 8 songs in an hour. Thus, $8*1\ 000\ 000*8*24*365=560\ 640\ 000\ 000$.
- v_6 is assigned a value of 87. The value is taken as an example.

- v_9 is assigned a value of 300 000. The value is derived from the following assumptions. There are 8 big radio station, each has constant audience of 1 000 000 listeners, and each plays 8 songs in an hour. Thus, $8*1\ 000\ 000*8*24*365=560\ 640\ 000\ 000$.
- v_{10} is assigned a value of 200 000. We previously assumed that value in Section 2.2.3, when discussion propagation function **pr10**.
- v_{14} is assigned a value of 100 000. We previously assumed that value in Section 2.2.3, when discussion propagation function **pr14**.
- v_{15} is assigned a value of 37 000. In the model, v_{15} is calculated as a sum of v_{16} and v_{18} . From the two anticipated values below, it follows that the value of v_{15} has to be 40 000. Nevertheless, anticipated values are not calculated within the model. To exemplify this, we take a value of 37 000.
- v_{16} is assigned a value of 30 000. Previously, we assumed that the search costs are 1000 euros per rights owner and that the rights owners are 100 new individuals per year. The search costs per year are the percentage of unregistered right owners times the costs of registration. Here, we assume that the registered rights owners are 70%.
- v_{18} is assigned a value of 10 000. We previously assumed that value in Section 2.2.3, when discussion propagation function **pr18**.

The anticipated values of goals from the goal model of a rights society are summarized in Table VIII-5.

Table VIII-5. Anticipated values for goals in the goal model of a rights society

<i>Capability code</i>	<i>Goal</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Anticipated value</i>
G2	Maximize contracted radio stations		
v_2	Rights to play a song to a listener	Natural numbers plus 0	560 640 000 000
G6	Get play-lists from top radio stations		
v_6	Song-listener coverage	Percentage	87
G9	Minimize operation costs		
v_9	Operations costs	Euro currency	300 000
G10	Minimize clearing costs		
v_{10}	Clearance costs	Euro currency	200 000
G14	Minimize search costs with the remaining music users		
v_{14}	Search costs	Euro currency	100 000
G15	Minimize repartitioning costs		
v_{15}	Repartitioning costs	Euro currency	37 000
G16	Minimize rights owners search costs		
v_{16}	Rights owners search costs	Euro currency	30 000
G18	Minimize payment costs		
v_{18}	Payment costs	Euro currency	10 000

3 Applying Our Method

In the following three sections, we demonstrate the application of our development method for the network constellation of businesses in the real-life example of making music content public. We apply the method for the actors radio station and rights society.

In Section 4, we describe in details the development of the value perspective. Following the procedure outlined in Chapter V Selection Procedure, we first identify the alternative goal models. For these, we match all leaf-goals with capabilities from the value pattern library. Consequently, the alternative goal models and the matching capability models undergo a number of transformations resulting in a single set of patterns that satisfies best the top-level goals of the combined goal model of a radio station and rights society. Further, the selected patterns are synthesized into a value model following the procedure outlined in Chapter VI Synthesis of Value and Process Patterns. Namely, the patterns are first instantiated and then linked to a single model.

In Section 5, we describe in details the development of the process perspective. We follow a procedure similar to the procedure for developing the value perspective. Specifically, alternative goal models are generated, which are further matched with capabilities of process patterns in the process library. A number of the matching patterns are selected, such that the top-level goals in the goal model are satisfied the best. After the patterns are instantiated, a process model is synthesized.

The only difference in content between Section 4 and 5 is the extending of the goal model in Section 5. (The design methods for the value and process perspectives do not differ.) The extending of the goal model is needed to have more matching goals. It does not affect the value perspective. The motivation and details are given in Section 5.

In Section 6, we check the developed perspectives for consistency. Following the outlined in Chapter VII Consistency between Value and Process Models, we transform the value and process model to reduced models. After that, we compare the two reduced models and verify the consistency.

4 Developing the Value Perspective

The starting position of this section is that we have (i) a library of value patterns and (ii) a goal model of the desired system which includes several businesses interacting to achieve individual goals. We want to develop a value model of the system. Thus, we have to select the relevant design patterns and to synthesize a value model out of the patterns.

The selection procedure is described in Chapter V Selection Procedure. In this section, we apply it and describe in detail the following steps. (The steps follow the selection procedure; though, the names or granularity may differ.)

- Generate alternative goal models
- Match goals and capabilities: build Matching table
- Find alternative solution capability models
- Propagate values in the alternative solution capability models
- Evaluate the alternative solution capability models: select the best alternative solution capability models

- Integrate each of the best alternative solution capability models with the capability models
- Propagate values and evaluate the best alternative solution capability models

The synthesis procedure is described in Chapter VI Synthesis of Value and Process Patterns. In this section, we apply it and describe in detail the following steps. (The steps follow the selection procedure; though, the names or granularity may differ.)

- Instantiate Patterns
 - Map the role of the matching capability to a business
 - Resolve the remaining roles in the pattern
 - Modify value objects
- Synthesize Patterns
 - Find identical actors.
 - Resolve cardinality between identical actors.
 - Add the pattern to the value model.
 - Merge identical actors.

Below, we describe the development of the value perspective step by step. Starting with Section 4.1.1 until Section 4.1.7, we show the procedure of selecting relevant pattern. Further from Section 4.2.1 to Section 4.2.2, we present the synthesis of the selected patterns.

4.1 Selection of Value Patterns

The business network (the system) that we want to design includes radio stations and a rights society which the radio stations use to clear their rights to make music content public¹. The goal model of the system is the goal model of a radio station (see Figure VIII-1) and the goal model of a rights society (see Figure VIII-2) combined. Further, we refer to the goal model of the real-life example as the goal model and when the model of a particular business is meant then it is explicitly referred to.

¹ Potentially, the business network includes more businesses. Nevertheless, they are considered secondary and are not represented by a goal model because we focus on clearance of rights.

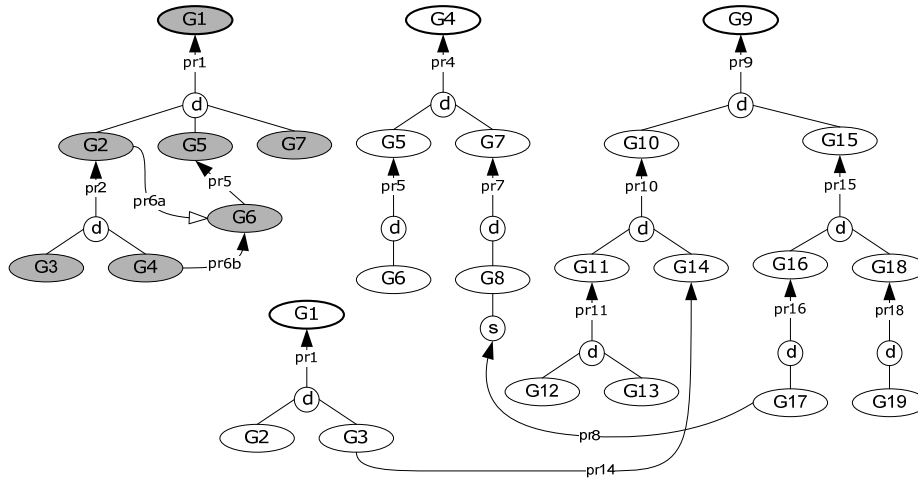


Figure VIII-3. Goal model of the business network under design

Figure VIII-3 depicts the combined goal models. The goals are labelled with their names without the prefix denoting to which goal model they belong. The grey goals belong to the goal model of a radio station; the white goals belong to the goal model of a rights society.

4.1.1 Generate Alternative Goal Models

The goal model contains 26 goals: 7 belong to the goal model of a radio station and 19 belong to the goal model of a rights society. Following the algorithm for generation of alternative goal model described in Chapter V Selection Procedure Section 4.1, we create 2912 alternative goal model¹. Table VIII-6 presents a sample of 10 alternative goal models. The columns list the goals with their full names i.e., the name is extended with a prefix denoting the goal model to which the goals belong. A row represents an alternative goal model where the cells marked with 'x' correspond to the goals participating in the alternative goal model. For example, the first row means a goal model made up of goals **RAD.G2, RAD.G4, RAD.G7, SOC.G1, SOC.G6, SOC.G11, SOC.G14, SOC.G16, SOC.G17, and SOC.G19.**

¹ The alternative goal models are managed in an Excel electronic table

Table VIII-6. Randomly selected sample of 10 out of 2912 alternative goal models

	RAD.G1	RAD.G2	RAD.G3	RAD.G4	RAD.G5	RAD.G6	RAD.G7	SOC.G1	SOC.G2	SOC.G3	SOC.G4	SOC.G5	SOC.G6	SOC.G7	SOC.G8	SOC.G9	SOC.G10	SOC.G11	SOC.G12	SOC.G13	SOC.G14	SOC.G15	SOC.G16	SOC.G17	SOC.G18	SOC.G19
x		x		x			x	x					x					x			x		x			x
	x								x	x			x					x			x		x			x
		x	x	x			x		x	x			x					x			x		x			x
	x					x	x		x	x			x					x			x		x			x
		x	x			x	x		x	x			x					x			x		x			x
	x		x			x		x	x				x					x			x		x			x
x								x			x							x	x	x		x				x

4.1.2 Match Goals and Capabilities: Build Matching Table

Each of the 26 goals in the goal model is checked for a match with a capability from the capability models in the library of value patterns (Appendix E Library of Value Patterns). The result is that four patterns have capability models with top-level capabilities that match goals from the goal model. The number of matches is five as one of the capabilities matches two goals. The result is presented in Table VIII-7 which is the Matching table. The columns list the goals with their full names; whereas, the rows contain matching capabilities. The table presents which capability matches which goal by means of cells marked with an ‘x’.

Table VIII-7. The Matching table for the goal model

	RAD.G1	RAD.G2	RAD.G3	RAD.G4	RAD.G5	RAD.G6	RAD.G7	SOC.G1	SOC.G2	SOC.G3	SOC.G4	SOC.G5	SOC.G6	SOC.G7	SOC.G8	SOC.G9	SOC.G10	SOC.G11	SOC.G12	SOC.G13	SOC.G14	SOC.G15	SOC.G16	SOC.G17	SOC.G18	SOC.G19
ADV.C2			x																							
INS.C3										x																x
SEG.C1																		x								
REG.C1																								x		

The result of matching of goals and capabilities is that the matched goals receive values from the capabilities. Goal variables get values also during the goal modelling when anticipated values are assigned to some goals. Table VIII-8 presents the known values of variables in the goal model. The table contains values received from the capabilities **ADV.C2**, **INS.C3**, **SEG.C1**, and **REG.C1**, shown in bold font, and the anticipated values taken from Table VIII-3 and Table VIII-5.

Table VIII-8. Variable values

goal	Value	goal	value
RAD.G1		SOC.G7	
RAD.G2		SOC.G8	
RAD.G3	True	SOC.G9	300 000
RAD.G4	12	SOC.G10	200 000
RAD.G5		SOC.G11	true
RAD.G6		SOC.G12	
RAD.G7	20 000 000	SOC.G13	
SOC.G1		SOC.G14	100 000
SOC.G2	560 640 000 000	SOC.G15	37 000
SOC.G3	True	SOC.G16	30 000
SOC.G4		SOC.G17	75
SOC.G5		SOC.G18	10 000
SOC.G6	87	SOC.G19	true

4.1.3 Find Alternative Solution Capability Models

The viable alternative goal models are these that contain only goals with known values: known either due to a matching capability or the value is anticipated. We refer to such alternative goal models as alternative solution capability models. From the 2912 alternative goal models only 16 constitute alternative solution capability models. Table VIII-9 presents these. The columns present the goals with their full names; whereas, the rows list the alternative solution capability models. The cells marked with ‘x’ determine which goal participates in which goal model.

Table VIII-9. Alternative solution capability models

	RAD.G1	RAD.G2	RAD.G3	RAD.G4	RAD.G5	RAD.G6	RAD.G7	SOC.G1	SOC.G2	SOC.G3	SOC.G4	SOC.G5	SOC.G6	SOC.G7	SOC.G8	SOC.G9	SOC.G10	SOC.G11	SOC.G12	SOC.G13	SOC.G14	SOC.G15	SOC.G16	SOC.G17	SOC.G18	SOC.G19
1			x	x			x		x	x			x			x								x		
2			x	x			x		x	x			x				x					x		x		
3			x	x			x		x	x			x					x			x	x		x		
4			x	x			x		x	x			x					x				x		x		
5			x	x			x		x	x			x				x						x	x	x	
6			x	x			x		x	x			x					x			x		x	x	x	
7			x	x			x		x	x			x					x					x	x	x	
8			x	x			x		x	x			x				x						x	x	x	
9			x	x			x		x	x			x					x			x			x	x	
10			x	x			x		x	x			x					x						x	x	
11			x	x			x		x	x			x				x						x	x		x
12			x	x			x		x	x			x					x			x		x	x		x
13			x	x			x		x	x			x					x					x	x		x
14			x	x			x		x	x			x				x							x	x	x
15			x	x			x		x	x			x					x			x			x	x	x
16			x	x			x		x	x			x					x						x	x	x

4.1.4 Propagate Values in the Alternative Solution Capability Models

To enable the evaluation of potential design determined by an alternative solution capability model, the known values in the goal model have to be propagated to the top-level goals. This is done according to the description in Chapter IV Goal-modelling Section 5.

The goal model contains 4 top-level goals:

- **RAD.G1**;
- **SOC.G1**;
- **SOC.G4**; and
- **SOC.G9**.

For each of the 16 alternative solution capability models, the values of the top-level goals are determined¹. Table VIII-10 presents the calculated values. The columns of the table contain the values for the particular top-level goal; whereas, the rows list the alternative solution capability models.

Table VIII-10. Values for the top-level goals

	RAD.G1 [euros]	SOC.G1 [euros]	SOC.G4 [euros]	SOC.G9 [euros]
1	44 920 457,43	39 409 050	26 250 000	300 000
2	44 920 457,43	39 409 050	26 250 000	237 000
3	44 920 457,43	39 409 050	26 250 000	187 000
4	44 920 457,43	39 409 050	26 250 000	167 000
5	44 920 457,43	39 409 050	26 250 000	240 000
6	44 920 457,43	39 409 050	26 250 000	190 000
7	44 920 457,43	39 409 050	26 250 000	170 000
8	44 920 457,43	39 409 050	26 250 000	235 000
9	44 920 457,43	39 409 050	26 250 000	185 000
10	44 920 457,43	39 409 050	26 250 000	165 000
11	44 920 457,43	39 409 050	26 250 000	238 000
12	44 920 457,43	39 409 050	26 250 000	168 000
13	44 920 457,43	39 409 050	26 250 000	168 000
14	44 920 457,43	39 409 050	26 250 000	233 000
15	44 920 457,43	39 409 050	26 250 000	183 000
16	44 920 457,43	39 409 050	26 250 000	163 000

4.1.5 Evaluate the Alternative Solution Capability Models: Select the Best Alternative Solution Capability Models

According to our selection procedure (see Chapter V Selection Procedure, Section 7,) the alternative solution capability models are rated on the basis of the values of the top-level goals. In general, we use a multiple-criteria decision making method because each model is evaluated with several values. In the particular case at hand, the alternative solution capability models have four top-level goals. This necessitates the use of a multiple-criteria decision making method. Although, the actual values for the variables of the top-

¹ The propagation of values is calculated in a goal model represented in an Excel electronic table

level goals simplify our multiattribute solving problem. The values of the top-level goals **RAD.G1**, **SOC.G1**, and **SOC.G4** are the same for every alternative solution capability model, which allows us to rate the models based on only the **SOC.G9** top-level goal.

Goal **SOC.G9** aims at keeping the operation costs as low as possible (see Section 2.2.1). This means that the goal is closer to satisfaction if the value of the variables is a smaller number. Figure VIII-4 presents the values for the **SOC.G9** variable per each alternative solution capability model. The alternative solution capability model number 1 is the furthest from satisfaction with value of 300 000 euros of operation costs. Respectively, the alternative solution capability model number 16 is the closest to satisfaction with value of 163 000 euros of operation costs.

The purpose of evaluating the alternative solution capability models is to select the three best satisfying the top-level goals. In our case, these are models 16, 10, and 4, and they form the set of best alternative solution capability models. Figure VIII-4 depicts them with solid bars.

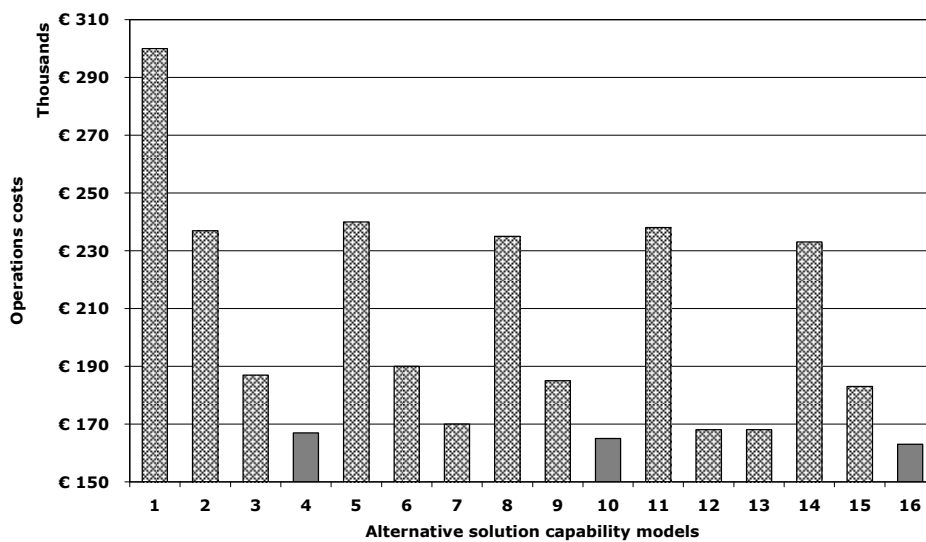


Figure VIII-4. Values for the SOC.G9 top-level goal

4.1.6 Integrate Each of the Best Alternative Solution Capability Models with the Capability Models

Selecting a design pattern as relevant for a particular system under development means that it is taken with all of its capabilities. This requires integrating the alternative solution capability models and the capability models (see Chapter V Selection Procedure, Section 6)

In our case, four patterns have top-level capabilities matching with goals. (Refer to Table VIII-7, which lists the matching capabilities, respectively capability models, and

goals.) Figure VIII-5 shows the capability models of the patterns: (a) – capability model of the Advertising (ADV) value pattern; (b) – capability model of the Insourcing (INS) value pattern; (c) – capability model of the Market segmentation (SEG) value pattern; and (d) – capability model of the Registration (REG) value pattern. The capability models are taken from Appendix E Library of Value Patterns where they are fully described. Each capability is shown with the role benefiting from it according to the pattern description.

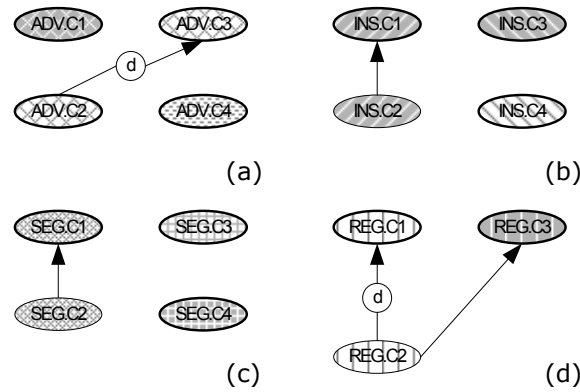


Figure VIII-5. Capability models: (a) – capability model of the Advertising (ADV) value pattern; (b) – capability model of the Insourcing (INS) value pattern; (c) – capability model of the Market segmentation (SEG) value pattern; and (d) – capability model of the Registration (REG) value pattern

Each of the best alternative solution capability models needs to be integrated with the four patterns from Figure VIII-5. According to Chapter V Selection Procedure Section 6, the integration step includes replacing goals with matching capabilities and adding relationship between capabilities and goals. For clarity of presentation and readability of the goal models, we add an extra step which removes goal and relations which cannot influence the values of the top-level goals. That is, we remove tail-goals in relationships in which the head- goal has a fixed value, coming from a capability or being anticipated.

We begin with alternative solution capability model number 16 as it is the best scoring model (see Figure VIII-4.) The matching capabilities and anticipated values apply to leaf-goals of the goal model of the system (shown in Figure VIII-3) with one exception. The **SOC.G11** goal matches capability **SEG.C1** and has two sub-goals. The consequence is that we begin the integration with remove goals **SOC.G12** and **SOC.G13** and the decomposition relationship with a head-goal goal **SOC.G11**. The result can be seen in Figure VIII-6 which depicts the integrated¹ alternative solution capability model: goals **SOC.G12** and **SOC.G13** are omitted from the model.

Further, we replace goals **RAD.G1**, **SOC.G2**, **SOC.G11**, **SOC.G17**, and **SOC.G19** with capabilities **ADV.C2**, **INS.C3**, **SEG.C1**, **REG.C1**, and **INS.C3**. Replacing goals with capabilities adds the complete capability models to the goals model. This is illustrated in

¹ Integrated means a goal model integrated with capability models. For details, refer to Chapter V Selection Procedure, Section 6.

Figure VIII-6 by adding the capabilities from one capability model in a vertical line such that the original position of the replaced goal is kept.

We finish the integration by adding the relationships between capabilities and goals. In the case of alternative solution capability model number 16, the matching capabilities do not interact with the goals. This leads to no added relationships.

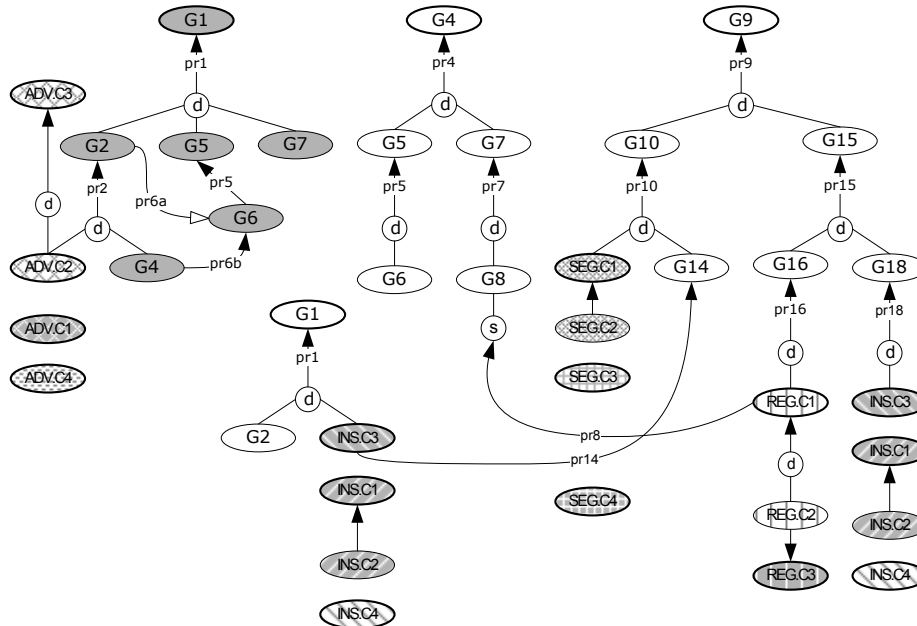


Figure VIII-6. Alternative solution capability model number 16 integrated with capability models ADV, INS, SEG, REG, and INS

The second alternative solution capability model to integrate is number 10 as it is the second-best scoring model (see Figure VIII-4.) The matching capabilities and anticipated values are such that: (1) we remove goals **SOC.G12** and **SOC.G13** and the decomposition relationship with a head-goal goal **SOC.G11**, for the same reasons as in the previous integration and (2) we remove goal **SOC.G19** and the decomposition relationship with a head-goal goal **SOC.G18**, due to the anticipated value for goal **SOC.G18**. The result can be seen in Figure VIII-7 which depicts the integrated alternative solution capability model: goals **SOC.G12**, **SOC.G13**, and **SOC.G19** are omitted from the model.

Further, we replace goals **RAD.G1**, **SOC.G2**, **SOC.G11**, and **SOC.G17** with capabilities **ADV.C2**, **INS.C3**, **SEG.C1**, and **REG.C1**. Similarly to Figure VIII-6, Figure VIII-7 adds the capabilities from one capability model in a vertical line such that the original position of the replaced goal is kept.

All matching patterns for alternative solution capability model number 10 match also goals in model number 16. This means that the included capabilities into the goal model do not interact with goals. This leads to no added relationships.

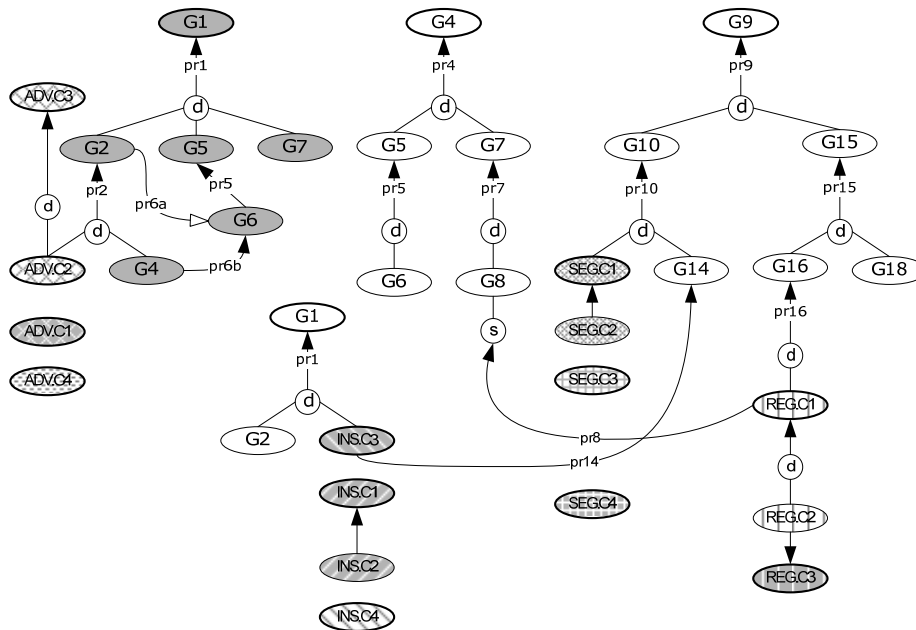


Figure VIII-7. Alternative solution capability model number 10 integrated with capability models ADV, INS, SEG, and REG

The third alternative solution capability model to integrate is number 4 as it is the third-best scoring model (see Figure VIII-4.). The matching capabilities and anticipated values are such that: (1) we remove goals **SOC.G12** and **SOC.G13** and the decomposition relationship with a head-goal goal **SOC.G11**, for the same reasons as in the previous integration; (2) we remove goals **SOC.G16** and **SOC.G18** and the decomposition relationship with a head-goal goal **SOC.G15**, due to the anticipated value for goal **SOC.G15**; (3) we remove goal **SOC.G19** and the decomposition relationship with a head-goal goal **SOC.G18**, due to the anticipated value for goal **SOC.G15**; and (4) we remove decomposition relationship with a head-goal goal **SOC.G16**, due to the anticipated value for goal **SOC.G15**. The result can be seen in Figure VIII-8 which depicts the integrated alternative solution capability model: goals **SOC.G12**, **SOC.G13**, **SOC.G16**, **SOC.G18**, and **SOC.G19** are omitted from the model.

Further, we replace goals **RAD.G1**, **SOC.G2**, **SOC.G11**, and **SOC.G17** with capabilities **ADV.C2**, **INS.C3**, **SEG.C1**, and **REG.C1**. Similarly to Figure VIII-6 and Figure VIII-7, Figure VIII-8 adds the capabilities from one capability model in a vertical line such that the original position of the replaced goal is kept.

Alternative solution capability models number 16 and 10 have a super set of matching pattern in comparison with model number 4. This means that the included capabilities into the goal model do not interact with goals. This leads to no added relationships.

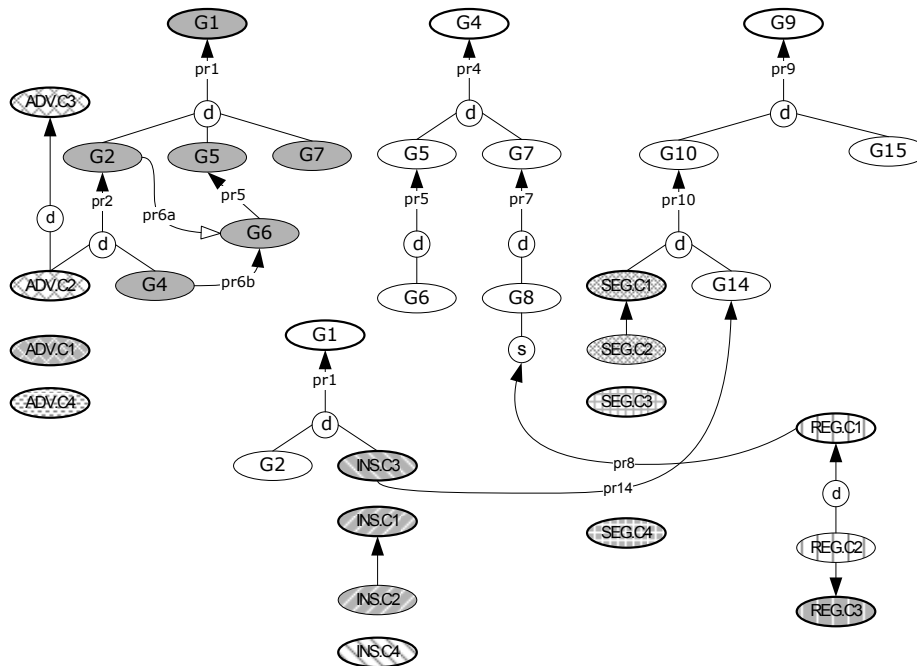


Figure VIII-8. Alternative solution capability model number 4 integrated with capability models ADV, INS, SEG, and REG

4.1.7 Propagate Values and Evaluate the Best Alternative Solution Capability Models

Due to the fact that no new relationships were added to any of the best alternative solution capability models, the propagation of values throughout the models leads to the same values for the top-level goals. Consequently, the evaluation leads to the same conclusion: alternative solution capability model number 16 has the best score and therefore is the one considered for further development.

4.2 Synthesis of Value Patterns

Synthesis links the value models of the selected patterns into one value model. The starting position is five patterns. These are: Advertising, two times Insourcing, Market Segmentation and Registration.

The names of roles and value objects used in the value model of a particular pattern may occur in the value model of other patterns. To avoid this conflict of name, we need to extend the name such that we uniquely identify the roles and value objects. Thus, each extended name contains the code of the pattern to which the role or value object belongs, and a number identifying patterns if more than one of a kind are matched with goals. An extended name has the following Backus-Naur form:

```
<extended name> ::= <pattern code> "." [<pattern number>
"."] <role or value object name>
```

For example, to refer to the Client role in the third matched Reservation pattern we use the following extended name: RES.3.Client.

4.2.1 Instantiate Patterns

Alternative solution capability model number 16 matches 5 patterns. Each one of them needs to be instantiated such that the roles and the value objects correspond to the business actors and values exchanges in the example at hand. The instantiation procedure includes the following three steps:

1. Map roles of matching goals and capabilities.
2. Resolve the remaining roles in the pattern.
3. Modify value objects.

Below, we instantiate each of the five patterns.

A. Advertising (ADV) pattern

The first pattern to instantiate is the Advertising (ADV) pattern which capability **ADV.C2** matches the radio station's goal **RAD.G3** (refer to Table VIII-7).

Step One: Map Roles Of Matching Goals And Capabilities. The **ADV.C2** capability benefits the ADV.Intermediary role (see Appendix E Library of Value Patterns, Pattern 1. Advertising, on page 301.) Respectively, the **RAD.G3** goal belongs to the Radio Station music user (see Section 2.1.2.) These two facts determine that the ADV.Intermediary role maps to the Radio Station actor.

Table VIII-11 presents the correspondence of roles and business actors for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of actor as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining roles in the pattern that need to be assigned to business actors are: ADV.Client and ADV.Advertiser. The information needed to resolve the mapping is available to the designer in the textual description of the pattern and the matching goal. Particularly, the client of an intermediary in the context of the pattern is the listener to a radio station in the real-life example context. Therefore, we map ADV.Client to Listener. Following the same line of reasoning, we map the ADV.Advertiser to Advertiser, a business that advertises in a radio broadcast. Table VIII-11 summarizes the mappings.

Table VIII-11. Correspondence table for roles in the Advertising (ADV) pattern

<i>Roles in the pattern</i>	<i>Business actors in the example</i>
ADV.Client	Listener
ADV.Intermediary	Radio Station
ADV.Advertiser	Advertiser

Step Three: Modify Value Objects. Similarly to the roles, the value objects from the pattern have to be modified such that they match the values exchanges in the real-life example. In the case of the Advertising pattern, only one value object needs

modification. Namely, Bundle of product and advertisement needs to be changed to Bundle of music content and commercials to reflect the actual bundling of products. Respectively, the correspondence of value objects is captured in Table VIII-12, wherein the first column contains the value objects as in the pattern and the second column the values objects in the real-life example.

Table VIII-12. Correspondence table for value objects in the Advertising (ADV) pattern

<i>Value objects as in the pattern</i>	<i>Value objects as in the example</i>
ADV.Bundle of product and advertisement	Bundle of music content and commercials
ADV.Attention	Attention
ADV.Exposure	Exposure
ADV.Fee	Fee

Value Model Transformation. Figure VIII-9 (a) depicts the value model of the pattern with the original roles and value objects. The instantiation of the pattern results in a value model in which the roles and the value objects are substituted with the corresponding business actors and value objects according to Table VIII-11 and Table VIII-12. Figure VIII-9 (b) shows the instantiated value model.

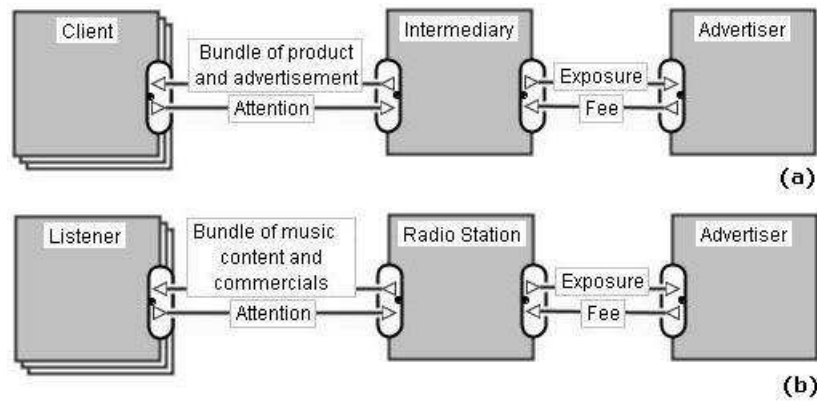


Figure VIII-9. Value models for the Advertising (ADV) pattern: (a) – original value model, copied from Appendix E Library of Value Patterns, Pattern 1. Advertising, on page 301 and (b) – instantiated value model

B. Insourcing (INS) pattern

The second pattern to instantiate is the first match with the Insourcing (INS) pattern which capability **INS.1.C3** matches the rights society’s goal **SOC.G3** (refer to Table VIII-7).

Step One: Map Roles Of Matching Goals And Capabilities. The **INS.1.C3** capability benefits the INS.1.Business role (see Appendix E Library of Value Patterns, Pattern 4.

Insourcing, on page 307.) Respectively, the **SOC.G3** goal belongs to the Rights Society (see Section 2.2.2.) These two facts determine that the INS.1.Business role maps to the Rights Society actor.

Table VIII-13 presents the correspondence of roles and business actors for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of actor as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining roles in the pattern that need to be assigned to business actors are: INS.1.Intermediary and INS.1.Client. The information needed to resolve the mapping is available to the designer in the textual description of the pattern and the matching goal. Particularly, the client of a business in the context of the pattern is a bar, disco or shop in the real-life example context. Therefore, we map INS.1.Client to Bar, Disco or Shop. The intermediary that delivers the product to the bars, discos, and shops is a branch organisation. Thus, we map the INS.1.Intermediary to Branch Organization. Table VIII-13 summarizes the mappings.

Table VIII-13. Correspondence table for roles in the Insourcing (INS) pattern

<i>Roles in the pattern</i>	<i>Business actors in the example</i>
INS.1.Intermediary	Branch Organization
INS.1.Business	Rights Society
INS.1.Client	Bar, Disco or Shop

Step Three: Modify Value Objects. In the case of the first Insourcing pattern, three value objects need modification. First, Product needs to be changed to Rights to make content public to reflect the actual product been delivered in the particular real-life example. Second, Product delivery needs to be changed to Contracting music users. And third, Money needs to be changed to Permission fees for the same reason. Respectively, the correspondence of value objects is captured in Table VIII-14, wherein the first column contains the value objects as in the pattern and the second column the values objects in the real-life example.

Table VIII-14. Correspondence table for value objects in the Insourcing (INS) pattern

<i>Value objects as in the pattern</i>	<i>Value objects as in the example</i>
INS.1.Product	Rights to make content public
INS.1.Product delivery	Contracting music users
INS.1.Money	Permission fees
INS.1.Fee	Fee

Value Model Transformation. Figure VIII-10 (a) depicts the value model of the pattern with the original roles and value objects. The instantiation of the pattern results in a value model in which the roles and the value objects are substituted with the corresponding

business actors and value objects according to Table VIII-13 and Table VIII-14. Figure VIII-10 (b) shows the instantiated value model.

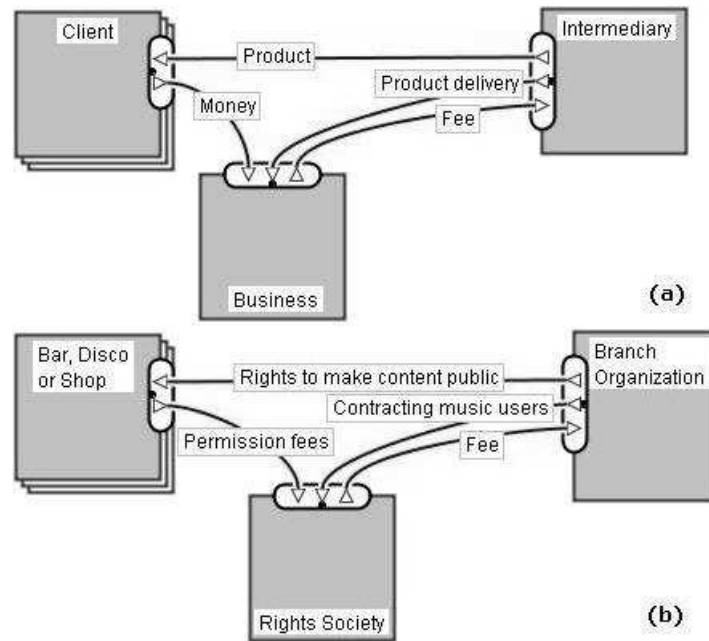


Figure VIII-10. Value models for the Insourcing (INS) pattern: (a) – original value model, copied from Appendix E Library of Value Patterns, Pattern 4. Insourcing, on page 307 and (b) – instantiated value model

C. Market Segmentation (SEG) pattern

The third pattern to instantiate is the Market Segmentation (SEG) pattern which capability **SEG.C1** matches the rights society's goal **SOC.G11** (refer to Table VIII-7).

Step One: Map Roles Of Matching Goals And Capabilities. The **SEG.C1** capability benefits the SEG.Provider role (see Appendix E Library of Value Patterns, Pattern 5. Market Segmentation, on page 309.) Respectively, the **SOC.G11** goal belongs to the Rights Society (see Section 2.2.2.) These two facts determine that the SEG.Provider role maps to the Rights Society actor.

Table VIII-15 presents the correspondence of roles and business actors for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of actor as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining roles in the pattern that need to be assigned to business actors are: **SEG.Basic Client** and **SEG.Experienced Client**. The information needed to resolve the mapping is available

to the designer in the textual description of the pattern and the matching goal. Particularly, the experienced clients of a provider in the context of the pattern are the big radio stations in the real-life example context. Therefore, we map SEG.Experienced Client to Radio Station. The internet radio stations are considered small in number of listeners and, therefore, they correspond to the basic clients. Thus, we map the SEG.Basic Client to Internet Radio Station. Table VIII-15 summarizes the mappings.

Table VIII-15. Correspondence table for roles in the Market Segmentation (SEG) pattern

<i>Roles in the pattern</i>	<i>Business actors in the example</i>
SEG.Provider	Rights Society
SEG.Basic client	Internet Radio Station
SEG.Experienced client	Radio Station

Step Three: Modify Value Objects. In the case of the Market Segmentation pattern, all value objects need modification. **Special price** offered to basic clients needs to be changed to **Fixed fee** as this is offered to interned radio stations in return of the rights to make music content public. The price that the radio stations pay for the same rights is the permission fees of the content producers. Following from this, **Price** is changes to **Permission fees** and **Product** delivered to both actors is changed to **Rights to make content public**. The premium that the rights society gets from the radio stations is play lists for which they compensate with negotiable term and conditions in the contract. Thus, **Premium** changes to **Play lists** and **Product complement** changes to **Negotiable term and conditions**. Table VIII-16 presents the correspondence of value objects, wherein the first column contains the value objects as in the pattern and the second column the values objects in the real-life example.

Table VIII-16. Correspondence table for value objects in the Market Segmentation (SEG) pattern

<i>Value objects as in the pattern</i>	<i>Value objects as in the example</i>
SEG.Special Price	Fixed fee
SEG.Product	Rights to make content public
SEG.Price	Permission fees
SEG.Premium	Play lists
SEG.Product complement	Negotiable term and conditions

Value Model Transformation. Figure VIII-11 (a) depicts the value model of the pattern with the original roles and value objects. The instantiation of the pattern results in a value model in which the roles and the value objects are substituted with the corresponding business actors and value objects according to Table VIII-15 and Table VIII-16. Figure VIII-11 (b) shows the instantiated value model.

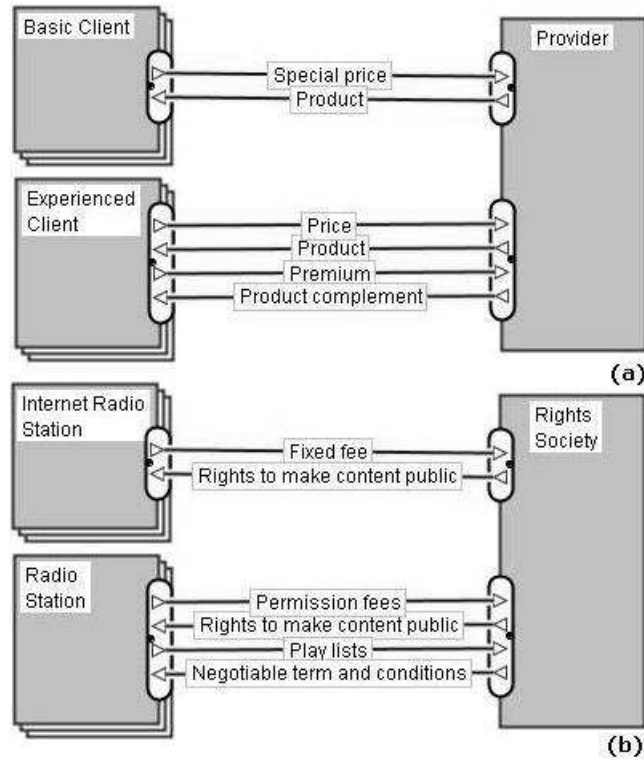


Figure VIII-11. Value models for the Market Segmentation (SEG) pattern: (a) – original value model, copied from Appendix E Library of Value Patterns, Pattern 5. Market Segmentation, on page 309 and (b) – instantiated value model

D. Registration (REG) pattern

The fourth pattern to instantiate is the Registration (REG) pattern which capability **REG.C1** matches the rights society’s goal **SOC.G17** (refer to Table VIII-7).

Step One: Map Roles Of Matching Goals And Capabilities. The **REG.C1** capability benefits the REG.Intermediary role (see Appendix E Library of Value Patterns, Pattern 7. Registration, on page 313.) Respectively, the **SOC.G17** goal belongs to the Rights Society (see Section 2.2.2.) These two facts determine that the REG.Intermediary role maps to Rights Society actor.

Table VIII-17 presents the correspondence of roles and business actors for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of actor as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining role in the pattern that needs to be assigned to a business actor is REG.Client. The information needed to resolve the mapping is available to the designer in the textual description of the

pattern and the matching goal. Particularly, the client of an intermediary in the context of the pattern is the content producer (the rights owner) in the real-life example context. Therefore, we map REG.Client to Content Producer (Rights Owner). Table VIII-17 summarizes the mappings.

Table VIII-17. Correspondence table for roles in the Registration (REG) pattern

<i>Roles in the pattern</i>	<i>Business actors in the example</i>
REG.Intermediary	Rights Society
REG.Client	Content Producer (Rights Owner)

Step Three: Modify Value Objects. In the case of the Advertising pattern, only one value object needs modification. Namely, Access to services needs to be changed to Rights protection, as to reflect the actual service offered in the real-life example. Respectively, the correspondence of value objects is captured in Table VIII-18, wherein the first column contains the value objects as in the pattern and the second column the values objects in the real-life example.

Table VIII-18. Correspondence table for value objects in the Registration (REG) pattern

<i>Value objects as in the pattern</i>	<i>Value objects as in the example</i>
REG.Access to services	Rights protection
REG.Identification information	Identification information

Value Model Transformation. Figure VIII-12 (a) depicts the value model of the pattern with the original roles and value objects. The instantiation of the pattern results in a value model in which the roles and the value objects are substituted with the corresponding business actors and value objects according to Table VIII-17 and Table VIII-18. Figure VIII-12 (b) shows the instantiated value model.

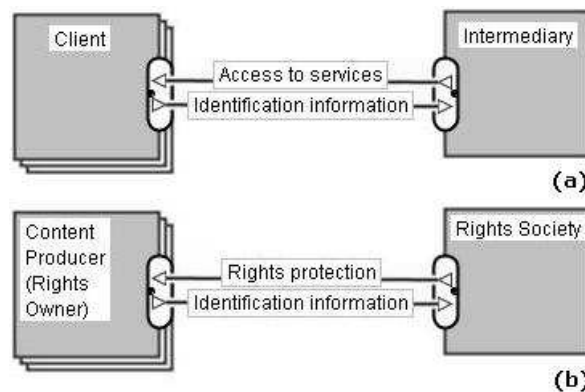


Figure VIII-12. Value models for the Registration (REG) pattern: (a) – original value model, copied from Appendix E Library of Value Patterns, Pattern 7. Registration, on page 313 and (b) – instantiated value model

E. Insourcing (INS) pattern

The fifth pattern to instantiate is the second match with the Insourcing (INS) pattern which capability **INS.2.C3** matches the rights society's goal **SOC.G19** (refer to Table VIII-7).

Step One: Map Roles Of Matching Goals And Capabilities. The **INS.2.C3** capability benefits the **INS.2.Business** role (see Appendix E Library of Value Patterns, Pattern 4. Insourcing, on page 307.) Respectively, the **SOC.G3** goal belongs to the **Rights Society** (see Section 2.2.2.) These two facts determine that the **INS.2.Business** role maps to the **Rights Society** actor.

Table VIII-19 presents the correspondence of roles and business actors for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of actor as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining roles in the pattern that need to be assigned to business actors are: **INS.2.Intermediary** and **INS.2.Client**. The information needed to resolve the mapping is available to the designer in the textual description of the pattern and the matching goal. Particularly, the client of an intermediary in the context of the pattern is the content producer receiving a service from a bank in the real-life example context. Therefore, we map **INS.2.Client** to **Content producer (Rights Owner)** and, respectively, **INS.2.Intermediary** to **Bank**. Table VIII-19 summarizes the mappings.

Table VIII-19. Correspondence table for roles in the Insourcing (INS) pattern

<i>Roles in the pattern</i>	<i>Business actors in the example</i>
INS.2.Intermediary	Bank
INS.2.Business	Rights Society
INS.2.Client	Content Producer (Rights Owner)

Step Three: Modify Value Objects. In the case of the second Insourcing pattern, three value objects need modification. First, **Product** needs to be changed to **Permission fees** to reflect the actual value object. Second, **Product delivery** needs to be changed to **Money transfer** to reflect the actual service being delivered. And third, **Money** needs to be changed to **Rights to make content public** for the same reason. The correspondence of value objects is captured in Table VIII-20, wherein the first column contains the value objects as in the pattern and, the second, the values objects as in the real-life example.

Table VIII-20. Correspondence table for value objects in the Insourcing (INS) pattern

<i>Value objects as in the pattern</i>	<i>Value objects as in the example</i>
INS.2.Product	Permission fees
INS.2.Product delivery	Money transfer
INS.2.Money	Rights to make content public
INS.2.Fee	Fee

Value Model Transformation. Figure VIII-13 (a) depicts the value model of the pattern with the original roles and value objects. The instantiation of the pattern results in a value model in which the roles and the value objects are substituted with the corresponding business actors and value objects according to Table VIII-19 and Table VIII-20. Figure VIII-13 (b) shows the instantiated value model.

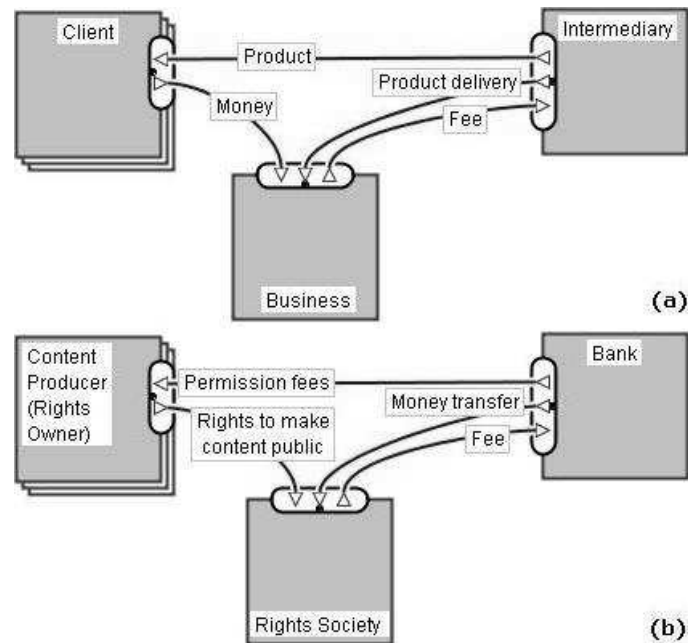


Figure VIII-13. Value models for the Insourcing (INS) pattern: (a) – original value model, copied from Appendix E Library of Value Patterns, Pattern 4. Insourcing, on page 307 and (b) – instantiated value model

4.2.2 Synthesize Patterns

To synthesize the selected patterns means to link their instantiated value models in a single value model. For clarity of presentation, we refer throughout this sub-section to the instantiated value models of patterns as patterns or pattern models. Correspondingly, the synthesized value model is referred to as the value model or the design.

Patterns are synthesized by taking them one by one indifferent of the order and adding them to the design. Patterns are added to the value model by merging actor and keeping value exchanges.

Below, we synthesize the patterns in our real-life example. After each added pattern, the intermediate value model is shown.

A. Adding the Advertising pattern to the value model

Initially, the design is empty: i.e., the value model contains no actors and no value exchanges. Adding the Advertising (ADV) pattern to the value model means that the

value model is the instantiated value model of the pattern as shown in Figure VIII-9 (b.) (This holds only for the first pattern added to the design because the design is empty.) Figure VIII-14 shows the intermediate value model after the first pattern.

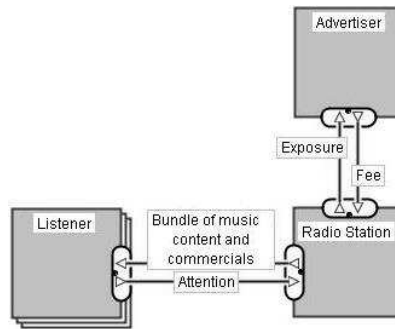


Figure VIII-14. Value model after the addition of the Advertising pattern

B. Adding the first Insourcing pattern to the value model

The second pattern to add to the value model is the first Insourcing (INS) pattern as shown in Figure VIII-10 (b.) The value model and the pattern do not contain identical actors. Therefore, the Insourcing pattern is added to the design but not linked with value exchanges with the rest of the value model. The intermediate result is shown in Figure VIII-15.

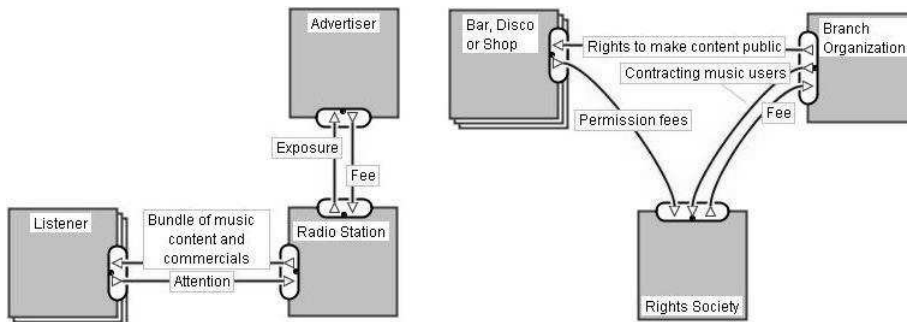


Figure VIII-15. Value model after the addition of the first Insourcing pattern

C. Adding the Market Segmentation pattern to the value model

The third pattern to add to the value model is the Market Segmentation (SEG) pattern as in Figure VIII-11 (b.) The value model and the pattern have two shared actors, namely: Radio Station and Rights Society. Despite the common name used to refer to the Radio Station actor in the value model and in the pattern, the actor differs in cardinality in the two models. In the value model, the Radio Station actor is just one business; whereas in the patterns, the Radio Station actor is modelled as a number of actors grouped together. Before we merge the two actors, we need to change the actor in the value model from a single actor to a group of actors. Then, we unify the Radio Station

actors to one which has all the value interfaces and exchanges previously associated with the separate actors. After merging the Radio Station actors, we merge the Rights Society actors. The result is shown in Figure VIII-16: the three patterns are linked with value exchanges.

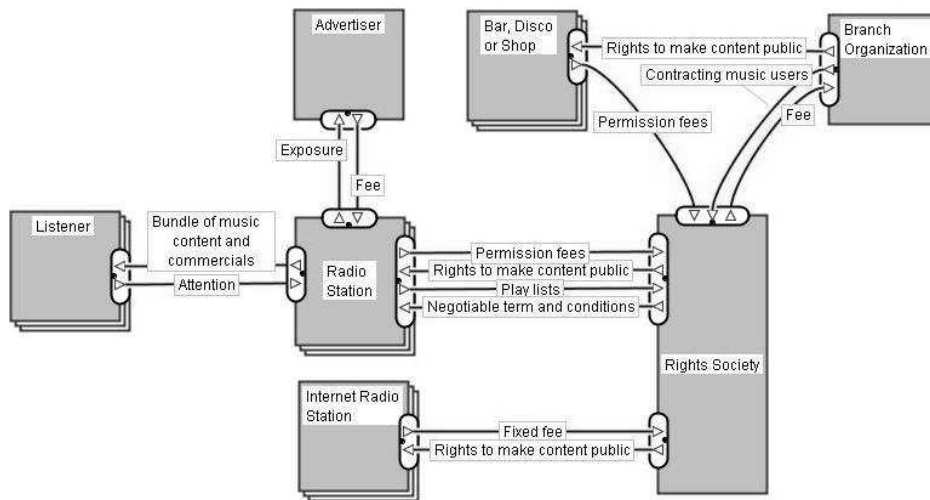


Figure VIII-16. Value model after the addition of the Market Segmentation pattern

D. Adding the Registration pattern to the value model

The fourth pattern to add to the value model is the Registration (REG) pattern as in Figure VIII-12 (b). Only one actor appears both in the value model and in the pattern, namely Rights Society. We merge the identical actors and add to the value model the value exchanges and actor present in the pattern. The resulting intermediate value model is depicted in the Figure VIII-17.

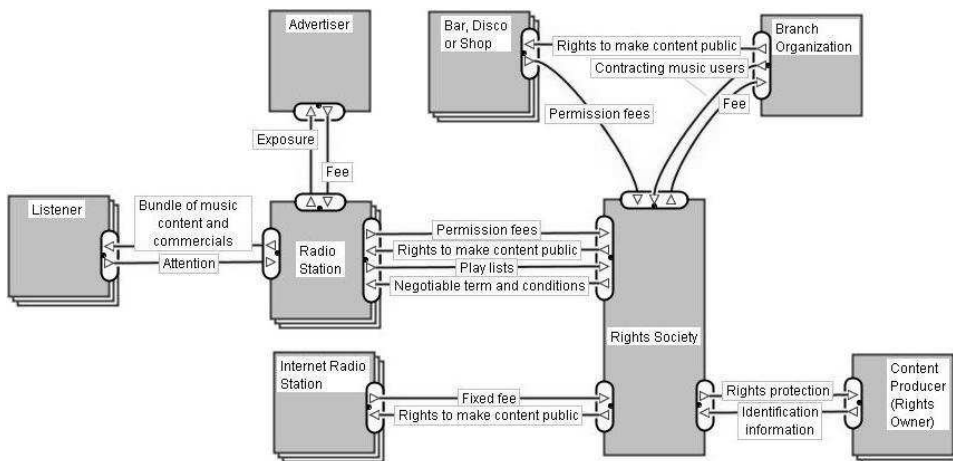


Figure VIII-17. Value model after the addition of the Registration pattern

E. Adding the second Insourcing pattern to the value model

The fifth and last pattern to add to the value model is the second Insourcing (INS) pattern, depicted in Figure VIII-13 (b.) The value model and the pattern have two shared actors, namely: Rights Society and Content Producer (Rights Owner). We merge these and keep all value exchanges associated with these. The result of the addition of the Insourcing pattern, which is also the result of the synthesis, is shown in Figure VIII-18.

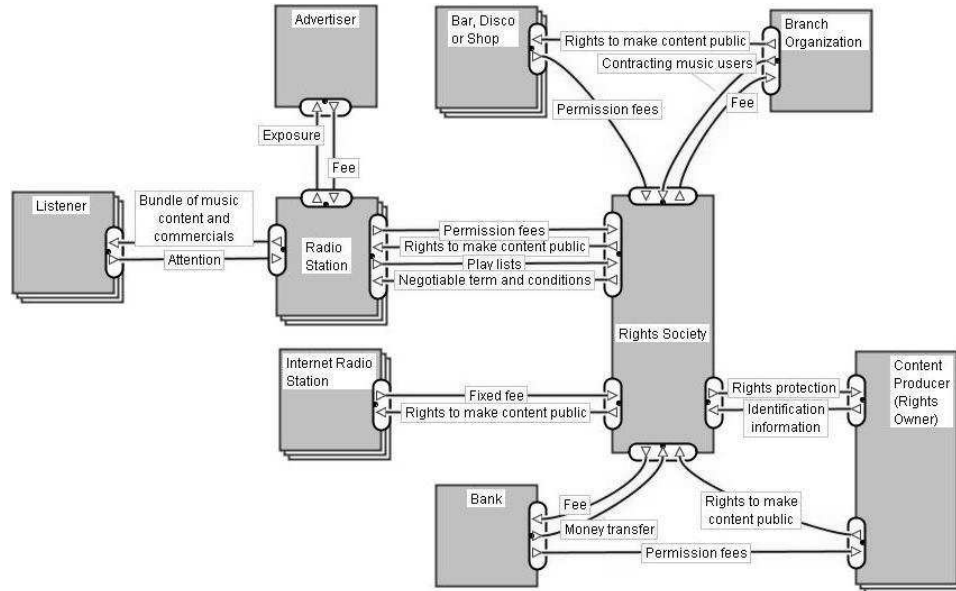


Figure VIII-18. Value model after the addition of the second Insourcing pattern

Figure VIII-18 presents only the synthesized model from matching patterns. It does not contain the model that achieves the goals with anticipated values.

5 Developing the Process Perspective

The business process perspective has the same structure as the economic value perspective. Thus, the starting position and the method of developing the process model are the same as for the e^3 -value model, see Section 4. Correspondingly, we begin with (i) a library of process patterns and (ii) a goal model of the desired system. We want to develop a process model of the system. Thus, we have to: one, select the relevant design patterns and, two, synthesize a process model out of the patterns.

The selection procedure (the same as for the development of the value perspective) is described in Chapter V Selection Procedure. In this section, we apply it and describe in detail the following steps. (The steps follow the selection procedure; though, the names or granularity may differ.)

- Generate alternative goal models

- Match goals and capabilities: build Matching table
- Find alternative solution capability models
- Propagate values in the alternative solution capability models
- Evaluate the alternative solution capability models: select the best alternative solution capability models
- Integrate each of the best alternative solution capability models with the capability models
- Propagate values and evaluate the best alternative solution capability models

The synthesis procedure is described in Chapter VI Synthesis of Value and Process Patterns. In this section, we apply it and describe in detail the following:

- Instantiate Patterns
 - Map the role of the matching capability to a business
 - Resolve the remaining roles in the pattern
 - Modify activities
 - Modify messages
- Synthesize Patterns
 - Determine swimlanes
 - Arrange activities
 - Add missing activities and messages
 - Link connectors

Below, we describe the development of the process perspective step by step. Starting with Section 5.2.1 until Section 5.2.7, we show the procedure of selecting relevant pattern. Further from Section 5.3.1 to Section 5.3.2, we present the synthesis of the selected patterns.

5.1 Extending the Goal Model

Developing the process perspective from the combined goal models of a radio station and a rights society (shown in Figure VIII-3) leads to a weak design because few process patterns match. Generally, the process patterns are finer grained than the value patterns. This means that several process patterns make one value pattern or that one value pattern can be implemented by several different process patterns. For details, see Chapter II Design Knowledge in Existing e-Business Models, Section 8 Relation between Value and Process Patterns. Due to the fact that the radio station goal model (see Figure VIII-1) and the rights society goal model (see Figure VIII-2) are refined to a level of detail that matches the value pattern capabilities, the goal models cannot select enough process patterns. The goal models need to be extended such that they contain leaf-goals which match the granularity of the capabilities in the process patterns. Below, we describe the newly added goals and how these are linked with the rest of the models.

5.1.1 Extending the Goal Model of Radio Station Music User

A. New Goals

We extend the goal model of a commercial radio station by refining goal **RAD.G3**. The goal requires music content and commercials to be bundled. This can be achieved by broadcasting a radio program that includes a number of commercial for which the

advertisers have paid the publication fee. This means, that goal **G3** decomposes into the following three sub-goals:

- **Ga: Broadcast radio program.**

We measure goal **Ga** with the ability of a radio to deliver the expected music content to listeners including advertisements in the radio program.

We operationalize goal **Ga** with:

- **va: Delivered bundled product and advertisement [Boolean].**

Variable v_a has value true if the radio has means of broadcasting a radio program which includes advertisements.

- **Gb: Bill advertisers for the broadcasted commercials.**

We measure goal **Gb** with the existence of billing mechanisms which ensure that advertising companies pay for the time slot they receive in the radio program.

We operationalize goal **Gb** with:

- **vb: Received payment [Boolean].**

Variable v_b has value true if the radio is able to issue invoices and collect the payment of these invoices.

- **Gc: Contract advertisers.**

We measure goal **Gc** with the ability of a radio station to efficiently contract advertisers. Therefore, we operationalize the goal with the money spent on a single contract.

- **vc: Average cost per contract [Euro].**

Variable v_c is the average cost to reach an agreement between a radio station and an advertiser indifferent on the number or duration of the advertisements.

An agreement can be reach in several ways. We model two and we represent them as alternative goals that cause the satisfaction of goal **Gc**. This means that goal **Gc** has two substitution sub-goals.

- **Gd: Negotiate with advertisers.**

The first option for an agreement process is iterative exchange of offers until both parties agree on the terms in the contract. Goal **Gd** aims at exactly such negotiation process. We measure the goal with the ability of a radio to handle such bargaining over contracts.

We operationalize goal **Gd** with a Boolean variable with measures the ability of disability to negotiate:

- **vd: Negotiable agreement [Boolean].**

Variable v_d has value true if the radio can execute a negotiation process of iterative exchange of contract offers.

- **Ge: Contract fixed offers.**

The second option for an agreement process is making a single offer and sticking to this. Goal **Ge** aims at business relationships in which advertisers have to take or leave the proposed contract. We measure the goal with the presence of such mechanism for contracting.

We operationalize goal **Ge** with:

- **ve: Fixed contract [Boolean].**

Variable v_e has value true if the radio contracts advertiser with offer that cannot be negotiated.

B. Extended Goal Model¹

The extended goal model of the radio station is shown in Figure VIII-19. The new goals are denoted with their abbreviation and letters from the Latin alphabet. To avoid further name conflicts as other goals may be abbreviated in a similar way, we uniquely identify the goals by prefixing their names with a code that refers to the goal model to which the goals belong. We refer to the extended goal model of a commercial radio station with the same code, namely **RAD**. This is sufficient to uniquely identify goals because the new goals have unique names within the model and the old goals do not change. In case the goal model is referred then we use: (i) extended goal model of a commercial radio station, (ii) extended goal model of a radio station, or (iii) extended **RAD**.

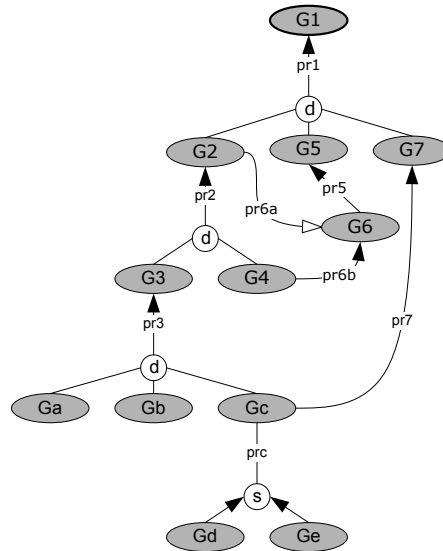


Figure VIII-19. Extended goal model of a commercial radio station

Table VIII-21 lists the operationalization of the newly modelled goals. The table also repeats old goals which are related to the new ones.

¹ The term extended goal model is not part of the framework and is used only in this chapter

Table VIII-21. Newly modelled goals of a commercial radio station and their operationalization

Goal code	Goal	
Variable code	Variable	Variable domain
G3	Bundle commercials with music content	
v ₃	Bundled product and advertisement	Boolean
G7	Minimize operation costs	
v ₇	Costs per year	Euro currency
Ga	Broadcast radio program	
v _a	Delivered bundled product and advertisement	Boolean
Gb	Bill advertisers for the broadcasted commercials	
v _b	Received payment	Boolean
Gc	Contract advertisers	
v _c	Average cost per contract	Euro currency
Gd	Negotiate with advertisers	
v _d	Negotiable agreement	Boolean
Ge	Contract fixed offers	
v _e	Fixed contract	Boolean

C. New Propagation Functions

The extended goal model of a commercial radio station in Figure VIII-19 contains 5 new goals which are related among each other and the rest of the goal model with 3 goal relationships: 1 decomposition relationship, 1 substitution and 1 dependency relationship. Each of the relationships is associated with a propagation function which captures the effect goals are causing one another. The number of propagation functions equals the number of head-goals: in the particular case, this is 3.

The first propagation function $pr3$ is associated with the decomposition relationship between goal **G3** (the head of the relationship) and goals **Ga**, **Gb**, and **Gc** (the tails of the relationship.) Propagation function $pr3$ calculates the value of goal variable v_3 out of the values of goal variables v_a , v_b , and v_c . In other words, the function calculates if a product and advertisement are bundled out of the delivery of a bundle to a listener, the payment, and costs of the contract with advertisers. The bundling of product and advertisement is evaluated with Boolean values. It is calculated with a logical AND operation on the variables of the sub-goals. Because variables v_c is not Boolean, we consider it to have value true if the costs are in the range of 50 and 1000 euros. Thus, the propagation function is:

$$\circ \quad v_3 = pr3(v_a, v_b, v_c) = v_a \text{ and } v_b \text{ and } (50 < v_c) \text{ and } (v_c < 1000)$$

The second propagation function $pr7$ is associated with the dependency relationship between goal **G7** (the head of the relationship) and goal **Gc** (the tail of the relationship.) Propagation function $pr7$ calculates the value of goal variable v_7 out of the value of goal variable v_c . In other words, the function calculates the effect of contracting costs on the operations costs of a rights society. The operations costs include more than the contracting costs. In Section 2.1.4, we model an anticipated value for the operations costs per year of 20 000 000 euros. Here, we assume that the costs of contracting advertisers

are 1% of the total costs. This means that the operations costs are equal to 19 800 000 plus the costs of contracting per year. Variable v_c measures the average costs per contract. To calculate yearly costs, we assume that there are on average 50 advertisements per month. Thus, the propagation function is:

- $v_7 = \text{pr7}(v_c) = 19\,800\,000 + 50 \cdot 12 \cdot v_c$

The third and last propagation function prc is associated with the substitution relationship between goal **Gc** (the head of the relationship) and goals **Gd** and **Ge** (the tails of the relationship.) Propagation function prc calculates the value of goal variable v_c out of the values of goal variables v_d and v_e . In other words, the function calculates the average cost per contract out of the type of contract. We assume that an agreement reached via a negotiation process is generally more expensive than a contract of take-it-or-leave-it type. We assign 100 and 500 euros per contracts achieved with giving a fix offer and negotiating, respectively. Thus, the propagation function is:

- $v_c = \text{prc}(v_d, v_e) = 500 \cdot v_d$, in case of goal **Gd**
- $v_c = \text{prc}(v_d, v_e) = 100 \cdot v_e$, in case of goal **Ge**

D. New Anticipated Values

There are no new anticipated values. Refer to Table VIII-3 for the anticipated values of the goals model before the extension.

5.1.2 Extending the Goal Model of Right Society

A. New Goals

We extend the goal model of a rights society by refining goals **SOC.G3**, **SOC.G12**, **SOC.G13**, and **SOC.G19** (see Figure VIII-2). The first goal to analyse is goal **SOC.G3**. It aims at outsourcing the contracting of music users such as bar, discos and shops via a third party, namely a branch organization. The goal decomposes into the following two sub-goals:

- **Ga: Contract outsourcing of rights management to branch organization.**

We measure goal **Ga** with the ability of a rights society to sign a contract which obliges a branch organization to manage the rights used by music users in the branch.

We operationalize goal **Ga** with:

- **va: Contracted outsourcing of a product [Boolean].**

Variable v_a has value true if the rights society has means of outsourcing its duty to contract collection of permission fees to a branch organization.

- **Gb: Bill the remaining music users for using rights.**

We measure goal **Gb** with the ability of a rights society to collect the due permission fees from music users which were contracted by a branch organization.

We operationalize goal **Gb** with:

- **vb: Received payment [Boolean].**

Variable v_b has value true if the rights society runs a business process which ensures the collection of permission fees.

The second goal to decompose is goal **SOC.G12**. It aims at contracting top-radio stations. Because the rights society needs the cooperation of the radio stations, the process of contracting is a negotiation. The rights society needs play-lists, which is a goal represented in the old goal model, in return for which it is willing to concede payment in terms. The remaining terms in the contract remain open for negotiation. Thus, goal **SOC.G12** decomposes to three goals only two of which are new.

- **Gc: Concede payment in terms.**

We measure goal **Gc** with the ability of a rights society to collect permission fees continuously through out the year.

We operationalize goal **Gc** with:

- **vc: Payment in terms [Boolean].**

Variable v_c has value true if the rights society can manage to bill music users periodically in the course of a year.

- **Gd: Negotiate the price per song-listener.**

We measure goal **Gd** with the ability of a rights society to bargain over, e.g., price per song-listener.

We operationalize goal **Gd** with:

- **vd: Negotiable agreement [Boolean].**

Variable v_d has value true if the rights society can execute a negotiation process of iterative exchange of contract offers until both parties agree on the terms in the contract.

The third goal to decompose is goal **SOC.G13**. It aims at contracting small radio stations. The rights society is the legal authority responsible for the collection of permission fees from rights to make content public. Thus, it has the power to dictate the contract terms to small radio stations. Therefore, goal **SOC.G13** decomposes to the following two goals:

- **Ge: Contract fixed offers.**

We measure goal **Ge** with the ability of a rights society to make radio stations accept fixed contracts.

We operationalize goal **Ge** with:

- **ve: Fixed contract [Boolean].**

Variable v_e has value true if the rights society can persuade music users to sign fixed contracts without using its power to legally enforce the contract.

- **Gf: Bill Internet radio station for using rights.**

We measure goal **Gf** with the ability of a rights society to collect due permission fees from small radio stations.

We operationalize goal **Gf** with:

- **vf: Received payment [Boolean].**

Variable v_f has value true if the rights society runs a business process which ensures the collection of permission fees.

The fourth and last goal to refine is goal **SOC.G19**. It aims at contracting a bank to do the money transfer from the rights society to the right owners. Goal **SOC.G19** decomposes to a single goal:

- **Gg: Contract outsourcing of money transfer.**

We measure goal **Gg** with the ability of a rights society to sign a contract which ensured the transfer of permission fees from the funds of the rights society to the registered rights owners.

We operationalize goal **Gg** with:

- **vg: Contracted outsourcing of a product [Boolean].**

Variable v_g has value true if the rights society has means of outsourcing its duty to transfer permission fees to a bank.

B. Extended Goal Model

The extended goal model of the rights society is shown in Figure VIII-20. The new goals are denoted with their abbreviation and letters from the Latin alphabet. To avoid further

name conflicts as other goals may be abbreviated in a similar way, we uniquely identify the goals by prefixing their names with a code that refers to the goal model to which the goals belong. We refer to the extended goal model of a commercial radio station with the same code, namely **SOC**. This is sufficient to uniquely identify goals because the new goals have unique names within the model and the old goals do not change. In case the goal model is referred then we use: (i) extended goal model of a rights society or (ii) extended **SOC**.

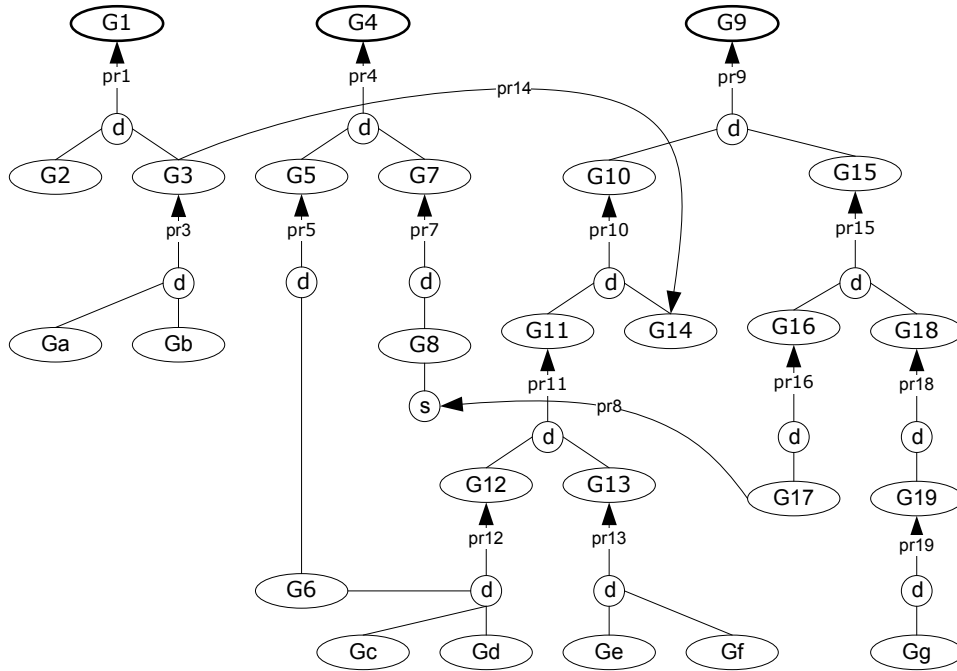


Figure VIII-20. Extended goal model of a rights society

Table VIII-22 lists the operationalization of the newly modelled goals. The table also repeats old goals which are related to the new ones.

Table VIII-22. Newly modelled goals of a rights society and their operationalization

Goal code	Goal	
Variable code	Variable	Variable domain
G3	Outsource the contracting of the remaining music users to branch organizations	
v ₃	Outsourced product delivery	Boolean
G6	Get play-lists from top radio stations	
v ₆	Song-listener coverage	Percentage
G12	Negotiate with big radio stations	
v ₁₂	Offered customizable contract	Boolean
G13	Offer fixed contract to small radio stations	

v ₁₃	Offered simple contract	Boolean
G19	Outsource payment	
v ₁₉	Outsourced product delivery	Boolean
Ga	Contract outsourcing of rights management to branch organization	
v _a	Contracted outsourcing of a product	Boolean
Gb	Bill the remaining music users for using rights	
v _b	Received payment	Boolean
Gc	Concede payment in terms	
v _c	Payment in terms	Boolean
Gd	Negotiate the price per song-listener	
v _d	Negotiable agreement	Boolean
Ge	Contract fixed offers	
v _e	Fixed contract	Boolean
Gf	Bill Internet radio station for using rights	
v _f	Received payment	Boolean
Gg	Contract outsourcing of money transfer	
v _g	Contracted outsourcing of a product	Boolean

C. New Propagation Functions

The extended goal model of a rights society in Figure VIII-20 contains 7 new goals which are related among each other and the rest of the goal model with 4 goal relationships, all of them decomposition relationships. Each of the relationships is associated with a propagation function which captures the effect goals are causing one another. The number of propagation functions equals the number of head-goals: in the particular case, this is 4.

The first propagation function **pr3** is associated with the decomposition relationship between goal **G3** (the head of the relationship) and goals **Ga** and **Gb** (the tails of the relationship.) Propagation function **pr3** calculates the value of goal variable **v₃** out of the values of goal variables **v_a** and **v_b**. In other words, the function calculates if the contracting of music users is outsourced to branch organizations out of the contracted outsourcing of branch organizations and, eventually, the delivered payments of music users. All variables are Boolean; thus, the value of **v₃** is logical AND of the values of **v_a** and **v_b**. The propagation function is:

$$\bullet \quad \mathbf{v}_3 = \mathbf{pr3}(\mathbf{v}_a, \mathbf{v}_b) = \mathbf{v}_a \text{ and } \mathbf{v}_b$$

The second propagation function **pr12** is associated with the decomposition relationship between goal **G12** (the head of the relationship) and goals **G6**, **Gc**, and **Gd** (the tails of the relationship.) Propagation function **pr12** calculates the value of goal variable **v₁₂** out of the values of goal variables **v₆**, **v_c**, and **v_d**. In other words, the function calculates if a negotiation with big radio stations is possible out of the statistics of broadcasted songs, the payment arrangements with radio stations, and the negotiated final agreement. To calculate the value of **v₁₂**, we need not convert the value of **v₆** from percentage to Boolean. We make the same assumption as before for propagation function **SOC.pr5**, namely that statistics over 85% of the listener is sufficient. Thus, the propagation function is:

$$\bullet \quad \mathbf{v}_{12} = \mathbf{pr12}(\mathbf{v}_6, \mathbf{v}_c, \mathbf{v}_d) = (\mathbf{85} < \mathbf{v}_6) \text{ and } \mathbf{v}_c \text{ and } \mathbf{v}_d$$

The third propagation function **pr13** is associated with the decomposition relationship between goal **G13** (the head of the relationship) and goals **Ge**, and **Gf** (the tails of the relationship.) Propagation function **pr13** calculates the value of goal variable v_{13} out of the values of goal variables v_e , and v_f . In other words, the function calculates if it is possible to effectively contract small radio stations out of offered fixed contracts and, eventually, received payments for used rights. It is calculated with a logical AND operation on the variables of the sub-goals. Thus, the propagation function is:

- o $v_{13} = \mathbf{pr13}(v_e, v_f) = v_e \text{ and } v_f$

The fourth and last propagation function **pr19** is associated with the decomposition relationship between goal **G19** (the head of the relationship) and goal **Gg** (the tail of the relationship.) Propagation function **pr19** calculates the value of goal variable v_{19} out of the value of goal variable v_g . In other words, the function calculates the outsourcing of a payment service out of the possibility to contract a bank to do the transfer. The assumption is that if there is a bank to accept to do the permission fees transfer then the outsourcing is achieved. Thus, the propagation function is:

- o $v_{19} = \mathbf{pr19}(v_g) = v_g$

D. New Anticipated Values

There are no new anticipated values. Refer to Table VIII-5 for the anticipated values of the goals model before the extension.

5.1.3 Extended Goal Model of the Business Network

The new goal model of the business network under design is the extended goal model of a radio station (see Figure VIII-19) and the extended goal model of a rights society (see Figure VIII-20) combined. Further, we refer to the new goal model as the extended goal model of the system. Figure VIII-21 depicts the extended goal model. The goals are labelled with their names without the prefix denoting to which goal model they belong. The grey goals belong to the goal model of a radio station; the white goals belong to the goal model of a rights society.

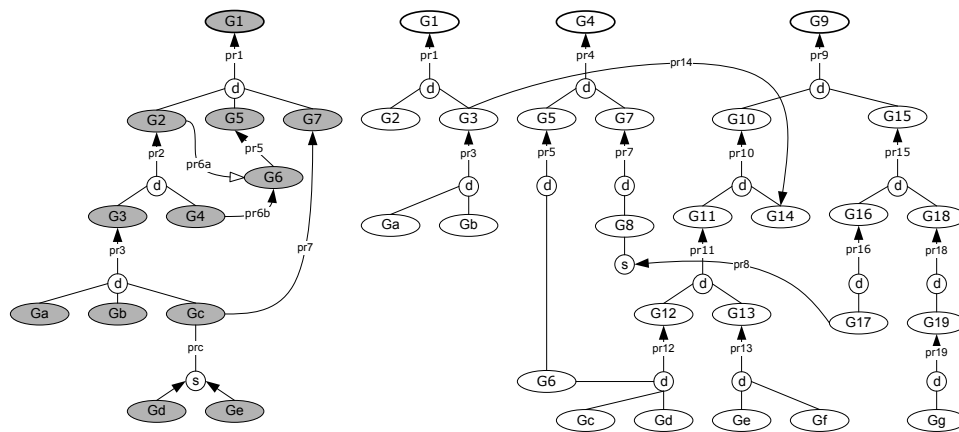


Figure VIII-21. Extended goal model of the business network under design

5.1.4 Implications for the Value Perspective

Changing the goal models requires redevelopment of the value perspective. Nevertheless, we do not do this because the particular extensions we make do not change the propagated values to the top-level goals. That is, the variable values coming from capabilities of value patterns or being anticipated always overwrite any potential values coming from the additional goals.

The design method does not prescribe iterative development of value and process perspectives. Iterations are not restricted but not supported by the method. We use this flexibility to easily present the real-life example.

5.2 Selection of Process Patterns

5.2.1 Generate Alternative Goal Models

The goal model contains 38 goals: 12 belong to the goal model of a radio station and 26 belong to the goal model of a rights society. Following the algorithm for generation of alternative goal model described in Chapter V Selection Procedure Section 4.1, we create 68 818 alternative goal models¹. The alternative goal models look like the alternative goal models in the value perspective shown in Section 4.1.1 in Table VIII-6. Here, we do not show a sample of the alternative goal models.

5.2.2 Match Goals and Capabilities: Build Matching Table

Each of the 38 goals in the goal model is checked for a match with a capability from the capability models in the library of process (Appendix G Library of Process Patterns). The result is that seven patterns have capability model with top-level capabilities that match goals from the goal model. The number of matches is twelve as some of the capabilities match more than one goal. The result is presented in Table VIII-23 which is the Matching table. The columns list the goals with their full names; whereas, the rows contain matching capabilities. The table presents which capability matches which goal by means of cells marked with an 'x'.

Table VIII-23. The Matching table for the extended goal model

	RAD.G1	RAD.G2	RAD.G3	RAD.G4	RAD.G5	RAD.G6	RAD.G7	RAD.Ga	RAD.Gb	RAD.Gc	RAD.Gd	RAD.Ge	SOC.G1	SOC.G2	SOC.G3	SOC.G4	SOC.G5	SOC.G6	SOC.G7
ADV.C2								x											
OCO.C1											x								
OUT.C2																			
REG.C1																			
PAY.C1									x										
PIT.C2																			
TOL.C1												x							

¹ The generation of alternative goal models is done with an Excel electronic table

	SOC.G8	SOC.G9	SOC.G10	SOC.G11	SOC.G12	SOC.G13	SOC.G14	SOC.G15	SOC.G16	SOC.G17	SOC.G18	SOC.G19	SOC.Ga	SOC.Gb	SOC.Gc	SOC.Gd	SOC.Ge	SOC.Gf	SOC.Gg
ADV.C2																			
OCO.C1																x			
OUT.C2													x						x
REG.C1										x									
PAY.C1														x				x	
PIT.C2															x				
TOL.C1																	x		

The result of matching goals and capabilities is that the matched goals receive values from the capabilities. Goal variables get values also during the goal modelling when anticipated values are assigned to some goals. Table VIII-24 presents the known values of variables in the extended goal model. The table contains values received from the capabilities, shown in bold font, and the anticipated values taken from Table VIII-3 and Table VIII-5.

To simplify the description of the development of the process perspective, we disregard the anticipated values for variables the value of which we can calculate. This means that the values in strikethrough font in Table VIII-24 are assumed unknown. The consequence is a reduction of the number of alternative solution capability model to evaluate. This, however, does not invalidate our approach; the remaining design freedom is sufficient to demonstrate the applicability of our method of development.

Table VIII-24. Variable values

goal	value	goal	Value
RAD.G1		SOC.G8	
RAD.G2		SOC.G9	300-000
RAD.G3		SOC.G10	200-000
RAD.G4	12	SOC.G11	
RAD.G5		SOC.G12	
RAD.G6		SOC.G13	
RAD.G7	20-000-000	SOC.G14	100-000
RAD.Ga	true	SOC.G15	37-000
RAD.Gb	true	SOC.G16	30-000
RAD.Gc		SOC.G17	85
RAD.Gd	true	SOC.G18	10-000
RAD.Ge	true	SOC.G19	
SOC.G1		SOC.Ga	true
SOC.G2	560 640 000 000	SOC.Gb	true
SOC.G3		SOC.Gc	true
SOC.G4		SOC.Gd	true
SOC.G5		SOC.Ge	true
SOC.G6	87	SOC.Gf	true
SOC.G7		SOC.Gg	true

5.2.3 Find Alternative Solution Capability Models

The viable alternative goal models are those that contain only goals with known values: known either due to a matching capability or the value is anticipated. We refer to such alternative goal models as alternative solution capability models. From the 68 818 alternative goal models only 2 constitute alternative solution capability models¹. Table VIII-25 presents these. The columns present the goals with their full names; whereas, the rows list the alternative solution capability models. The cells marked with ‘x’ determine which goal participates in which goal model.

Table VIII-25. Alternative solution capability models

	RAD.G1	RAD.G2	RAD.G3	RAD.G4	RAD.G5	RAD.G6	RAD.G7	RAD.Ga	RAD.Gb	RAD.Gc	RAD.Gd	RAD.Ge	SOC.G1	SOC.G2	SOC.G3	SOC.G4	SOC.G5	SOC.G6	SOC.G7
1				x			x	x	x		x			x					x
2				x			x	x	x			x		x					x

	SOC.G8	SOC.G9	SOC.G10	SOC.G11	SOC.G12	SOC.G13	SOC.G14	SOC.G15	SOC.G16	SOC.G17	SOC.G18	SOC.G19	SOC.Ga	SOC.Gb	SOC.Gc	SOC.Gd	SOC.Ge	SOC.Gf	SOC.Gg
1										x			x	x	x	x	x	x	x
2										x			x	x	x	x	x	x	x

5.2.4 Propagate Values in the Alternative Solution Capability Models

To enable the evaluation of potential design determined by an alternative solution capability model, the known values in the goal model have to be propagated to the top-level goals. This is done according to the description in Chapter IV Goal-modelling Section 5 Satisfaction Calculation.

The goal model contains 4 top-level goals:

- **RAD.G1;**
- **SOC.G1;**
- **SOC.G4;** and
- **SOC.G9.**

For each of the 2 alternative solution capability models, the values of the top-level goals are determined. Table VIII-26 presents the calculated values². The columns of the table contain the values for the particular top-level goal; whereas, the rows list the alternative solution capability models.

¹ The alternative goal models are managed in an Excel electronic table

² The propagation of values is calculated in a goal model represented in an Excel electronic table

Table VIII-26. Values for the top-level goals

	RAD.G1 [euros]	SOC.G1 [euros]	SOC.G4 [euros]	SOC.G9 [euros]
1	44 820 457.43	39 409 050	27 750 000	153 000
2	45 060 457,43	39 409 050	27 750 000	153 000

5.2.5 Evaluate the Alternative Solution Capability Models: Select the Best Alternative Solution Capability Models

According to our selection procedure (see Chapter V Selection Procedure, Section 7,) the alternative solution capability models are rated on the basis of the values of the top-level goals. In general, we use a multiple-criteria decision making method because each model is evaluated with several values. In the particular case at hand, the values of the top-level goals **SOC.G1**, **SOC.G4**, and **SOC.G9** are the same for every alternative solution capability model, which allows us to rate the models based on only the **RAD.G1** top-level goal.

Goal **RAD.G1** aims at making as much money as possible (see Section 2.1.1). This means that the goal is closer to satisfaction if the value of the variables is a bigger number. Figure VIII-22 presents the values for the **RAD.G1** variable per each alternative solution capability model. The alternative solution capability model number 1 is further from satisfaction compared to alternative solution capability model number 2.

The purpose of evaluating the alternative solution capability models is to select the three best satisfying the top-level goals. In our case, we select both models.

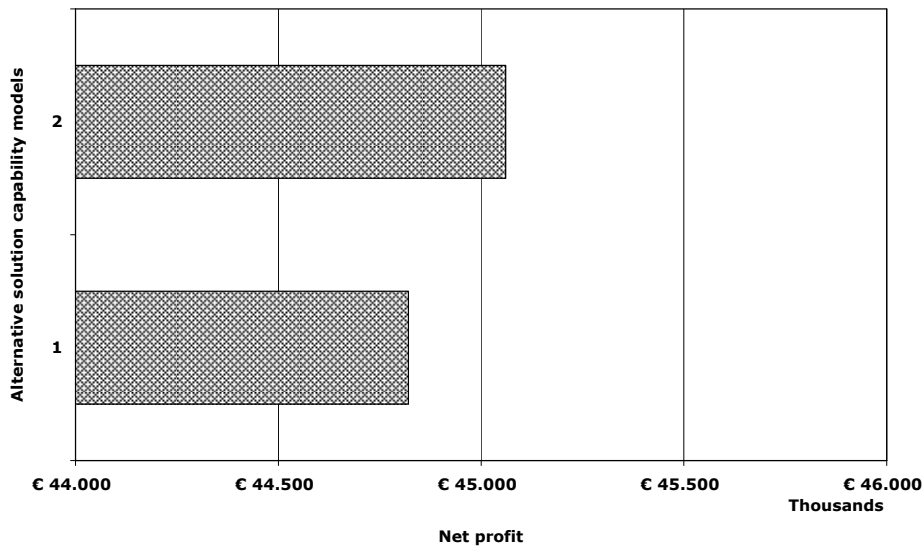


Figure VIII-22. Values for the RAD.G1 top-level goal

5.2.6 Integrate Each of the Best Alternative Solution Capability Models with the Capability Models

Selecting a design pattern as relevant for a particular system under development means that it is taken with all of its capabilities. This requires integrating the alternative solution capability models and the capability models (see Chapter V Selection Procedure, Section 6.)

In our case, there are seven patterns that have top-level capabilities matching with goals. (Refer to Table VIII-23, which lists the matching capabilities, respectively capability models, and goals.) Figure VIII-23 shows the capability models of the patterns: (a) – capability model of the Advertising (ADV) value pattern; (b) – capability model of the Insourcing (INS) value pattern; (c) – capability model of the Market segmentation (SEG) value pattern; and (d) – capability model of the Registration (REG) value pattern. The capabilities models are taken from Appendix G Library of Process Patterns, where they are fully described.

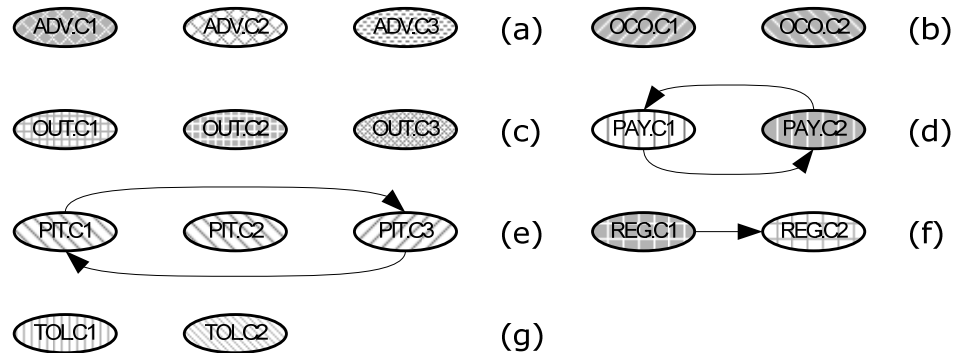


Figure VIII-23. Capability models: (a) – capability model of the Advertising (ADV) process pattern; (b) – capability model of the Offer counter offer (OCO) process pattern; (c) – capability model of the Outsourcing (OUT) process pattern; (d) – capability model of the Payment (PAY) process pattern; (e) – capability model of the Payment in terms (PIT) process pattern; (f) – capability model of the Registration (REG) process pattern; and (g) – capability model of the Take it or leave it (TOL) process pattern

Each of the best alternative solution capability models needs to be integrated with the seven patterns from Figure VIII-23. According to Chapter V Selection Procedure, Section 6, the integration step includes replacing goals with matching capabilities and adding relationship between capabilities and goals.

We begin with alternative solution capability model number 1. We replace goals **RAD.Ga**, **RAD.Gb**, **RAD.Gd**, **SOC.G17**, **SOC.Ga**, **SOC.Gb**, **SOC.Gc**, **SOC.Gd**, **SOC.Ge**, **SOC.Gf**, and **SOC.Gg** with capabilities **ADV.C2**, **PAY.C1**, **OCO.C1**, **REG.C1**, **OUT.C2**, **PAY.C1**, **PIT.C2**, **OCO.C1**, **TOL.C1**, **PAY.C1**, and **OUT.C2**. Replacing goals with capabilities adds the complete capability models to the goals model. This is illustrated in Figure VIII-24 by adding the capabilities from one capability model in a vertical line such that the original position of the replaced goal is kept.

We finalize the integration by adding the relationships between capabilities and goals. In the case of alternative solution capability model number 1, the matching capabilities do not interact with the goals. This leads to no added relationships.

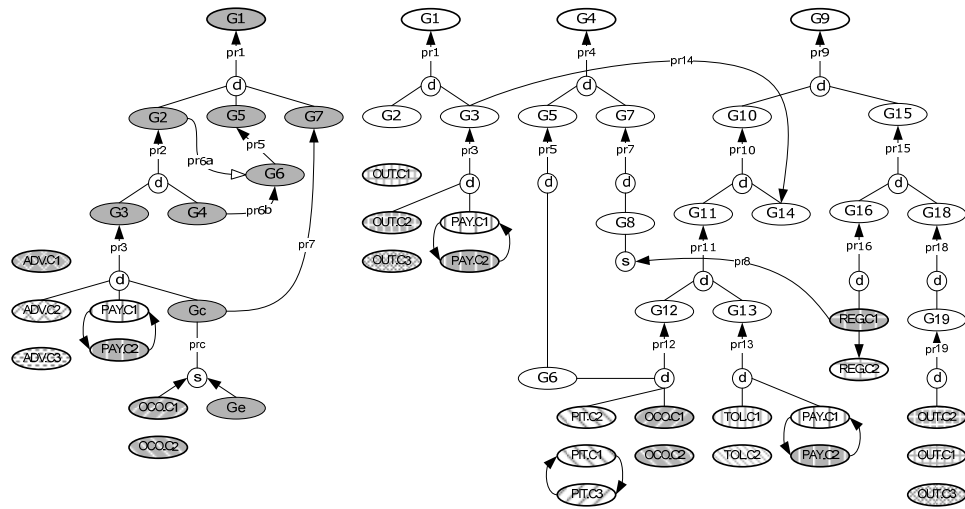


Figure VIII-24. Alternative solution capability model number 1 integrated with capability models ADV, PAY, OCO, REG, OUT, PIT, and TOL

The second alternative solution capability model to integrate is number 2. We replace goals **RAD.Ga**, **RAD.Gb**, **RAD.Ge**, **SOC.G17**, **SOC.Ga**, **SOC.Gb**, **SOC.Gc**, **SOC.Gd**, **SOC.Ge**, **SOC.Gf**, and **SOC.Gg** with capabilities **ADV.C2**, **PAY.C1**, **TOL.C1**, **REG.C1**, **OUT.C2**, **PAY.C1**, **PIT.C2**, **OCO.C1**, **TOL.C1**, **PAY.C1**, and **OUT.C2**. Similarly to Figure VIII-24, Figure VIII-25 adds the capabilities from one capability model in a vertical line such that the original position of the replaced goal is kept.

The last step of the integration (see Chapter V Selection Procedure, Section 6) is the adding of relationships between capability and goals. In the case of alternative solution capability model number 1, the matching capabilities do not interact with the goals. This leads to no added relationships.

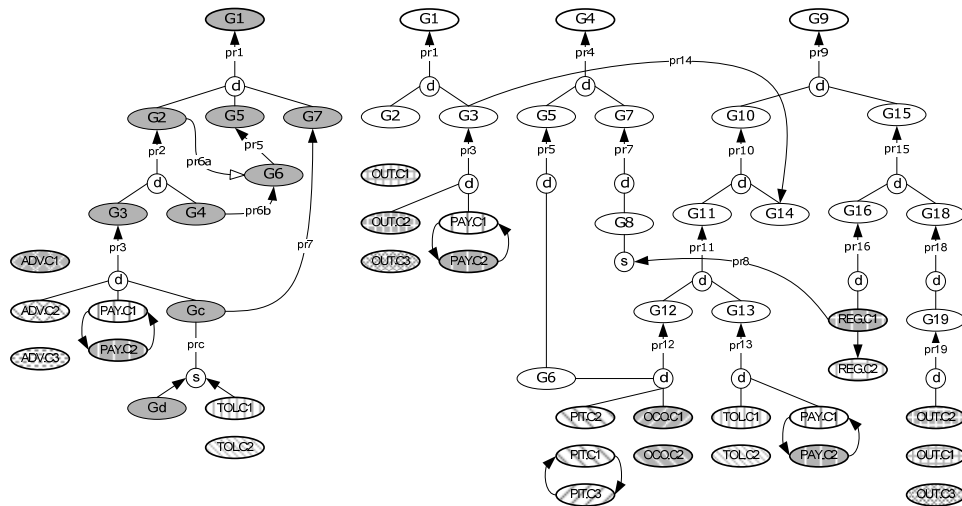


Figure VIII-25. Alternative solution capability model number 2 integrated with capability models ADV, PAY, TOL, REG, OUT, PIT, and OCO

5.2.7 Propagate Values and Evaluate the Best Alternative Solution Capability Models

Due to the fact that no new relationships were added to any of the best alternative solution capability models, the propagation of values throughout the models leads to the same values for the top-level goals. Consequently, the evaluation leads to the same conclusion: alternative solution capability model number 2 has the best score and therefore is the one considered for further development.

5.3 Synthesis of Process Patterns

Synthesis links the Activity diagram fragments of the selected patterns into one process model. The starting position is eleven patterns. These are: Advertising, Offer counter offer, two times Outsourcing, three times Payment, Payment in terms, Registration, and two times Take it or leave it.

The names of swimlanes, activities and messages¹ used in the process model of a particular pattern may occur in the process model of other patterns. To avoid this conflict of name, we need to extend the name such that we uniquely identify the swimlanes, activities and objects. Thus, each extended name contains the code of the pattern to which it belongs, and a number identifying patterns if more than one of a kind are matched with goals. An extended name has the following Backus-Naur form:

```
<extended name> ::= <pattern code> "." [<pattern number>
"."] <role or value object name>
```

¹ To avoid confusion of terminology as the e3-value and Activity diagram notations use the concept object, we refer to the objects and object flows in an Activity diagram as messages and message exchanges, respectively

The extension of names is the same as in Section 4.2. For an example, look in Section 4.2.

5.3.1 Instantiate Patterns

Alternative solution capability model number 2 matches 11 patterns. Each one of these needs to be instantiated such that the roles, activities, and messages correspond to the swimlanes, activities, and messages of real-life example at hand. The instantiation procedure includes the following three steps:

1. Map roles from the pattern with swimlanes from the matching goals.
2. Resolve the remaining roles in the pattern.
3. Modify activities.
4. Modify messages.

Below, we select the **Outsourcing** pattern that matches goal **SOC.Gg** to show an example of how the instantiation works. The instantiation of the remaining patterns is summarized in Annex - Instantiation of Process Patterns on page 250.

A. Outsourcing (OUT) pattern

We demonstrate the instantiation process with the second Outsourcing (OUT) pattern which capability **OUT.2.C2** matches the rights society's goal **SOC.Gg** (refer to Table VIII-23).

Step One: Map Roles From The Pattern With Swimlanes From The Matching Goals. The **OUT.2.C2** capability benefits the OUT.2.Business role (see Appendix G Library of Process Patterns, Pattern 6. Outsourcing, on page 363.) Respectively, the **SOC.Gg** goal belongs to the Rights Society (see Section 5.1.2.) These two facts determine that the OUT.2.Business role maps to the Radio Station swimlane.

Table VIII-27 presents the correspondence of roles and swimlanes for the pattern. The first column lists the roles as named in the pattern; the second column contains the names of swimlanes as in the real-life example. The rows in bold font are the mappings resulting from the matching capabilities and goals.

Step Two: Resolve The Remaining Roles In The Pattern. The remaining roles in the pattern that need to be assigned to swimlanes are: OUT.Intermediary and OUT.Client. The information needed to resolve the mapping is available to the designer in the textual description of the pattern and the matching goal. Particularly, the intermediary in the context of the pattern is a bank in the real-life example context. Therefore, we map OUT.Intermediary to Bank. Following the same line of reasoning, we map the OUT.Client to Content Producer. Table VIII-27 summarizes the mappings.

Table VIII-27. Correspondence table for roles and swimlanes in the second Outsourcing (OUT) pattern

<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
OUT.2.Intermediary	Bank
OUT.2.Business	Rights Society
OUT.2.Client	Content Producer

Step Three: Modify Activities. Similarly to the roles, the activities from the pattern have to be modified such that they match the activities in the real-life example. In the case of the Outsourcing pattern, several activities need modification. For example, **Request outsourcing** needs to be changed to **Request money transfer service** to reflect the actual requested service. The same holds for pattern activities **Make an offer**, **Deliver product**, and **Bill**, which change to **Respond to request**, **Transfer permission fees**, and **Charge for transactions**. The correspondence of activities is captured in Table VIII-28, wherein the first column contains the activities as in the pattern and the second column the activities as in the real-life example.

Table VIII-28. Correspondence table for activities in the second Outsourcing (OUT) pattern

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
OUT.2.Request outsourcing	Request money transfer service
OUT.2.Evaluate offer	Evaluate offer
OUT.2.Reject offer	Reject offer
OUT.2.Take offer	Take offer
OUT.2.Pay dues	Pay dues
OUT.2.Make an offer	Respond to request
OUT.2.Deliver product	Transfer permission fees
OUT.2.Bill	Charge for transactions
OUT.2.Collect dues	Collect dues

Step Four: Modify Messages. Similarly to the roles and activities, the messages from the pattern have to be modified such that they match the messages in the real-life example. In the case of the Outsourcing pattern, two messages need modification. Namely, **product** needs to be changed to **permission fees** to reflect the actual products being delivered; and **request** needs to be changed to **recipients** to reflect the actual data sent as a request. Respectively, the correspondence of messages is captured in Table VIII-29, wherein the first column contains the messages as in the pattern and the second column the messages as in the real-life example.

Table VIII-29. Correspondence table for messages in the second Outsourcing (OUT) pattern

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
OUT.2.product	permission fees
OUT.2.request	recipients
OUT.2.offer	offer
OUT.2.rejection	rejection
OUT.2.contract	contract
OUT.2.invoice	invoice
OUT.2.fee	fee

Activity Diagram Transformation. Figure VIII-26 (a) depicts the activity diagram of the pattern with the original roles, activities and messages. The instantiation of the pattern results in an activity diagram in which the roles, activities and messages are substituted

with the corresponding swimlanes, activities and messages according to Table VIII-27, Table VIII-28, and Table VIII-29. Figure VIII-26 (b) shows the instantiated value model.

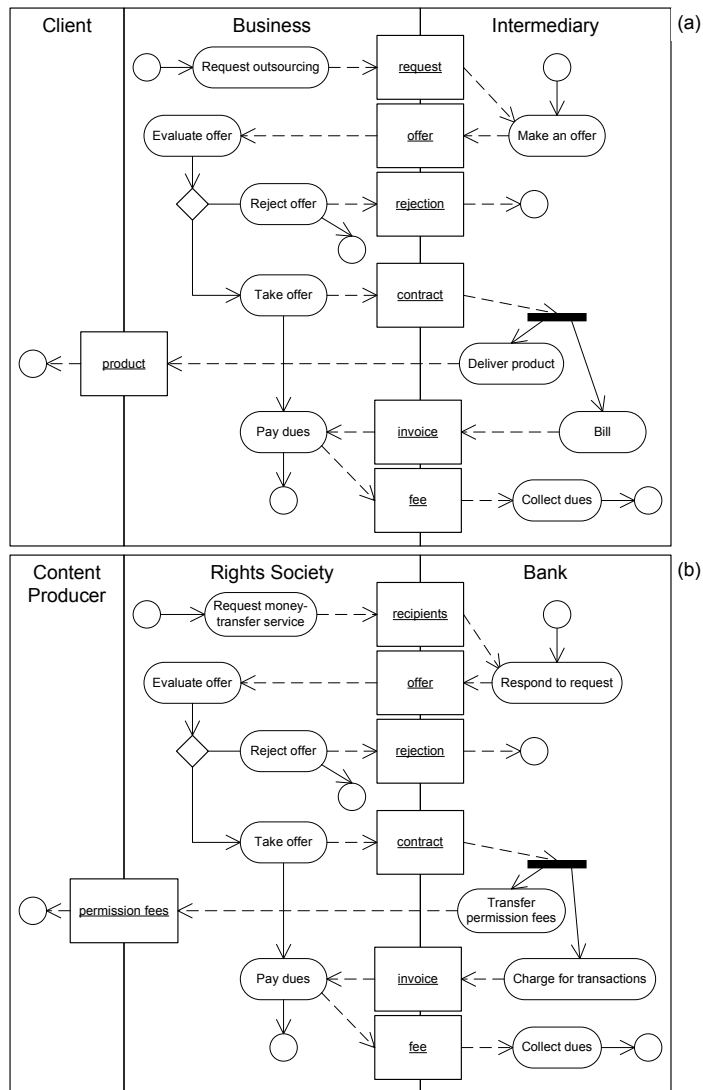


Figure VIII-26. Activity diagrams for the Outsourcing (OUT) pattern: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 6. Outsourcing, on page 363 and (b) – instantiated activity diagram

5.3.2 Synthesize Patterns

To synthesize the selected patterns means to link their instantiated process models in a single process model. Patterns are linked into a process model by merging swimlanes and

keeping activities and message exchanges. The synthesis procedure is described in detail in Chapter VI Synthesis of Value and Process Patterns.

The process model is too big to fit on one page. Therefore, we split it into 5 sub-processes¹. We do the synthesis step-by-step for the first sub-process which we name Advertising. For short references to the model we abbreviate the name with A. The remaining sub-processes are shown after the synthesis in Annex - Synthesized Sub-Processes on page 261. Their names and abbreviations are: Contracting branch organization (CBO), Contracting Internet radio station (CIRS), Contracting radio station (CRS), and Outsourcing payment (OP).

For clarity of presentation, we refer throughout this sub-section to the instantiated process models of patterns as patterns or pattern models. Correspondingly, the synthesized process model is referred to as the process model or the design.

A. Choose Patterns for the Advertising Sub-Process Model

Before we proceed with the synthesis steps as outlined in the beginning of Section 5, we need to choose the patterns taking part in the Advertising sub-process. We decide to include all patterns matching goals directly or indirectly affecting goal **RAD.G3** in the goal model of a Radio Station. This means that we consider for synthesis patterns Advertising (ADV), first Payment (PAY.1) and first Take it or leave it (TOL.1) as they match goals **RAD.Ga**, **RAD.Gb**, and **RAD.Ge**, respectively (see Table VIII-23 on page 217.)

B. Determine Swimlanes

Initially, the design contains only of a start symbol and a parallel split. From the pattern taking part in the particular sub-process, namely Advertising (ADV), first Payment (PAY.1) and first Take it or leave it (TOL.1), we determine the number of distinctive swimlanes to be 3 with names Advertiser, Radio Station and Listener. We add one connector for each swimlane and link this to the parallel split. The result is in Figure VIII-27.

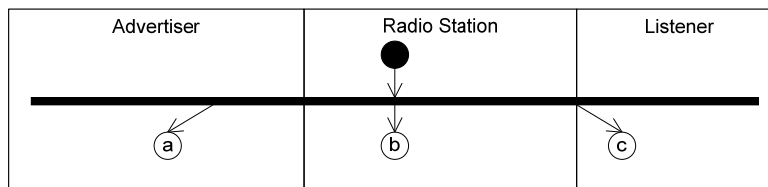


Figure VIII-27. Intermediate result for the Advertising sub-process after determining the swimlanes

C. Arrange Activities

In this step of the synthesis procedure, we add all pattern process fragments in to the process model. We ensure that all activities are placed in the appropriate swimlanes. Further, we assign a unique identifier to each connection, in this case the Latin letters form **d** to **s**. The result is in Figure VIII-28.

¹ See Chapter VI Synthesis of Value and Process Patterns for details on the notion of sub-processes

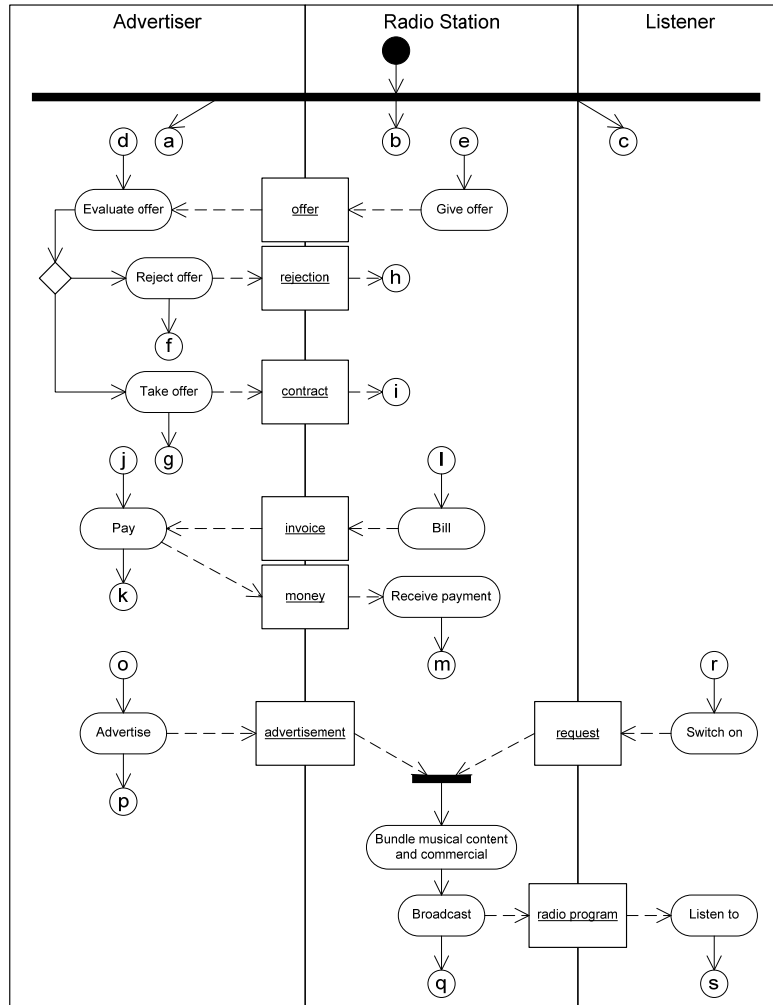


Figure VIII-28. Intermediate result for the Advertising sub-process after arranging activities

D. Add Missing Activities and Messages

In this step, we add activities and messages which are needed to make a complete model but are not available in the library of process patterns.

We add a fragment with connectors **t**, **u**, **v**, and **w** to represent in the process model the fact the advertiser is requesting an offer for a broadcast time. The fragment contains two activities, one in the Advertiser and one in the Radio Station swimlanes, coordinated with a message representing a request for quotation. See at the top of Figure VIII-29.

Another fragment that we add is the one with a connector **x**. It is an activity which concludes the interaction with the advertising company and terminates the process in the Radio Station swimlane.

E. Link Connectors

In this step, we order the pattern. First the Take it or leave it pattern is executed, followed by the Payment. Last is the Advertising pattern. To achieve this sequence, and also to include the additional fragment, we form the following pairs of connectors: **a – t**, **b – v**, **c – r**, **d – u**, **e – w**, **f – y**, **g – j**, **h – x**, **i – l**, **k – o**, **m – z**, **p – A**, **q – B**, and **s – C**.

We link the paired connector; remove the connectors; and arrive at the final model shown in Figure VIII-30. The grey areas in the figure mark the activities and messages and their pattern of origin.

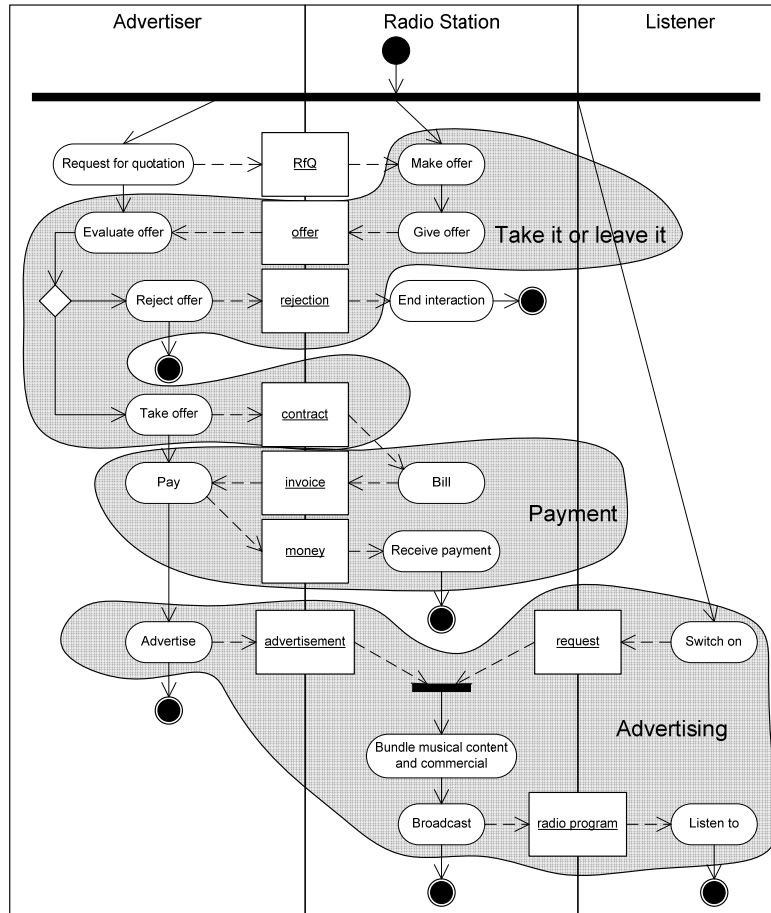


Figure VIII-30. Synthesized Advertising sub-process

The remaining Contracting branch organization (CBO), Contracting Internet radio station (CIRS), Contracting radio station (CRS), and Outsourcing payment (OP) sub-processes are shown in Annex - Synthesized Sub-Processes on page 261.

6 Checking Consistency

To be consistent, the value and process specifications have to have equivalent reduced models. We check consistency by transforming the value model of our real-life example, shown in Figure VIII-18, according to the transformation procedure described in Chapter VII Consistency between Value and Process Models Section 6.2 Transformation from an *e3-value* Model to Reduced Models on page 152 and, respectively, we transform the activity diagrams in Annex - Synthesized Sub-Processes on page 261, comprising the process model, according to the transformation procedure described in Chapter VII Consistency between Value and Process Models Section 6.3 Transformation from an Activity Diagram to Reduced Models on page 153. Consequently, we compare the resulting reduced models.

6.1 Transform the Value Model to Reduced Models

The value specification consists of one *e³-value* model. This resulted from the synthesis procedure described in Section 4. Figure VIII-31 depicts the value model. It is a copy of Figure VIII-18 such that the value objects are numbered to be uniquely identified.

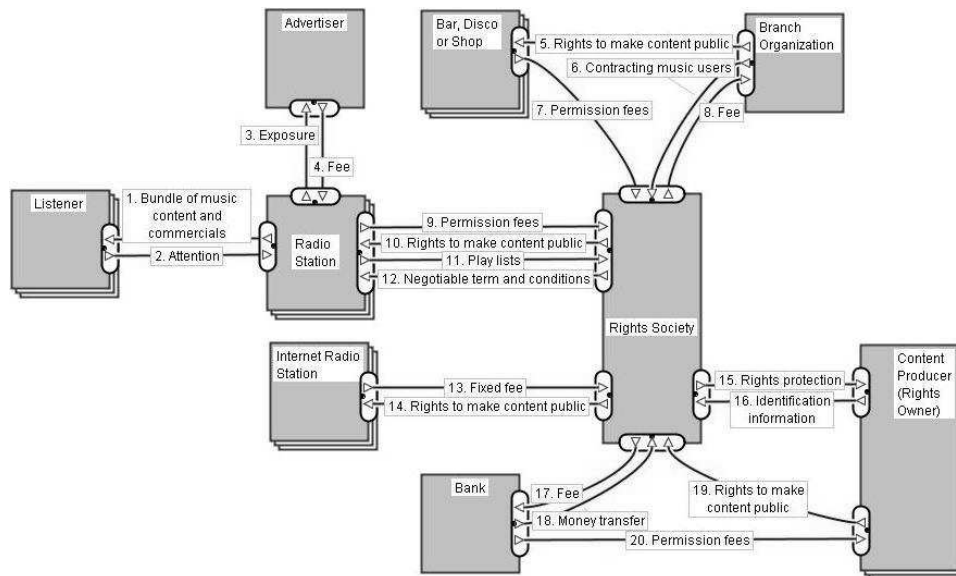


Figure VIII-31. The *e³-value* model constituting the value specification. Copy of Figure VIII-18 with included numbering of the value objects

The transformation from a value model to a reduced model takes three steps. First, we single the possible alternatives in the scenario. The second step selects the actors and value objects to be represented in the reduced model as units and common objects. Further, it builds the transformation tables for actors and value objects. The third step transforms actors and value objects to units and common objects.

6.1.1 Separate Alternatives

Due to the particular methodology used to develop the value perspective, the value model contains only one alternative scenario path. This includes all value exchanges. This means that in the first step of the transformation procedure no action is performed.

6.1.2 Select Value Objects and Actors

To select value objects means to identify the product value objects and entitle these for transformation; whereas the experience value objects are removed from the value model. In our value model, we have only one experience value object, namely 2.Attention. This is removed from further consideration in the transformation procedure.

To select actors means to identify the isolated actor, i.e. actors without value exchanges, and remove these from the value model. In our model, there are not isolated actors and, therefore, all actors are selected for transformation.

6.1.3 Build the Transformation Tables

The transformation tables capture the mapping between concept instances in the value and reduced models. For each unique instance of an actor and value object a unit and common object is chosen. We list the remaining value objects (without the removed experience value object) and actors, and name the corresponding common object and unit. Table VIII-30 shows the mapping of value objects and common objects; respectively, Table VIII-31 shows the mapping of actors and units.

Table VIII-30. Mapping of value object of the e^3 -value model to common value objects of the reduced model

<i>e³-value model</i>	<i>Reduced model</i>
1. Bundle of music content and commercials	Radio program
3. Exposure	Advertising
4. Fee	Advertising fee
5. Rights to make content public	Licence
6. Contracting music users	Contracting music users
7. Permission fees	Licence fee
8. Fee	Contracting fee
9. Permission fees	Make-public fees
10. Rights to make content public	Rights to make content public
11. Play lists	Play lists
12. Negotiable term and conditions	Negotiable contract
13. Fixed fee	Broadcast licence fee
14. Rights to make content public	Broadcast licence
15. Rights protection	Rights protection
16. Identification information	Identification information
17. Fee	Transaction fee
18. Money transfer	Money transfer
19. Rights to make content public	Rights
20. Permission fees	Permission fees

Table VIII-31. Mapping of actors of the e^3 -value model to business units of the reduced model

e^3 -value model	Reduced model
Listener	Listener
Advertiser	Advertiser
Bar, Disco or Shop	Music User
Branch Organization	Branch Organization
Radio Station	Radio
Internet Radio Station	Internet Radio
Rights Society	Rights Society
Bank	Bank
Content Producer (Rights Owner)	Content Producer

6.1.4 Generate Reduced Models

This step transforms the only alternative in the e^3 -value model into a reduced model. In particular, actors and value objects are transformed into business units and common value objects as specified in the mapping tables (see Table VIII-30 and Table VIII-31). As a result, the specific information on reciprocity of value exchanges and on bundling of value objects is omitted. The reduced model, derived from the e^3 -value model (Figure VIII-31), is depicted in Figure VIII-32.

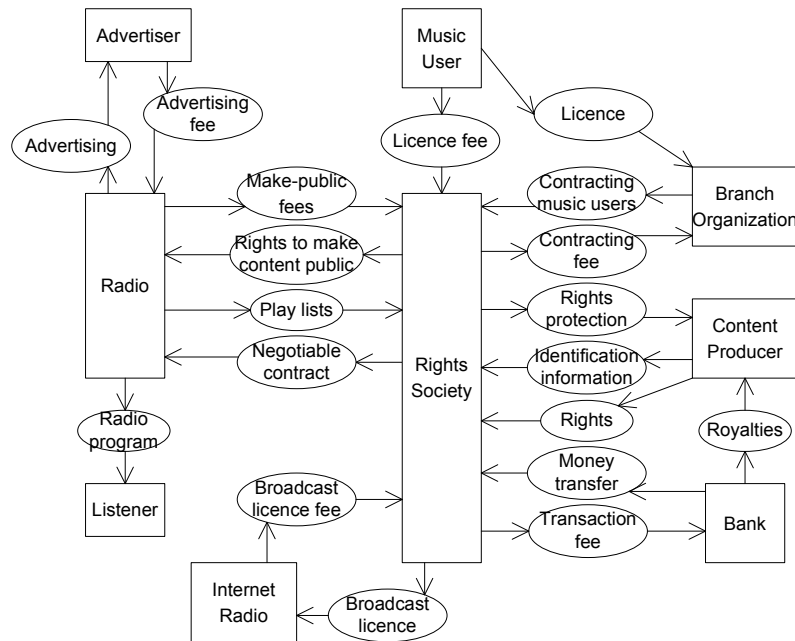


Figure VIII-32. Reduced model corresponding to the value model

6.2 Transform the Process Model to Reduced Models

The process specification consists of five activity diagrams. These resulted from the synthesis procedure described in Section 5. Annex - Synthesized Sub-Processes on page 261 features the activity diagrams making the process specification.

The transformation from activity diagrams to reduced models takes three steps. The first step breaks choices and cycles in the control flow. The second step identifies sequences of messages that form a transaction and marks single messages within the transaction as corresponding to value exchanges in the e^3 -value-model sense. Moreover, it builds the transformation tables for swimlanes and selected messages. The third step transforms swimlanes to units and messages to common objects.

Due to the particular methodology used to develop the process perspective, the specification contains several activity diagrams. Because the transformation procedure from Chapter VII Consistency between Value and Process Models Section 6.3 Transformation from an Activity Diagram to Reduced Models on page 153 works with only one activity diagram, we assume that all activity diagrams have single starting event and they are all executed in parallel. Despite the assumption, we apply the transformation procedure per activity diagram. This results in a number of reduced models originating from one activity diagram. To compensate the transformation of a single activity diagram, we take afterwards all combinations of reduced models.

The activity diagrams comprising the process specification contain elements (actors and messages) with the same names; nevertheless, these model different real-life instances. To uniquely identify swimlanes and messages, we use the abbreviated names of the activity diagrams as name extensions. For example, we refer to the Advertiser actor within the activity diagram Advertising (A) with A.Advertiser. The abbreviated names are given in Section 5.3.2 and repeated in the Annex - Synthesized Sub-Processes on page 261.

The process specification contains the following five activity diagrams:

- Advertising (A)
- Contracting branch organization (CBO)
- Contracting Internet radio station (CIRS)
- Contracting radio station (CRS)
- Outsourcing payment (OP)

Each of them will produce a number of alternative activity diagrams, which respectively will result in a number of reduced models. To distinguish between alternatives, we give names that include the abbreviation of the activity diagram, concatenated with natural number. For example, the second alternative from the Outsourcing payment (OP) activity diagram is named OP2.

6.2.1 Remove Choices and Cycles

The first step in the transformation procedure is to remove the choices in the activity diagram. By this, we identify alternative execution paths and further consider them separately such that each leads to one reduced model. A desired side effect of the removal of choices is the breaking of cycles in the activity flow. The activity diagram is traversed from the beginning to the end and, each time a branch leads to an already visited part of the diagram, the branch is not followed. Considering iterations through cycles would not

change the reduced models because transactions produced from a cycle would be mapped to the same common value object.

The result of the removal of choices is shown in the next sub-section after we discuss the identification of transactions and messages.

6.2.2 Identify Eligible for Transformation Swimlanes and Messages

The second step identifies transactions and marks single messages within the transactions as corresponding to value object. Transactions of messages are easier to identify before the activity diagrams are split into alternative paths. Therefore, we identify the transactions before removing choices and, later, judge if a transaction is present in the alternative paths. The identification of a message within a transaction that will represent the value object being exchanged is also identified in the activity diagram before the splitting of choices.

Below, we represent the identified transactions, the messages representing the value object within transactions, and the alternative activity diagrams. The transactions are denoted in the figures as grey clouds surrounding the exchanges messages. The selected message within a transaction is shown with grey background. Each alternative is shown in a separate activity diagram.

Below each of the five activity diagrams (and the respective alternatives), a summary is presented of the identified messages and swimlanes to be mapped to common objects and units. A message is eligible for transformation if a particular alternative contains all messages in the transaction. A swimlane is eligible for transformation if it takes part in at least one exchange of a selected message.

A. Transactions, Messages and Alternatives for the Advertising (A) Activity Diagram

The activity diagram Advertising (A) is shown in Figure VIII-33 (a). The single choice contained in it leads to two alternative activity diagrams, presented in Figure VIII-33 (b) and Figure VIII-33 (c).

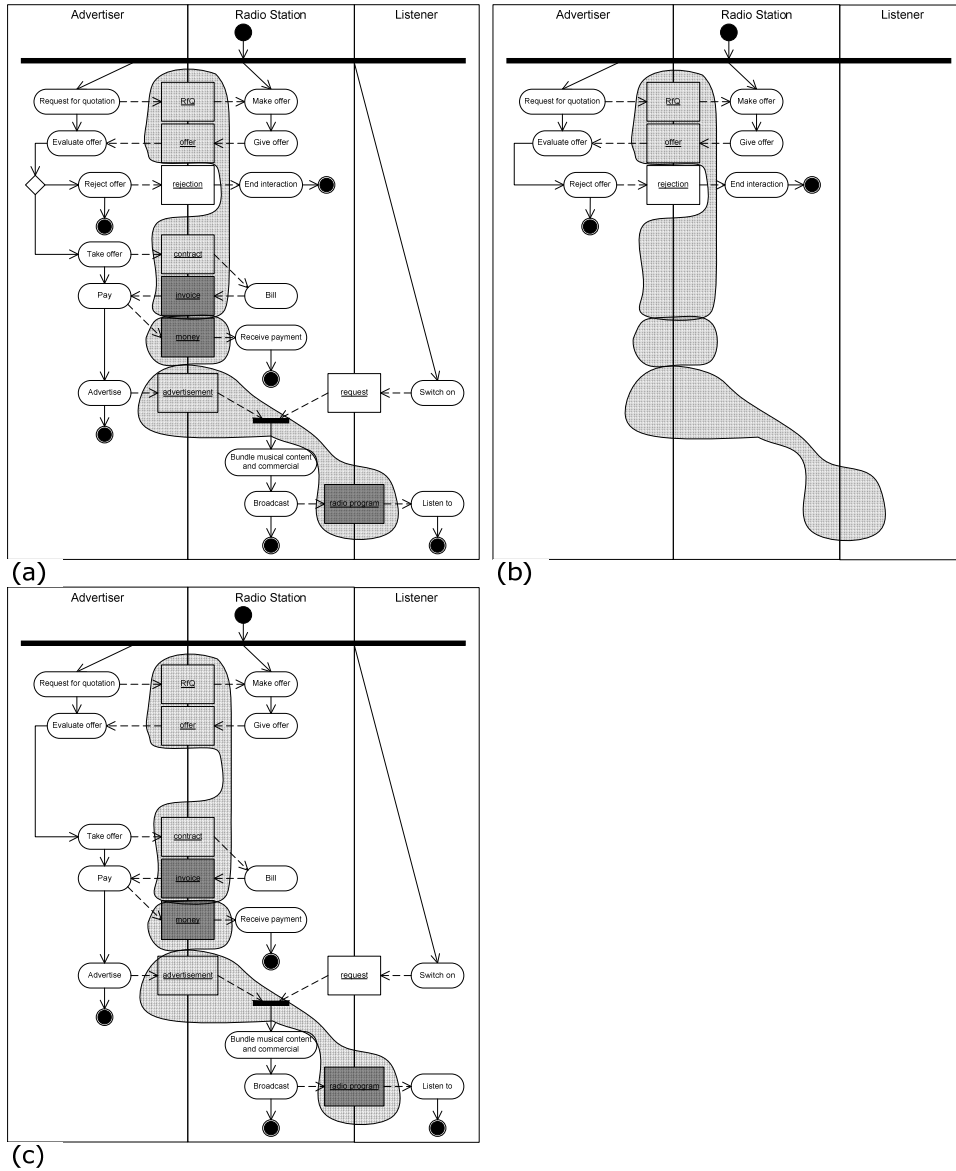


Figure VIII-33. Advertising activity diagram and alternatives, including transaction and messages: (a) – activity diagram A; (b) – first alternative activity diagram A1; and (c) – second alternative activity diagram A2

The Advertising activity diagram in Figure VIII-33 (a) contains three transactions (marked in grey) each of which nominates one message (marked in grey) to represent the exchanged value object. The selected messages are:

- A.invoice;

- A.money; and
- A.radio program.

The set of senders and receivers of these messages covers all present swimlanes. Thus, the eligible for transformation swimlanes are:

- A.Advertiser;
- A.Radio Station; and
- A.Listener.

Table VIII-32 presents per alternative activity diagram the eligible for transformation swimlanes and messages. The first alternative A1, Figure VIII-33 (b), contains no swimlanes and no messages. The second alternative A2, Figure VIII-33 (c), includes all swimlanes and messages.

Table VIII-32. Swimlanes and messages, eligible for transformation, in the Advertising alternative activity diagrams

<i>Alternative</i>	<i>Swimlanes</i>	<i>messages</i>
A1, Figure VIII-33 (b)		
A2, Figure VIII-33 (c)	A.Advertiser A.Radio Station A.Listener	A.invoice A.money A.radio program

B. Transactions, Messages and Alternatives for the Contracting branch organization (CBO) Activity Diagram

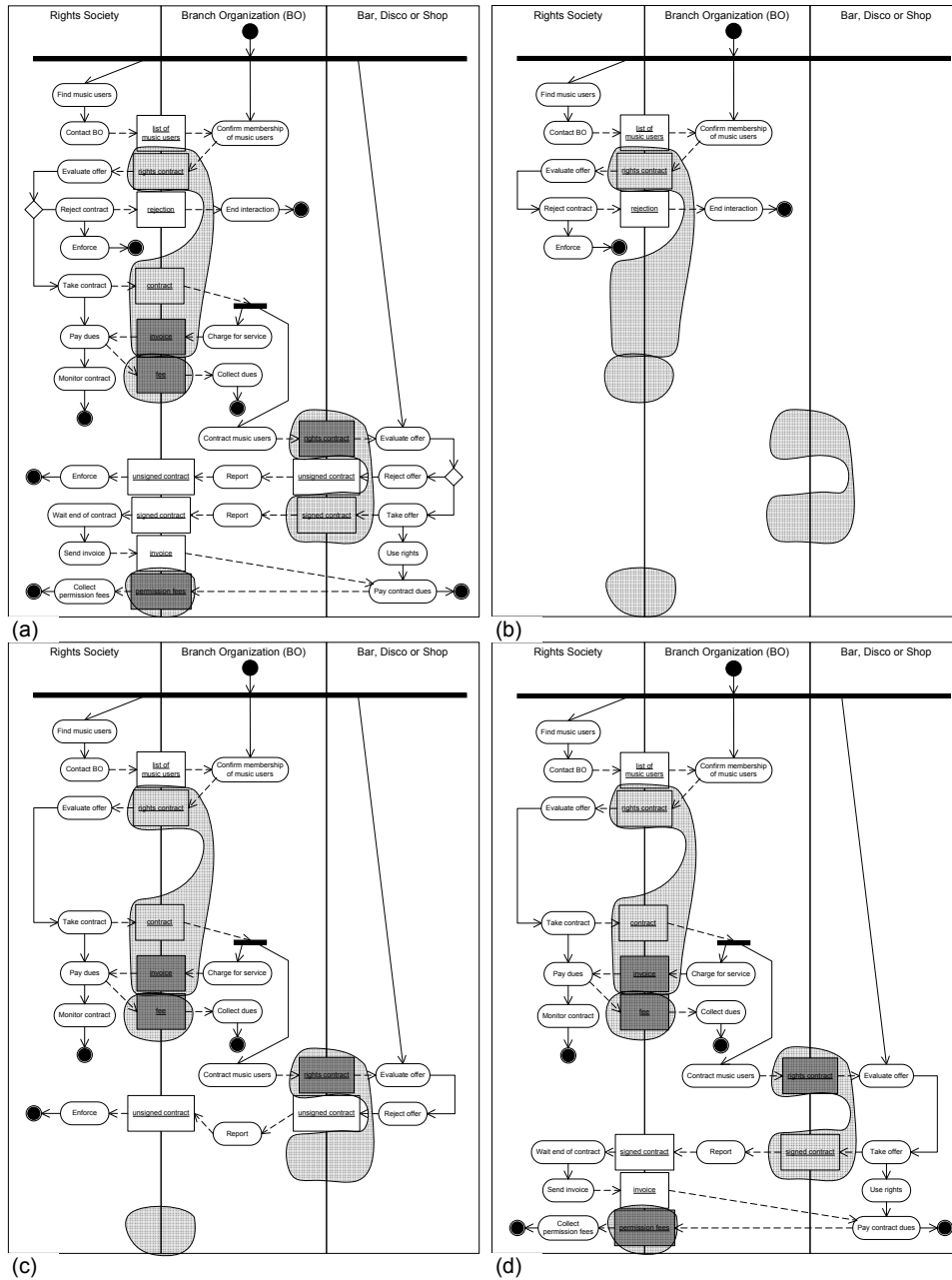


Figure VIII-34. Contracting branch organization activity diagram and alternatives, including transaction and messages: (a) – activity diagram CBO; (b) – first alternative activity diagram CBO1; (c) – second alternative activity diagram CBO2; and (d) – third alternative activity diagram CBO3

The Contracting branch organization activity diagram in Figure VIII-34 (a) contains four transactions (marked in grey) each of which nominates one message (marked in grey) to represent the exchanged value object. The selected messages are:

- CBO.invoice;
- CBO.fee;
- CBO.rights contract; and
- CBO.permission fees.

The set of senders and receivers of these messages covers all present swimlanes. Thus, the eligible for transformation swimlanes are:

- CBO.Rights Society;
- CBO.Branch Organisation (BO); and
- CBO.Bar, Disco or Shop.

Table VIII-33 presents per alternative activity diagram the eligible for transformation swimlanes and messages. The first alternative CBO1, Figure VIII-34 (b), contains no swimlanes and no messages. The second alternative CBO2, Figure VIII-34 (c), includes two of the swimlanes and two of the messages. The third alternative CBO3, Figure VIII-34 (d), includes all swimlanes and messages.

Table VIII-33. Swimlanes and messages, eligible for transformation, in the Contracting branch organization alternative activity diagrams

<i>Alternative</i>	<i>Swimlanes</i>	<i>Messages</i>
CBO1, Figure VIII-34 (b)		
CBO2, Figure VIII-34 (c)	CBO.Rights Society CBO.Branch Organisation (BO)	CBO.invoice CBO.fee
CBO3, Figure VIII-34 (d)	CBO.Rights Society CBO.Branch Organisation (BO) CBO.Bar, Disco or Shop	CBO.invoice CBO.fee CBO.rights contract CBO.permission fees

C. Transactions, Messages and Alternatives for the Contracting Internet radio station (CIRS) Activity Diagram

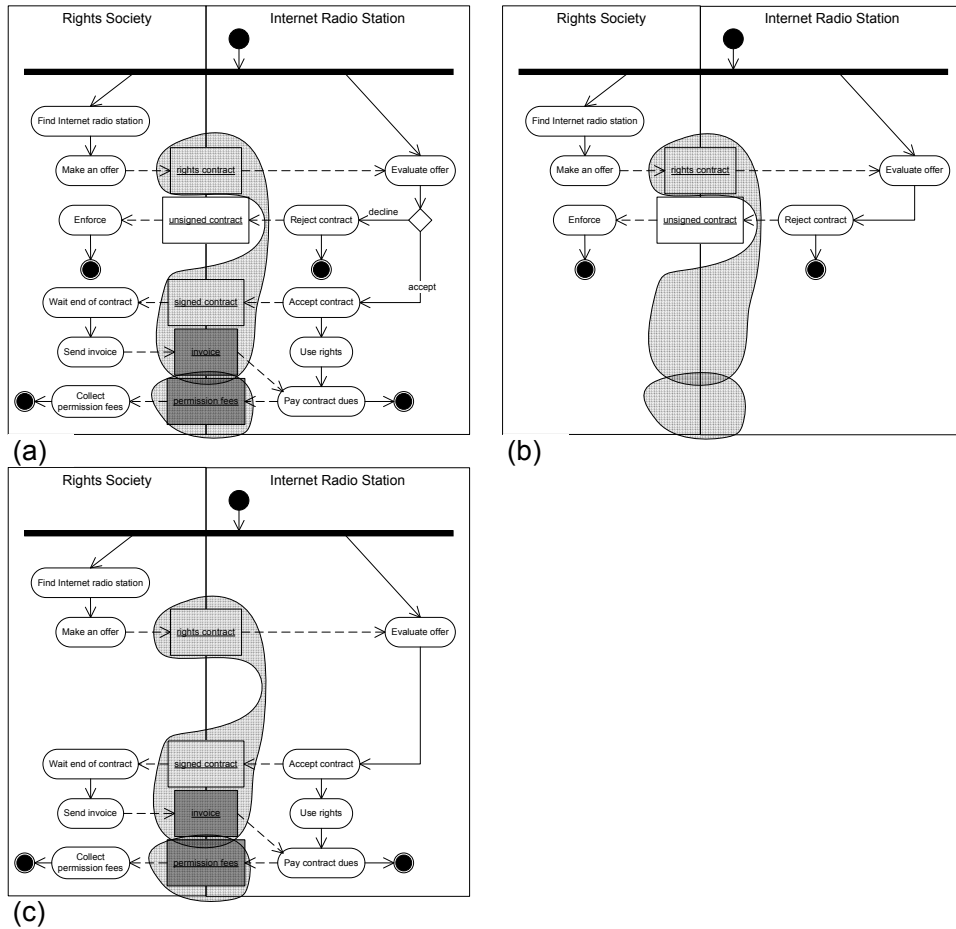


Figure VIII-35. Contracting Internet radio station activity diagram and alternatives, including transaction and messages: (a) – activity diagram CIRS; (b) – first alternative activity diagram CIRS1; and (c) – second alternative activity diagram CIRS2

The Contracting Internet radio station activity diagram in Figure VIII-35 (a) contains two transactions (marked in grey) each of which nominates one message (marked in grey) to represent the exchanged value object. The selected messages are:

- CIRS.invoice; and
- CIRS.permission fees.

The set of senders and receivers of these messages covers all present swimlanes. Thus, the eligible for transformation swimlanes are:

- CIRS.Rights Society; and
- CIRS.Internet Radio Station.

Table VIII-34 presents per alternative activity diagram the eligible for transformation swimlanes and messages. The first alternative CIRS1, Figure VIII-35 (b), contains no swimlanes and no messages. The second alternative CIRS 2, Figure VIII-35 (c), includes all swimlanes and messages.

Table VIII-34. Swimlanes and messages, eligible for transformation, in the Contracting Internet radio station alternative activity diagrams

<i>Alternative</i>	<i>Swimlanes</i>	<i>Messages</i>
CIRS1, Figure VIII-35 (b)		
CIRS2, Figure VIII-35 (c)	CIRS.Rights Society CIRS.Internet Radio Station	CIRS.invoice CIRS.permission fees

D. Transactions, Messages and Alternatives for the Contracting radio station (CRS) Activity Diagram

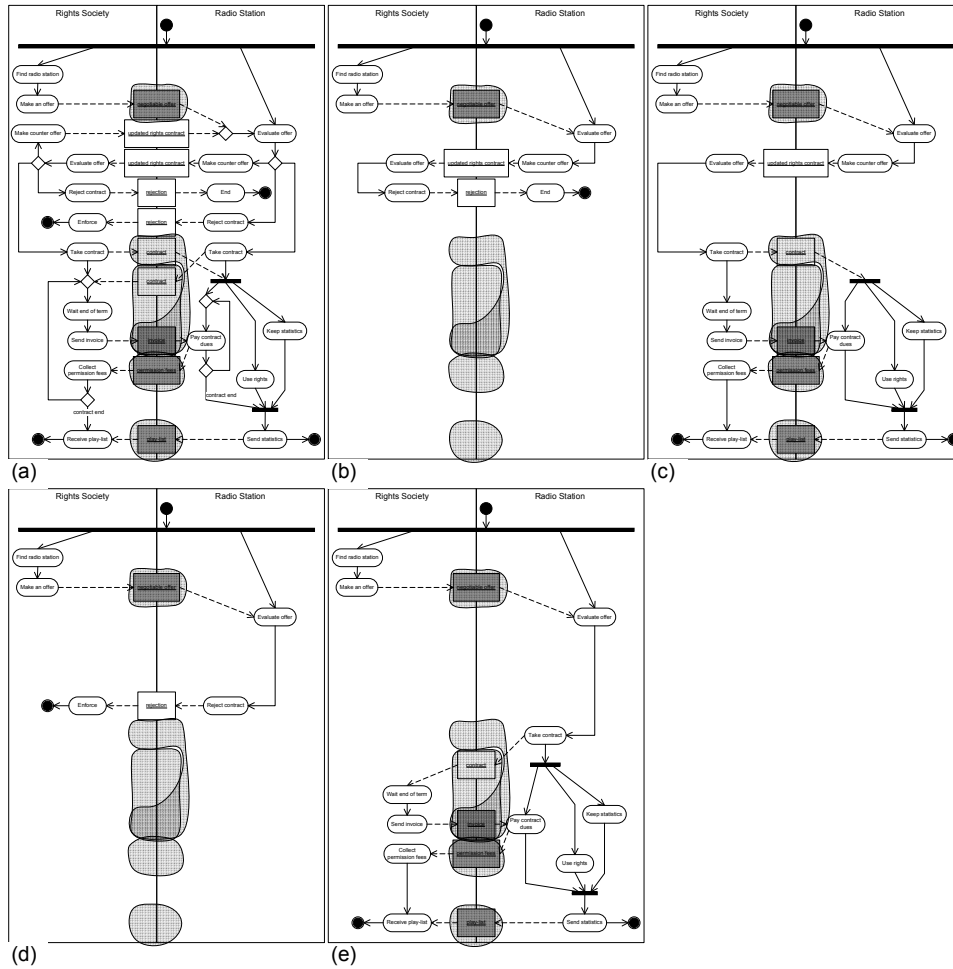


Figure VIII-36. Contracting radio station activity diagram and alternatives, including transaction and messages: (a) – activity diagram CRS; (b) – first alternative activity diagram CRS1; (c) – second alternative activity diagram CRS2; (d) – third alternative activity diagram CRS3; and (e) – fourth alternative activity diagram CRS4

The Contracting radio station activity diagram in Figure VIII-36 (a) contains four transactions (marked in grey) each of which nominates one message (marked in grey) to represent the exchanged value object. The selected messages are:

- CRS.negotiable offer;
- CRS.invoice;
- CRS.permission fees; and
- CRS.play-list.

The set of senders and receivers of these messages covers all present swimlanes. Thus, the eligible for transformation swimlanes are:

- CRS.Rights Society; and
- CRS.Radio Station.

Table VIII-35 presents per alternative activity diagram the eligible for transformation swimlanes and messages. The first alternative CRS1, Figure VIII-36 (b) and Figure VIII-36 (e), contains two swimlanes and one message. The second alternative CRS2, Figure VIII-36 (c) and Figure VIII-36 (d), includes all swimlanes and messages.

Table VIII-35. Swimlanes and messages, eligible for transformation, in the Contracting radio station alternative activity diagrams

<i>Alternative</i>	<i>Swimlanes</i>	<i>Messages</i>
CRS1, Figure VIII-36 (b) and Figure VIII-36 (e)	CRS.Rights Society CRS.Radio Station	CRS.negotiable offer
CRS2, Figure VIII-36 (c) and Figure VIII-36 (d)	CRS.Rights Society CRS.Radio Station	CRS.negotiable offer CRS.invoice CRS.permission fees CRS.play-list

E. Transactions, Messages and Alternatives for the Outsourcing payment (OP) Activity Diagram

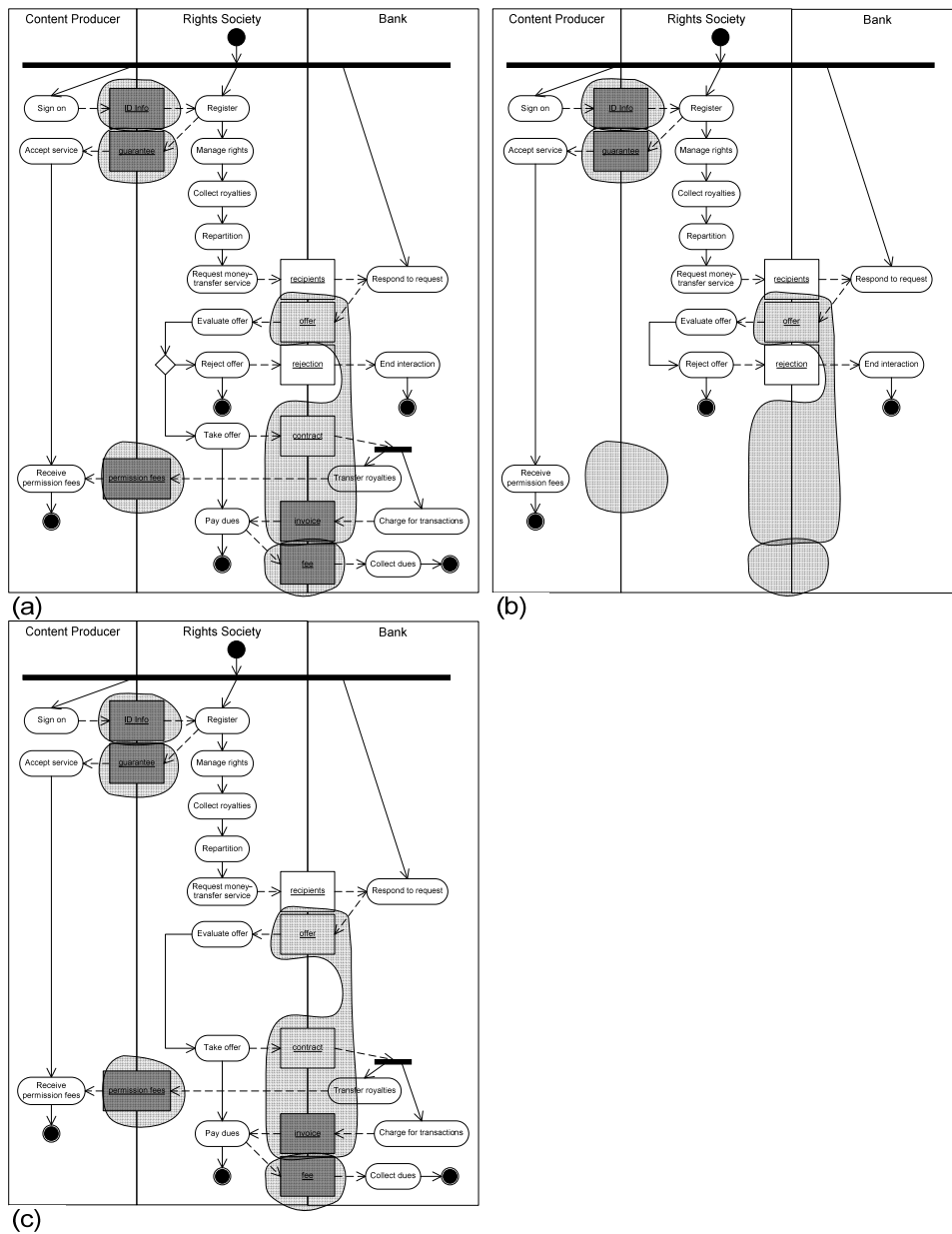


Figure VIII-37. Outsourcing payment activity diagram and alternatives, including transaction and messages: (a) – activity diagram OP; (b) – first alternative activity diagram OP1; and (c) – second alternative activity diagram OP2

The Outsourcing payment activity diagram in Figure VIII-37 (a) contains five transactions (marked in grey) each of which nominates one message (marked in grey) to represent the exchanged value object. The selected messages are:

- OP.ID info;
- OP.guarantee;
- OP.permission fees;
- OP.invoice; and
- OP.fee.

The set of senders and receivers of these messages covers all present swimlanes. Thus, the eligible for transformation swimlanes are:

- OP.Content Producer;
- OP.Rights Society; and
- OP.Bank.

Table VIII-36 presents per alternative activity diagram the eligible for transformation swimlanes and messages. The first alternative OP1, Figure VIII-37 (b), contains two swimlanes and two message. The second alternative OP2, Figure VIII-37 (c), includes all swimlanes and messages.

Table VIII-36. Swimlanes and messages, eligible for transformation, in the Outsourcing payment alternative activity diagrams

<i>Alternative</i>	<i>Swimlanes</i>	<i>Messages</i>
OP1, Figure VIII-37 (b)	OP.Content Producer OP.Rights Society	OP.ID info OP.guarantee
OP2, Figure VIII-37 (c)	OP.Content Producer OP.Rights Society OP.Bank	OP.ID info OP.guarantee OP.permission fees OP.invoice OP.fee

6.2.3 Build the Transformation Tables

The transformation tables capture the mapping between concept instances in the activity diagrams and reduced models. For each unique instance of a swimlane and message a unit and common object is chosen. We collect the swimlanes and selected messages from all activity diagrams and name the corresponding units and common objects. Table VIII-37 shows the mapping of messages and common objects; respectively, Table VIII-38 shows the mapping of swimlanes and units.

Table VIII-37. Mapping of the selected messages from the process specification (all activity diagrams) to common value objects in the reduced model

<i>Activity diagram</i>	<i>Reduced model</i>
A.radio program	Radio program
A.invoice	Advertising
A.money	Advertising fee
CBO.rights contract	Licence

CBO.invoice	Contracting music users
CBO.permission fees	Licence fee
CBO.fee	Contracting fee
CRS.permission fees	Make-public fees
CRS.invoice	Rights to make content public
CRS.play-list	Play lists
CRS.negotiable offer	Negotiable contract
CIRS.permission fees	Broadcast licence fee
CIRS.invoice	Broadcast licence
OP.guarantee	Rights protection
OP.ID info	Identification information
OP.fee	Transaction fee
OP.invoice	Money transfer
OP.permission fees	Permission fees

Table VIII-38. Mapping of swimlanes from the process specification (all activity diagrams) to units in the reduced model

<i>Activity diagram</i>	<i>Reduced model</i>
A.Listener	Listener
A.Advertiser	Advertiser
CBO.Bar, Disco or Shop	Music User
CBO.Branch Organization (BO)	Branch Organization
A.Radio Station	Radio
CRS.Radio Station	
CIRS.Internet Radio Station	Internet Radio
CBO.Rights Society	Rights Society
CIRS.Rights Society	
CRS.Rights Society	
OP.Rights Society	
OP.Bank	Bank
OP.Content Producer	Content Producer

6.2.4 Generate reduced models

This step generates the reduced models corresponding to the process specification. In particular, swimlanes and messages are transformed into business units and common value objects according to the mapping tables Table VIII-37 and Table VIII-38. As a result, the specific information on sequence of message exchanges is omitted.

Before we derive the reduced models, we need to consider the fact that our process specification contains not one but five activity diagrams. Earlier (in the beginning of Section 6.2,) we stated that the compensation for the many activity diagrams is taking all combinations of the alternative activity diagrams resulted from the removing of choices. In particular, we have the following case:

- Advertising (A) results in alternatives A1 and A2;

- Contracting branch organization (CBO) results in alternatives CBO1, CBO2 and CBO3;
- Contracting Internet radio station (CIRS) results in alternatives CIRS1 and CIRS2;
- Contracting radio station (CRS) results in alternatives CRS1 and CRS2;
- Outsourcing payment (OP) results in alternatives OP1 and OP2.

Therefore, the reduced models are $2 \times 3 \times 2 \times 2 = 48$.

In Figure VIII-38, we show only one of the reduced models, namely the reduced model of combination A2, CBO3, CIRS2, CRS2, and OP2.

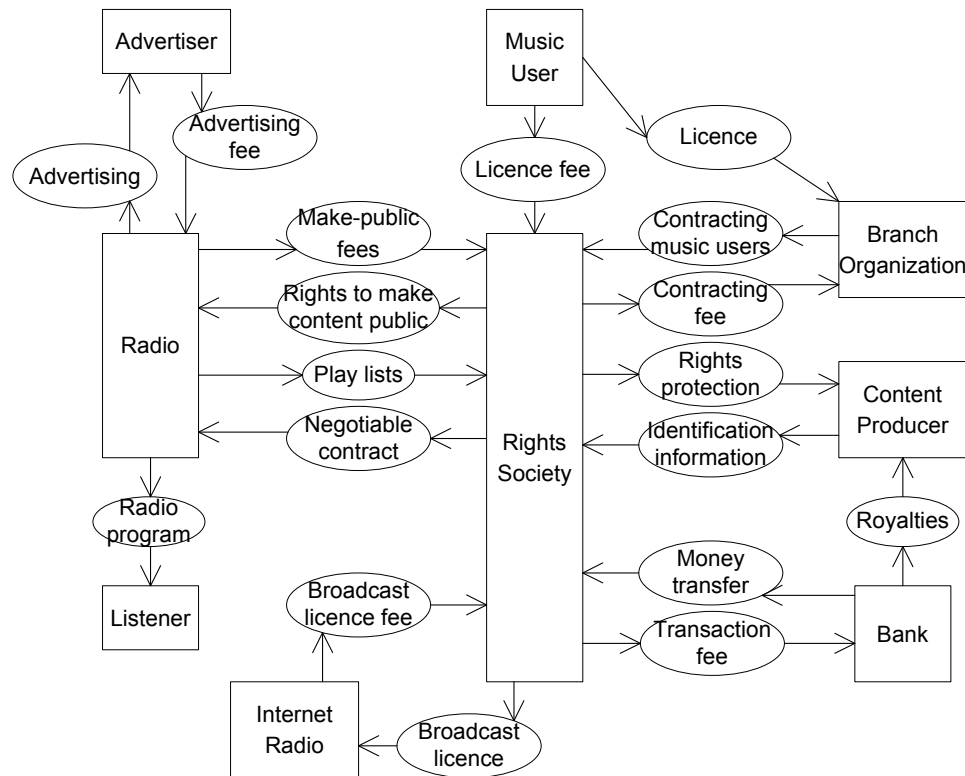


Figure VIII-38. Reduced model corresponding to the combination of A2, CBO3, CIRS2, CRS2, and OP2 activity diagrams

6.3 Evaluate Reduced Models

Clearly the value and process specification are not consistent. Consistency requires equivalent reduced models (see Chapter VII Consistency between Value and Process Models;) where the first condition of equivalence is to have equal number of reduced models produced by each specification.

The activity diagrams contain details about the option to refuse some exchanges. For example, the choice in the Advertising (A) activity diagram (Figure VIII-30 on page 230)

allows Advertiser to reject certain offers. This is not captured in the value model and produces alternative activity diagram and reduced model that does not have correspondence in the value model. If we abstract details about such alternative paths we can discard alternatives A1, CBO1, CBO2, CIRS1, CRS1, and OP1. The result is that we have only one reduced model, namely the one shown in Figure VIII-38.

Considering the reduced model from the value model (Figure VIII-32) and the reduced model for alternatives A2, CBO3, CIRS2, CRS2, and OP2 (Figure VIII-38), we conclude again that the value and process specification are not consistent. The reason is that the reduced model from the process model lacks one common value object, namely the Rights object flowing from the Content Producer unit to the Rights Society.

Examining the process model, we cannot find a message that represents rights flowing from the rights owners to the rights society. The intellectual property rights of the content producers are transformed into rights to make content public and given to a rights society to manage by a regulatory authority. The rights are assumed by the rights society in the moment it is founded.

In a scenario with several rights societies, we can assume that the rights are given in the moment of registration at the rights society. Following the assumption results in the messages OP.ID info mapping to two value objects: the Identification information and the 'missing' Rights. This mapping is one message to many value objects. In Chapter VII Consistency between Value and Process Models, Section 9 Granularity of Models: Many-to-Many Relationships between Instances, we describe our approach of splitting messages as a way to deal with one-to-many mappings. After splitting the OP.ID info message into two new messages which on their turn map the Identification information and the Rights value objects, the resulting reduced models are consistent.

7 Evaluating the Design Method

We validate the method in terms of the seven claims we made in Section 1.2. For each claim, we state our evaluation criteria. In the course of applying the method to the real-life example, we give arguments to believe that the claim holds true. The claims demonstrate the properties of the method.

Claim 1 (C1): Correct Models Can Be Developed. To confirm claim C1, we asked two experts in the field of value and process modelling to review the models. The experts were instructed to look for:

- *syntactic errors*, including modelling concepts and relations between concepts not allowed by the syntax of the particular notation;
- *semantic errors*, including syntactically correct statements that do not model the real world. Violation of well-formedness of models was also included; and
- *pragmatic errors*, including parts of the model with mixed granularity or style.

Both experts agreed that the value and process models represented possible state of reality.

Claim 2 (C2): Consistent Specifications Can Be Developed. The development of the real-life example resulted in an inconsistent specification according to the method itself

(Section 6.3.) The flexibility of the framework and method allowed us to identify the inconsistent statements. Furthermore, it allowed us to make assumptions about the source of inconsistencies until we eliminated these.

The inconsistencies were resolved by iterative interpretation of the intermediate models in the real-life example. This process showed that a consistent specification can be developed.

Claim 3 (C3): The Exploration of the Design Space Is Extensive and Automated. We confirm claim C3 by showing that every viable combination of design fragments is considered and compared with the others. Applying the design method for the process perspective in Section 5 resulted in 68 818 alternative designs (see Section 5.2.1) each of which was ranked, pre-selected and ranked again.

In the spectrum of development methods between the brute-force generation of all possible solutions and an assumption-based single path in the solution-space, our method begins with an automated generation and ranking of all possible solutions and, then, nominates the best candidates based on an informed choice.

Sections 4 and 5, where we apply the method to develop the value and process perspectives, make the validity of claim C3 evident.

Claim 4 (C4): Design Knowledge Is Reused. We confirm claim C4 by showing that patterns are reused. Particularly for the process model, we demonstrate that the final models contain patterns. The Annex - Synthesized Sub-Processes on page 261 shows the synthesized process models in which the reused patterns are marked. The same holds for the value model. Thus, we conclude that claim C4 holds true.

Claim 5 (C5): The Library Is the First Source of Design Fragments. We confirm claim C5 by showing that (1) the method searches first in the library of patterns for available designs and (2) all patterns in the final models come from the library.

The evidence that the method searches first into the library is in the description of the real-life example. Sections 4 and 5 sequence the development steps from which it is apparent that the first choice of design fragments are the libraries of patterns.

The examination of the synthesized value model in Figure VIII-18 on page 207 and the synthesized process models in Annex - Synthesized Sub-Processes on page 261 finds no design fragments that are patterns and do not come from the libraries. This shows that all patterns in our real-life example come from the two libraries.

We conclude that claim C5 holds true.

Claim 6 (C6): Fragments in the Final Specification Can Be Traced Back To Requirements. We confirm claim C6 by giving an example. We trace back to requirements one of the patterns in the business process model of the developed real-life example. The trace is as follows:

1. We take the first of the five sub-processes comprising the business process specification, namely the Advertising (A) sub-process in Annex - Synthesized Sub-Processes on page 261.
2. We take the top pattern in the activity diagram, namely the Take it or leave it pattern.

3. From the synthesis of the Advertising (A) sub-process in Section 5.3.2 on page 226, we find that the pattern is identified with the identifier TOL.1.
4. From the instantiation of the pattern shown in Annex - Instantiation of Process Patterns on page 250, we find the link to the pattern template.
5. From the library of process patterns in Appendix G Library of Process Patterns, Pattern 12. Take it or leave it, on page 375, we make the link to the capability model.
6. From the evaluation in Section 5.2.7 on page 223, we see that the capability model is part of the alternative solution capability model 2.
7. From Figure VIII-25 on page 223, we get back the match between the **C1** capability in the TOL.1 capability model and goal **Ge** from the **RAD** goal model.
8. From Table VIII-21 on page 216, we uncover that the requirement behind goal **RAD.Ge** is **Contract fixed offers**. Tracing further to a top-level goal, we reach in Figure VIII-19 on page 210 goal **RAD.G1: Maximize profitability**, passing through goals **RAD.Gc**, **RAD.G3**, and **RAD.G2**.

The example above confirms claim C6.

Claim 7 (C7): Design Decisions Are Explicit. The demonstration of claim C6 is an evidence also for claim C7. The possibility to trace a fragment in the model to a requirement means that the design decisions are explicit.

Properties of the Design Method. The seven claims above validate that our design method exhibits the four properties listed in the beginning of this section, namely:

- The method develops a consistent specification;
- The method is effective;
- The method is efficient; and
- The method provides explicit justification of design decision.

The four properties demonstrate the usefulness of the design framework, design method, and libraries of patterns.

8 Summary

This chapter validated our approach to reuse design patterns in the development process of e-business models. We applied the framework and method on a real-life example, which confirmed a number of claims that we had made about the design approach. The claims validated that the method exhibits a number of properties, which demonstrated the usefulness of the framework and libraries.

Annex - Instantiation of Process Patterns

This annex contains a summary of the instantiation of the selected process patterns. Every pattern is described in a bullet followed by the Correspondence table for roles, Correspondence table for activities, Correspondence table for messages, the original

activity diagram of the patterns, and the instantiated activity diagram. The Correspondence table for roles contains all roles and features in bold font the automatically mapped role and swimlane. The remaining correspondence tables contain only the activities and messages that change during the instantiation.

- Advertising (ADV) pattern

Correspondence table for roles and swimlanes

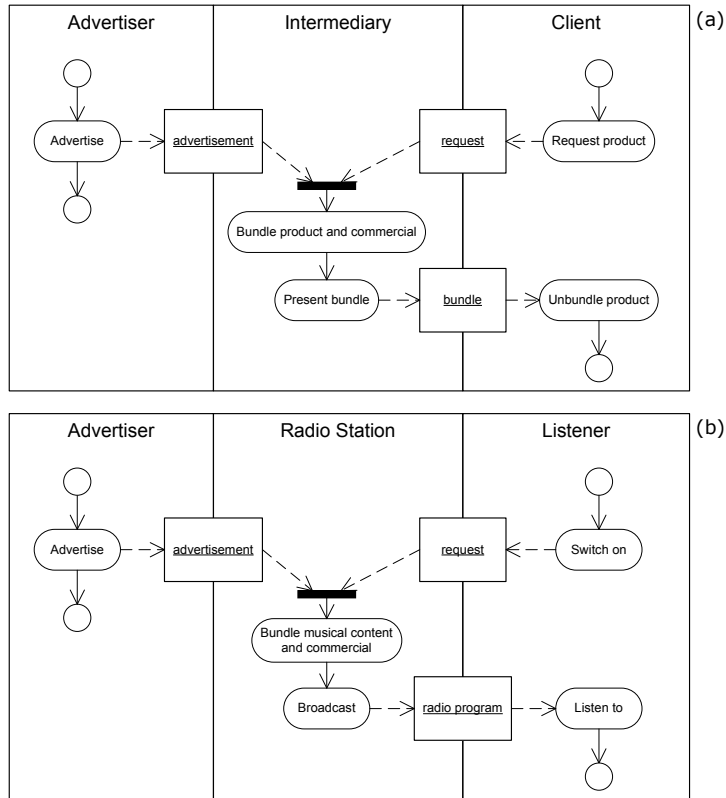
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
ADV.Advertiser	Advertiser
ADV.Intermediary	Radio station
ADV.Client	Listener

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
ADV.Bundle product and commercial	Bundle music content and commercial
ADV.Present bundle	Broadcast
ADV.Request product	Switch on
ADV.Unbundle product	Listen to

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
ADV.bunde	radio program



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 1. Advertising, on page 353 and (b) – instantiated activity diagram

- Offer counter-offer (OCO) pattern

Correspondence table for roles and swimlanes

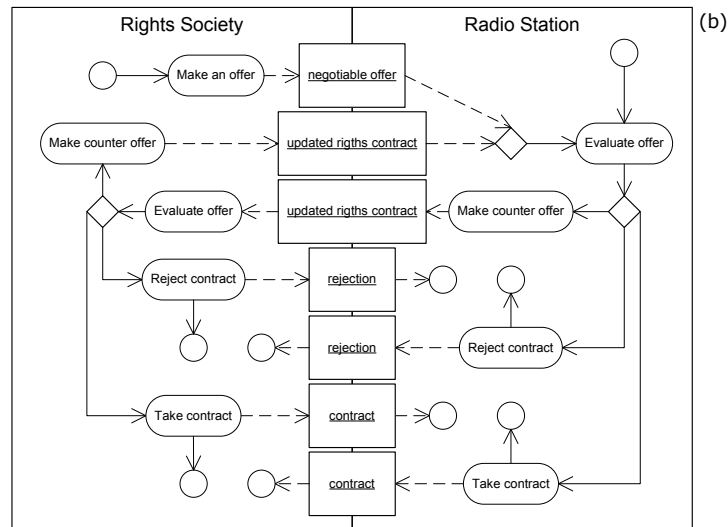
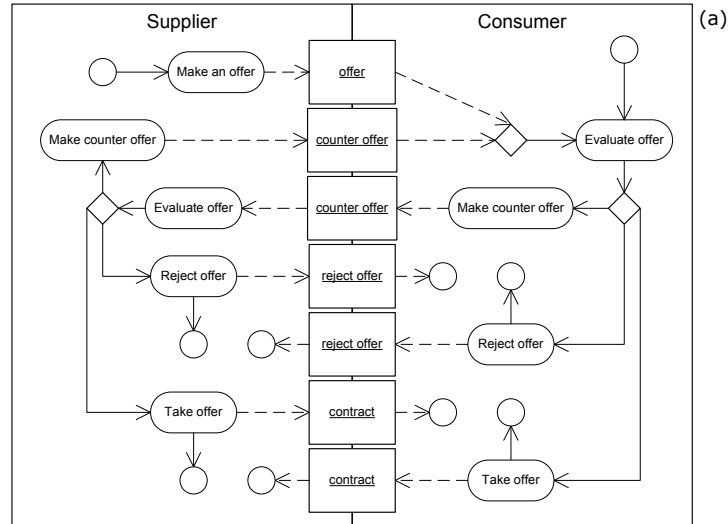
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
OCO.Supplier	Rights Society
OCO.Consumer	Radio station

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
OCO.Reject offer	Reject contract
OCO.Take offer	Take contract
OCO.Reject offer	Reject contract
OCO.Take offer	Take contract

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
OCO.offer	negotiable offer
OCO.counter offer	updated rights contract
OCO.counter offer	updated rights contract
OCO.reject offer	rejection
OCO.reject offer	rejection



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 5. Offer counter-offer, on page 361 and (b) – instantiated activity diagram

- First Outsourcing (OUT) pattern

Correspondence table for roles and swimlanes

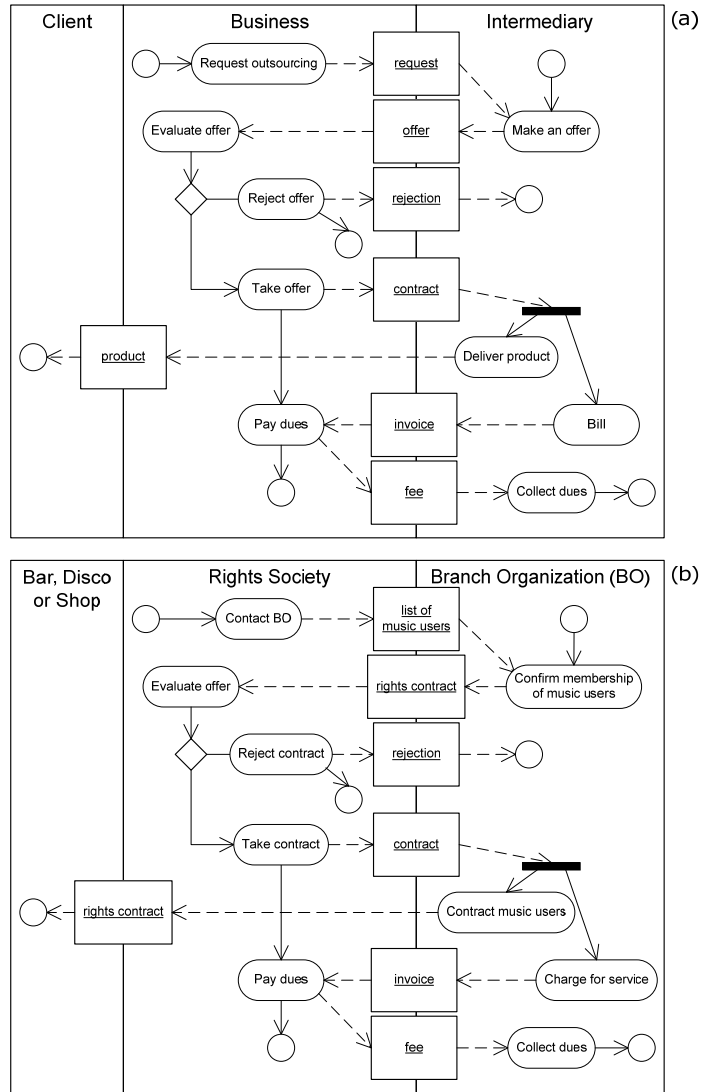
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
OUT.1.Intermediary	Branch Organization (BO)
OUT.1.Business	Rights Society
OUT.1.Client	Bar, Disco or Shop

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
OUT.1.Request outsourcing	Contact BO
OUT.1.Reject offer	Reject contract
OUT.1.Take offer	Take contract
OUT.1.Make an offer	Confirm membership of music users
OUT.1.Deliver product	Contract music users
OUT.1.Bill	Charge for service

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
OUT.1.product	rights contract
OUT.1.request	list of music users
OUT.1.offer	rights contract

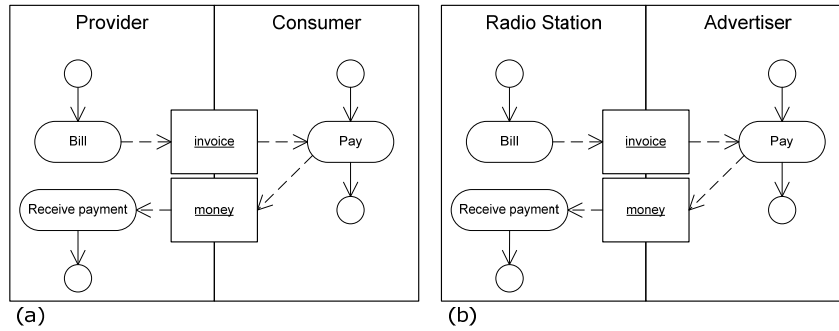


Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 6. Outsourcing, on page 363 and (b) – instantiated activity diagram

- First Payment (PAY) pattern

Correspondence table for roles and swimlanes

<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
PAY.1.Provider	Radio station
PAY.1.Consumer	Advertiser



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 8. Payment, on page 367 and (b) – instantiated activity diagram

- Second Payment (PAY) pattern

Correspondence table for roles and swimlanes

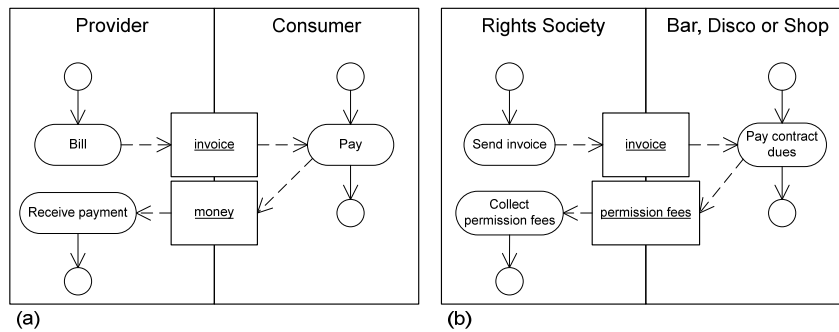
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
PAY.2.Provider	Rights Society
PAY.2.Consumer	Bar, Disco or Shop

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
PAY.2.Bill	Send invoice
PAY.2.Receive payment	Collect permission fees
PAY.2.Pay	Pay contract dues

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
PAY.2.money	permission fees



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 8. Payment, on page 367 and (b) – instantiated activity diagram

- Third Payment (PAY) pattern

Correspondence table for roles and swimlanes

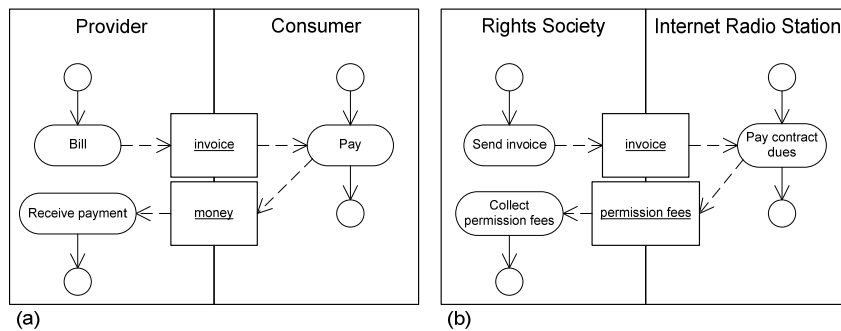
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
PAY.3.Provider	Rights Society
PAY.3.Consumer	Internet Radio Station

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
PAY.3.Bill	Send invoice
PAY.3.Receive payment	Collect permission fees
PAY.3.Pay	Pay contract dues

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
PAY.3.money	permission fees



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 8. Payment, on page 367 and (b) – instantiated activity diagram

- Payment in terms (PIT) pattern

Correspondence table for roles and swimlanes

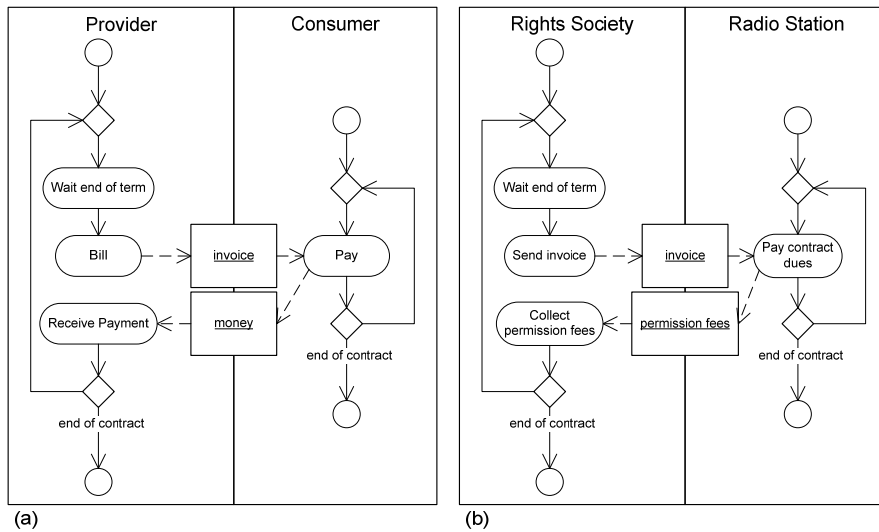
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
PIT.Provider	Rights Society
PIT.Consumer	Radio Station

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
PIT.Bill	Send invoice
PIT.Receive payment	Collect permission fees
PIT.Pay	Pay contract dues

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
PIT.money	permission fees



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 9. Payment in terms, on page 369 and (b) – instantiated activity diagram

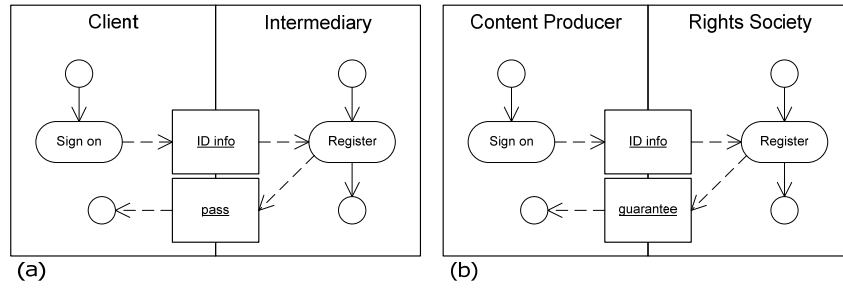
- Registration (REG) pattern

Correspondence table for roles and swimlanes

<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
REG.Intermediary	Rights Society
REG.Client	Content Producer

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
REG.pass	guarantee



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 10. Registration, on page 371 and (b) – instantiated activity diagram

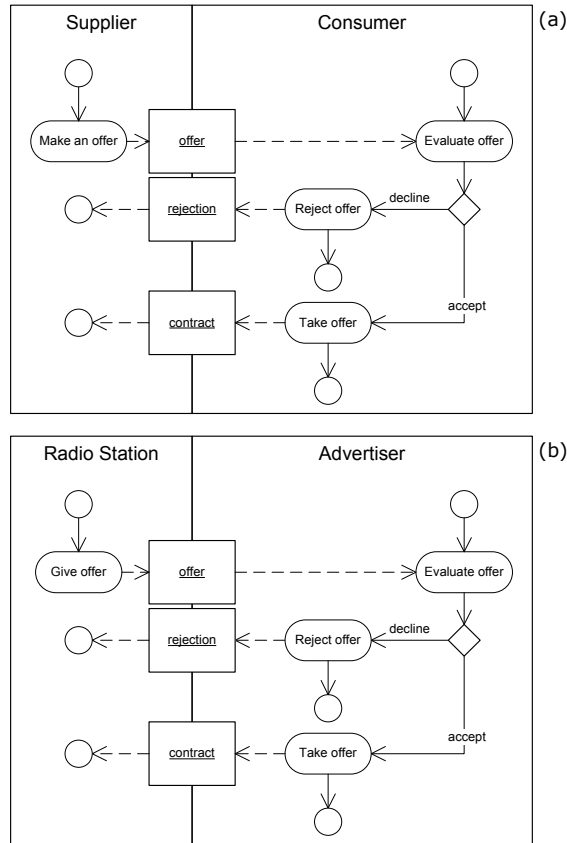
- First Take it or leave it (TOL) pattern

Correspondence table for roles and swimlanes

<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
TOL.1.Supplier	Radio Station
TOL.1.Consumer	Advertiser

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
TOL.1.Make an offer	Give offer



Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 12. Take it or leave it, on page 375 and (b) – instantiated activity diagram

- Second Take it or leave it (TOL) pattern

Correspondence table for roles and swimlanes

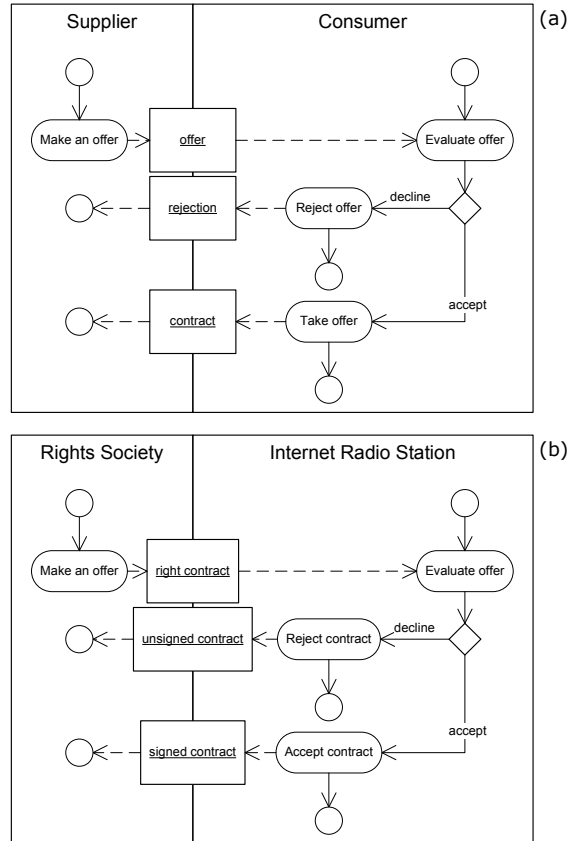
<i>Roles in the pattern</i>	<i>Swimlanes in the example</i>
TOL.2.Supplier	Rights Society
TOL.2.Consumer	Internet Radio station

Correspondence table for activities

<i>Activities as in the pattern</i>	<i>Activities as in the example</i>
TOL.2.Reject offer	Reject contract
TOL.2.Take offer	Accept contract

Correspondence table for messages

<i>Messages as in the pattern</i>	<i>Messages as in the example</i>
TOL.2.offer	Rights contract
TOL.2.rejection	unsigned contract
TOL.2.contract	signed contract

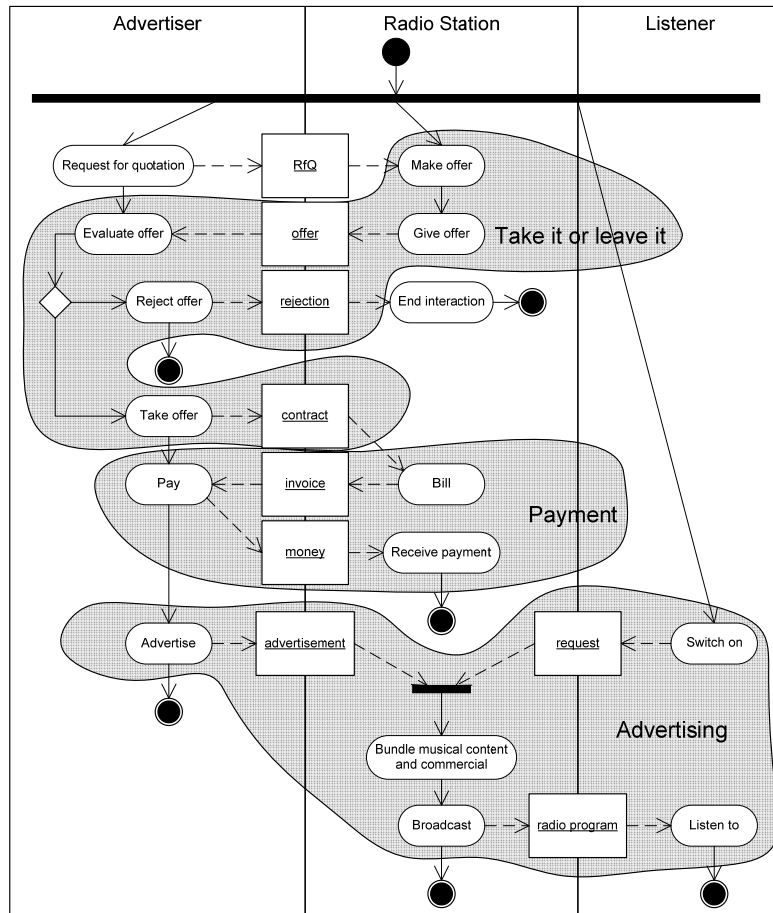


Activity diagrams: (a) – original activity diagram, copied from Appendix G Library of Process Patterns, Pattern 12. Take it or leave it, on page 375 and (b) – instantiated activity diagram

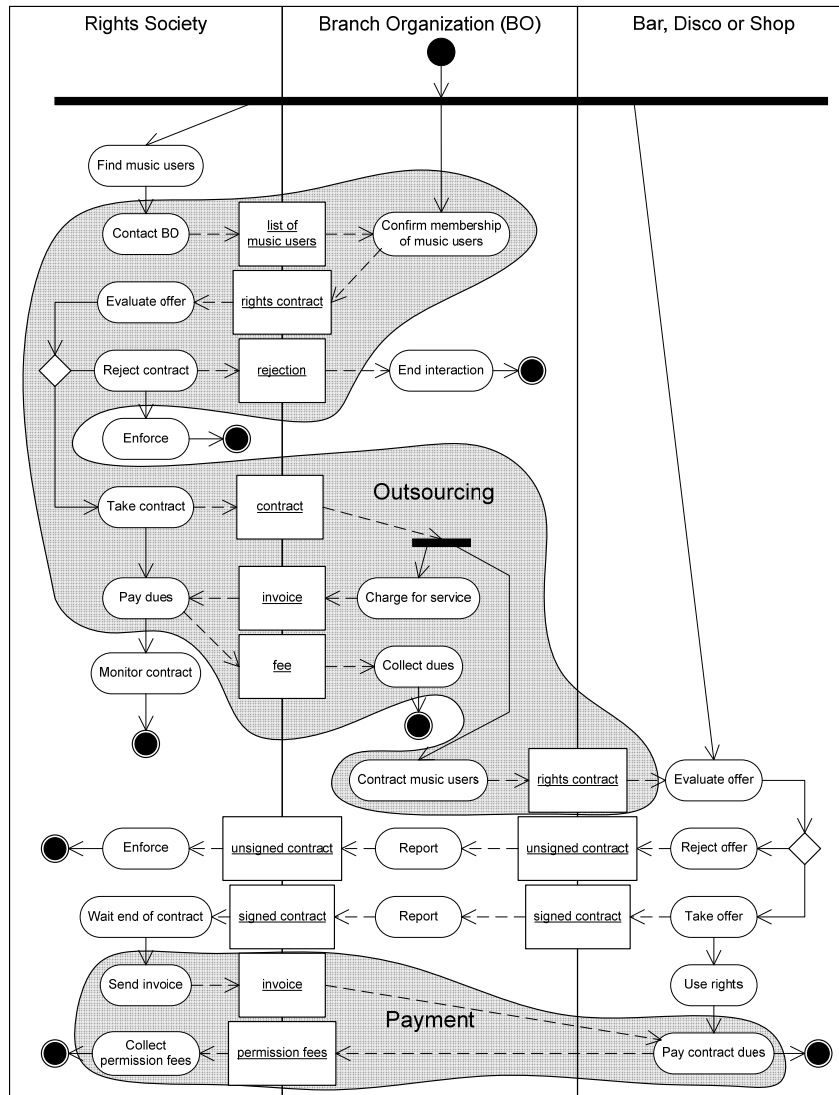
Annex - Synthesized Sub-Processes

This annex contains the five synthesised sub-processes. The synthesis of the first model, Advertising (A), is described in detail in Section 5.3.2. Here, we show the final results. Further, we present the final result of the synthesis of sub-processes Contracting branch organization (CBO), Contracting Internet radio station (CIRS), Contracting radio station (CRS), and Outsourcing payment (OP).

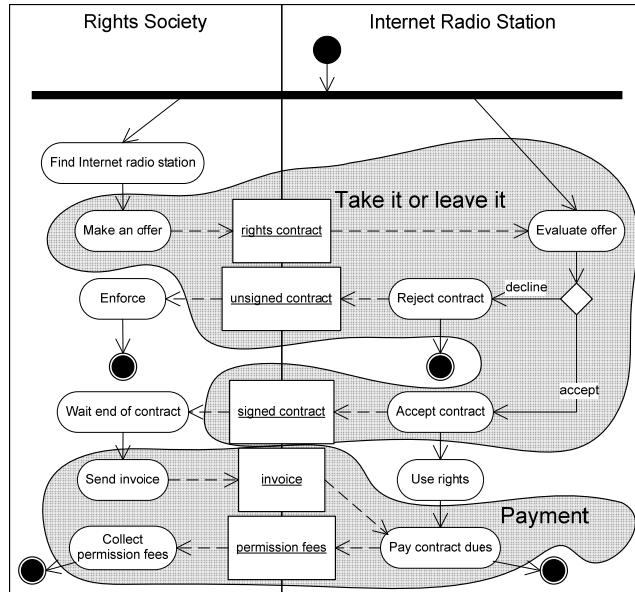
- Advertising (A) – synthesized from Advertising (ADV), Payment (PAY.1) and Take it or leave it (TOL.1)



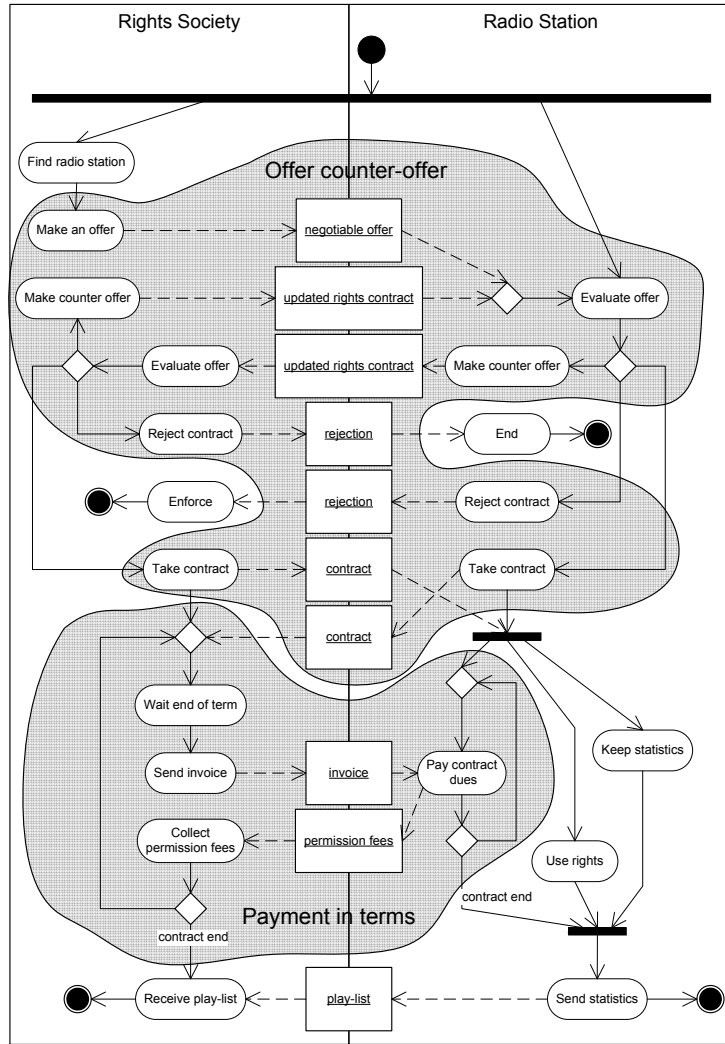
- Contracting branch organization (CBO) – synthesized from Outsourcing (OUT.1) and Payment (PAY.2)



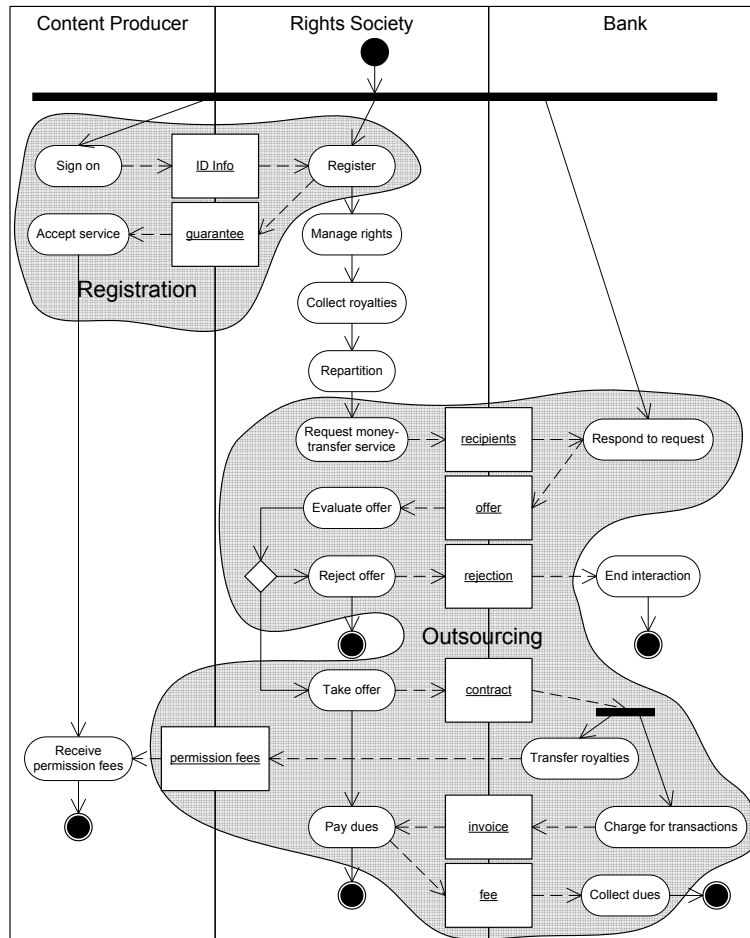
- Contracting Internet radio station (CIRS) – synthesized from Take it or leave it (TOL.2) and Payment (PAY.3)



- Contracting radio station (CRS) – synthesized from Offer counter-offer (OCO) and Payment in terms (PIT)



- Outsourcing payment (OP) – synthesized from Outsourcing (OUT.2) and Registration (REG)



Chapter IX

Conclusion

This chapter summarizes the main contributions and lists known limitations. It also gives directions for future research.

1 Contributions

As we stated in the introduction of the thesis, our main interest is the reuse of design knowledge in the development process of new e-business models. We have narrowed down the scope of our research goals to identification and reuse of design patterns in economic value and business process models of e-businesses. In addition to the constrain on type of design knowledge, we have restricted the domain of e-businesses to intermediaries that offer negotiation services. As a result, our first contribution is

- two libraries of design patterns.

The first library collects knowledge about exchanges of value objects between the participants of a business network. The second library contains knowledge about coordination messages exchanged between the participants in a business process. Both libraries derive patterns from models of negotiating intermediaries. Nevertheless, every found pattern was included in the library even though it may seem to be applicable also to another business domain. That means that the libraries are not limited to negotiating intermediaries; the restriction in scope was merely to make the identification doable in the given time. More, the documenting structure of patterns does not refer to the domain of negotiation. This allows adding new patterns. Moreover, the structure of the libraries enables extension. It is a list of independent items, which does not require relating new patterns with the rest.

The second contribution refers to the representation of the design knowledge. It is:

- formal description of pattern capabilities.

The annotation of each pattern with a capability model provides a measure of what the pattern can achieve. The measure is at a granularity level of an individual capability: a pattern can have several capabilities. However, we do not compare individual capabilities, although, this is generally possible. In our approach, we compare the capability of a combination of patterns to satisfy a number of business goals. The individual formalization of capabilities is used when matching capabilities and business requirements.

The two libraries can be used as-is; i.e., the design knowledge can be applied whenever appropriate in the development process of a new e-business model. To maximize the impact though, we provide a systematic approach to development which is our next contribution:

- a design framework for reuse of design patterns.

The framework supports business analysts in their creative job of developing new business models. It helps identify known solutions to the problems at hand. By using the

framework, parts of the final model are generated from catalogued patterns which exemplify previously applied and tested design decisions. The framework offers: (1) a single view on the system through a common set of requirements for all perspectives; (2) an automated evaluation of all possible design alternatives with the available patterns; (3) assistance in the synthesis process with two individual synthesis procedures, of which the one for value patterns is the first of its kind; and (4) a consistency definition and consistency check procedure for e^3 -value models and UML Activity diagrams which are also the first of their kind. Below, we emphasize on each of the four above-mentioned contributions.

Thus, the next contribution is:

- a formalized goal-modelling technique for business requirements, including propagation of goal-satisfaction values.

In our approach, requirements are represented as goals. The modelling technique formalizes goals and provides a measure of goal satisfaction. The satisfaction measure is stated in terms of the universe of discourse of the goal. The relationships among goals are also formalized. This gives an advantage in comparison with similar goal-modelling techniques because it offers qualitative measure of the interaction and satisfaction of goals. The ability to calculate the effects that goals cause one another allows propagating satisfaction values to a particular goal from the values of related goals.

Knowing the value of a goal allows us to assess if the goal is achieved: i.e. an approach to describe what is already present. Setting concrete values to goals turns the goals into capabilities. In this way, we model capabilities of patterns as goal models with fixed satisfaction values. This further allows us to integrate requirements and capabilities, and propagate satisfaction values throughout the joint model.

The propagation of satisfaction values is a key property of our goal-modelling technique. It is the basis for our next contribution:

- a selection procedure to explore the design space of all possible combinations of matching patterns for a particular design.

The selection procedure includes a matching step in which pattern capabilities are automatically matched with goals from the goal model. The combinations of such pre-selected patterns form the design space. Values from the pattern capabilities are propagated to top-level goals in the requirements model. This allows us to rate and pick the best combination of patterns for synthesis.

The actual reuse of design knowledge is the ability to link a fragment with the rest of the design into one model. Our next contribution is on this issue:

- synthesis procedures for value and process patterns.

The procedure assists the designer in the instantiation of the patterns. This includes assigning actual business participants to the roles defined in the patterns. After the instantiation step, the two procedures differ. The synthesis of value patterns automatically suggests a final model; whereas, the synthesis procedure of process pattern suggests a possible intermediate model per added pattern. Both procedures check for loose ends in the models and ensure technical quality of the models, meaning that the models are valid with respect to the e^3 -value and Activity diagram notation.

The validity of individual models is covered by the previous contribution. The next one ensures consistency of the final specification, namely:

- a consistency definition for value and process models, and a procedure to check the consistency criteria.

The proposed consistency definition between value and process models is the first of its kind. It results in a number of consistency criteria which are checked by the consistency check procedure. The procedure is specific to the *e³-value* and Activity diagram notations.

Overall, the assembly of the design framework offers:

- an improved alignment between goals, value and process models.

The framework links (1) business requirements represented as goals, (2) partial designs from value-exchange models, and (3) partial design from business process models. Its validation with a real-life example of a rights society shows that the three modelling notation can work together to produce meaningful specifications.

2 Positioning

From a Design theory [56] point of view, our approach is a prescriptive model that instructs the designers in the development process. The prescriptive model (i.e. the design framework and method) is not computable because it includes a number of manual steps: the manual steps use knowledge that is not part of the prescriptive design model. An evolutionary design process model prescribes how to design in iterative steps. In comparison, our framework and method are a design process with one evolutionary step. This is adding the process model to the value model or the other way around, and, consequently, checking the specification for consistency.

Further, we position the components of our framework and method with respect to other research. Figure IX-1 shows an overview of our framework and method. The numbered components are discussed below.

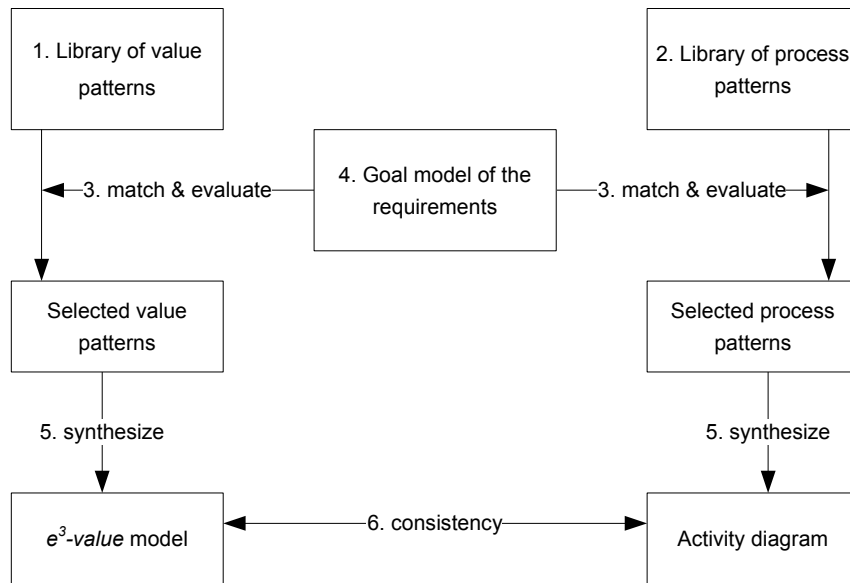


Figure IX-1. Overview of the framework (adapted from Figure III-6)

The value patterns (number 1 in Figure IX-1) share some properties with Weill and Vitale's [59] atomic business models. Both model exchanged value objects between businesses; however, the advantage of our representation of the design knowledge is that this is searchable on the basis of pattern capabilities.

The process patterns (number 2 in Figure IX-1) capture the same design knowledge as in some of the RosettaNet [51] Partner Interface Processes (PIPs). The difference is that the PIPs are designed for composition: i.e. each PIP has a predefined set of other PIPs that can link to. In addition to the flexibility, our patterns are annotated with machine interpretable capability models. The IBM Patterns for e-business [4, 31] are patterns that cover only the communication perspective, whereas our process patterns include both communication and coordination perspectives.

The selection procedure (number 3 in Figure IX-1) is based on the idea of evaluating patterns with respect to their contribution to the satisfaction of goals in a goal model. Gross, D. and Yu, E. [28] and Weiss, M. [61] use the same approach. However, our approach has an advantage of propagating qualitative satisfaction values throughout the model.

Our goal model (number 4 in Figure IX-1) is a goal theory about causal relationships among goal variables. We use it to represent requirements for or capabilities of a system. To our knowledge, the use of a goal model to represent capabilities is first of its kind; while, there are various approaches that represent requirements. For example, i^* [72] is a goal-modelling technique that models actors, tasks, resources, and goals. Despite that we do not have a notion of tasks and resources, the most important difference is in the relations between goals. i^* gives means-to-achieve-an-end semantics to relationships; whereas, our approach models causal relationships among the variables operationalizing the goals.

With respect to the propagation of satisfaction values throughout the model, various approaches use reasoning with goals. For example, Giorgini, P. et. al. [24], as part of the TROPOS [11] project, propose a technique to reason with partially satisfied goals. In their quantitative approach, each goal is assigned a value from 0 to 1 to represent the evidence of satisfiability of a goal. We differ in the domain of value assigned to goals and propagation functions. In TROPOS, evidence of satisfiability can be interpreted as, e.g., a probability of a goal to be satisfied or percentage of satisfaction. In our approach, goal variables have a domain and value corresponding to the goal that they operationalize. This enables designer and specialists to express and read their concerns in the particular domain. Moreover, the evaluation functions in our approach provide a measure of satisfaction of goals. Such evaluation functions and domain specific values are use in the KAOS [36] design method. In particular, Letier, E. and Lamsweerde van, A. [39] propose a propagation functions assigned to relationships between goals and a formalization of goals with variables and objective functions from the universe of discourse of the goal. We differ in the restriction on relationships between goals. We allow any causal relationship between the variables the relationships, while they permit only refinement relationships between goals.

Our synthesis procedure (number 5 in Figure IX-1) automates the process of constructing one design from a number of patterns. The procedure aims at the right balance between automatic and manual activities. With respect to that, it is in the middle between approaches such as OOram [49] and POAD [70]. OOram [49] specifies a role based method for automatic synthesis of patterns into safe models (models with guaranteed dynamic properties of the individual patterns in the synthesized model.) Our synthesis is automated and the resulting designs are unsafe in the sense of OOram. Our synthesis procedure is meant to support the designers rather than to automatically synthesize model fragments. Contrary, the POAD design method is prescription of manual steps to be performed by the designers. It is more a guideline for manual execution of the process, in which the construction of designs uses external information about pattern dependencies.

The approach to consistency (number 6 in Figure IX-1) with a common semantic model reuses research done before; however, the definition of a reduced model for e^3 -value models and activity diagrams is the first of its kind. It operationalizes Wieringa and Gordijn's [62] idea of correctness relationship between an e^3 -value model and a process model. The work of Dijkman et al. [15] is also based on the common semantic model approach. It maps concepts and relations in models to basic concepts. Our approach differs in how the reduced model is defined. We do not require a pre-defined reduced model with abstract basic constructs, but we determine the reduced model after the modelling notations are selected. This allows defining richer reduced models in terms of common concepts and relations.

3 Limitations

Our work has several known limitations. We distinguish two types of limitations: *contingent* limitations are those that could be removed with more work and *necessary*

limitations are those that cannot be removed without essentially changing the approach. Using this division, we provide below the known limitations.

3.1 Contingent limitations

Our first limitation concerns the capability models of patterns. The values assigned to capability variables are assumed to demonstrate the applicability of the approach. The

- capability values of patterns are realistic but only an example.

A mitigation action would be an empirical derivation of concrete value for the capability variables.

The next limitation comes from the scope of the research. Due to the surveyed business models,

- the libraries contain only patterns from the domain of negotiating intermediaries.

This limitation applies to the current version of the libraries. It can be removed by surveying the value and process models of e-businesses from other domains.

Another limitation is that the

- requirements, represented as goals, match only top-level capabilities of patterns.

The current procedure of finding patterns that solve a particular problem accounts only for top-level capabilities of patterns. The limitation can be eliminated by extending the matching procedure to all pattern capabilities. The implication is that there will be more patterns to consider in the development process and, consequently, the design space will be bigger. The extension is useful if the information added by finer capabilities is more than the information lost from approximation before the manual steps. This trade-off needs investigation.

In general, different people develop the goal model of the requirements and the capability models of patterns. Most probably, they use different knowledge sources and perspectives. This results in models, the goals of which are difficult to match automatically. We simplify the matching activity by assuming that the

- requirements, represented as goals, and capabilities are represented in a common conceptual model.

The limitation can be mitigated by delegating more responsibilities to a person or by mapping the conceptual models of the involved domains.

The process model is often seen as an implementation of the value model. We do not hold this view and, generally, this does not interfere with our consistency definition. However, the consistency definition requires

- the value and process models to be with comparable granularity.

The limitation can be overcome with more elaborate reduced model and transformation procedures, which would allow many-to-many relationships between the two models.

The last limitation that could have been removed with more work concern the validation of the whole approach. The time allowed us to perform

- one real-life example.

Although this is enough to demonstrate the usefulness of the approach, more experience with the framework could provide information about the usefulness of the assumptions we make.

3.2 Necessary limitations

The way we approach the matching of goals and capabilities is by introducing a number of simplifying assumptions. The result is the limitation that the

- requirements, represented as goals, and capabilities match only one-to-one.

An alternative approach that would allow matching, e.g., a group of goals with a group of capabilities would affect the selection procedure for patterns.

The selection procedure for relevant patterns contains manual steps. The integration of capability models into the goal model of the requirements is one of these. The manual step has to be performed on every alternative goal model. The alternative goal models are much more than ten in a real-life example. Assuming that ten is too many for manual integration, we keep the design space manageable with the following limitation:

- alternative solution capability models are pre-selected with incomplete information.

The implication is that discarded alternative solution capability models could have scored better after the integration step. This limitation can be reduced by modifying the selection procedure such that all alternative goal models undergo integration. The manual work can be aided with reuse of knowledge from already integrated alternatives.

The current synthesis procedure for value patterns generates a final model from fragments in one step. However,

- the dependency path between exchanges in the value model cannot be determined by the design method.

To mitigate this limitation, value patterns need to account for the dependencies between value exchanges. Furthermore, the synthesis procedure has to adapt to the richer design knowledge in one pattern.

The current synthesis procedure of process patterns needs an input from a person to determine the execution order of the activities. The limitation is that

- the sequence of activities and message exchanges cannot be determined without additional information.

The execution order cannot be determined with information coming from the pattern. The sequence of activities and message exchanges can have an impact on the business. Thus to mitigate this limitation, additional information in a form of requirement needs to be made available to the designer or synthesis procedure.

4 Future Work

Above, we identified actions that would mitigate the known limitations of our approach. Below, we list future work that would extend our design framework.

In the current version of the libraries, patterns are not described as templates. They are interpreted as templates and the synthesis procedure includes an instantiation step. However, the particular modelling notations (*e³-value* and UML Activity diagram) are used with modified semantics (e.g. the actor concept in the *e³-value* notation is interpreted as a role in the pattern solution.) To remove the ambiguity,

- a new or extended notation have to be defined such that the schematic description of patterns represent a template for design fragments.

In Chapter II Design Knowledge in Existing e-Business Models, Section 8 Relation between Value and Process Patterns, we intentionally unlinked the value and process patterns to simplify the overall structure of the design framework. However, the information in such relations can be used to determine relevant patterns. If a value pattern is selected this will automatically lead to selection of a process pattern. Therefore, we need to

- find relations between value and process patterns.

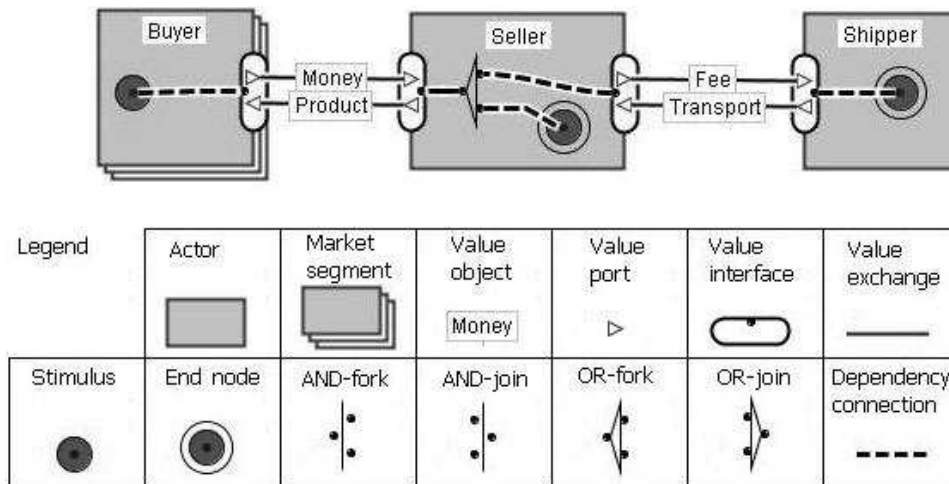
Currently, goals and capabilities have only exact matches. That is, when a capability matches a goal, the goal is replaced by the capability in the goal model. A better utilization of the available libraries requires to

- facilitate inexact matches between goals and capabilities.

Matching capabilities then would not replace the goals but link to these with propagation functions.

Appendix A e^3 -value Modelling Notation

This appendix briefly explains the e^3 -value modelling notation used to model businesses from economic value perspective throughout this thesis. For a detail description of the e^3 -value ontology, we refer to Gordijn, J., & Akkermans, J.M., 2003, Value based requirements engineering: exploring innovative e-commerce idea, Requirements Engineering Journal, Springer Verlag, 8, 2, pp. 114—134. Here, we present an adaptation of the original definitions of the concepts we use.



Example, using e^3 -value modelling notation (Note: the Legend is only for explanatory purposes and is not part of the e^3 -value modelling technique itself)

We illustrate e^3 -value by means of an example (see the figure.) We consider an example with the following businesses taking part: a buyer, a seller, and a shipping company. The seller has a shop and a warehouse at two different locations. It can directly sell products to customers only from the shop. If a product is purchased that is not present in the shop then a delivery from the warehouse must be made. A shipping company is paid to arrange the logistics to the buyer's home. The seller processes two payment methods: (1) in a case of an off-the-shelf product, the seller requires immediate payment in cash; (2) in a case of a purchase from the warehouse, the seller allows late payment by, e.g., a bank transfer.

Below, we explain the semantics of the e^3 -value concepts; we refer to the figure for the graphical representation of these.

- **Actor.** An actor is perceived by its environment as an independent economic (and often also legal) entity. Economically independent refers to the ability of an actor to be profitable after a reasonable period of time (in case of an enterprise), or to increase economic utility for him/herself (in case of an end-consumer). In a

sound and viable business value model every actor should be able to make a profit. Actors are represented as rectangles. Seller and Shipper are examples of actors. Buyer is visualized as stacked actors, denoting a **market segment**. In our interpretation of market segment, actors in a segment attribute equal economic value to objects. The decision to model an entity as an actor or as a segment is determined by the modelling and analysis purpose: e.g. in the figure the motivation can be that we are interested in analyzing potential profit for a chain of companies in relation to an end-customer market segment.

- **Value Object.** Actors exchange value objects. A value object can be a service, a right, a good or even a consumer experience. The important point is that a value object represents a value for one or more actors. Value objects are shown as text in a rectangle. Examples of value objects in the figure are Money, Product, Fee, and Transport.
- **Value Port.** An actor uses a value port to show to its environment that it wants to provide or requests value objects. The concept of a port abstracts away from the internal business processes, and focuses on how external actors and other components of the e-business value model can be 'plugged in'. Ports are shown as small triangles.
- **Value Interface.** Actors have one or more value interfaces. A value interface consists of value ports offering or requesting value objects. It shows the value object(s) an actor is willing to exchange in return for other value object(s). Such willingness is expressed by a decision function on the value interfaces, which shows under what conditions an actor wants to exchange a value object for another value object. The exchange of value objects is atomic at the level of the value interface; i.e. either all exchanges occur as specified by the value interface or none at all. Note that a value interface does not indicate the temporal ordering of objects to be exchanged on its ports. It only indicates which value object is available, in return for some other value object. A value interface is shown by a rounded box, connected to an actor. For example, the value interfaces in the figure connected to the Buyer market segment denotes that a buyer request a product and offer money in return.
- **Value Exchange.** A value exchange is used to connect two value ports with each other. A value exchange represents one or more potential trades of value objects between value ports. As such, it is a prototype for actual trades between actors. It shows which actors are willing to exchange value objects with each other. A value exchange is shown by a line from port to port.

A *dependency path* indicates via which value interfaces objects of value must be exchanged, as a result of a start stimulus, or as result of exchanges via other value interfaces. A dependency path consists of:







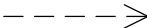
- **Stimulus.** A stimulus represents a consumer need that triggers the chain of exchanges. A dependency path starts with a start stimulus. The model in the figure contains a stimulus inside the market segment Buyer. A stimulus is represented by a bullet.
- **End node.** An end node represents the model boundary: i.e., an end node marks the point beyond which the further exchanges are not important for the analysis. A stop stimulus indicates that a dependency path ends. The model in the figure

contains an end node inside the actors Seller and Shipper. An end node is represented by a bull's eye.




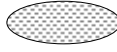

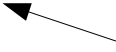
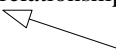
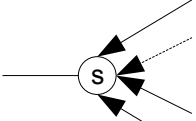
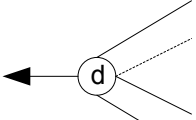



- **AND-fork.** An AND-fork splits a dependency path into two or more sub paths. The exchanges in all sub paths are carried out. The model in the figure does not contain an AND-fork. Look in the legend for a graphical representation.
- **AND-join.** An AND-join collapses sub paths into a single path. It is complementary to the AND-fork. The model in the figure does not contain an AND-join. Look in the legend for a graphical representation.
- **OR-fork.** An OR-fork models a continuation of the dependency path into one direction that is to be chosen from a number of alternatives. One the exchanges in the selected sub path are carried out. The model in the figure contains an OR-fork in the actor Seller.
- **OR-join.** An OR-join merges two or more paths into one path. It is complementary to the OR-fork. The model in the figure does not contain an OR-join. Look in the legend for a graphical representation.
- **Dependency connection.** A dependency connection connects dependency nodes and value interfaces. The dependency connection links the dependency nodes into a dependency path. It is represented by a broken line and the model in the figure contains dependency connection in every actor and market segment.

Appendix B UML Activity Diagram Modelling Notation

The notation used to represent process models is the UML Activity Diagram modelling notation, part of the Unified Modeling Language v1.5 specification. Below, we summarized the symbols we use. The descriptions are adapted from the Unified Modeling Language v1.5 specification [45].

swimlane	Actions may be organized into swimlanes. Swimlanes are used to organize responsibility for actions and subactivities. They often correspond to organizational units in a business model
	An action state is a shorthand for a state with an entry action and at least one outgoing transition involving the implicit event of completing the entry action. A common use of an action state is to model a step in the execution of a workflow process
	Pseudo state representing the beginning of a workflow process
	Pseudo state representing the end of a workflow process
	A simple transition is a relationship between two activities indicating that an instance in the first activity will enter the second activity
	The actions of an action state may be executed more than once concurrently. The number of concurrent invocations is determined at runtime by a concurrency expression, which evaluates to a set of argument lists, one argument list for each invocation
	An activity diagram expresses a decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object. UML provides a shorthand for showing decisions and merging their separate paths back together. Each possible outcome should appear on one of the outgoing transitions
	A dashed arrow is drawn from an action state to an output object, and a dashed arrow is drawn from an input object to an action state. The same object may be (and usually is) the output of one action and the input of one or more subsequent actions
ObjectFlowState	Actions operate by and on objects. These objects either have primary responsibility for initiating an action, or are used or determined by the action

Appendix C Goal Modelling Notation

1.  an oval represents a goal
2.  an oval with bold contour represents a top-level goal
3.   the colour or pattern filling the oval represents goal ownership. In a goal model, all goals in one colour or pattern belong to one actor, role, individual, organization, stakeholder, or business.
4.  a text with in the oval represents one or more of the following: goal number; goal description; goal variable; and evaluation function. For example the text 'G2.1, var7' represent the number of the goal, namely G2.1, and the name of the variable, namely var7.
5.  an arrow with a solid arrow-head represents a positive dependency relationship between two goals
6.  an arrow with a hollow arrow-head represents a negative dependency relationship between two goals
7.  a cycle with a letter s inside, linked to goals with an arbitrary number of incoming arrows with solid arrow-heads and a single line represents a substitution relationship in which the goal linked with a line is substituted by the goals linked with arrows
8.  a cycle with a letter d inside, linked to goals with an arbitrary number of lines and a single outgoing arrow with a solid arrow-head represents a decomposition relationship in which the goal linked with an arrow is decomposed to the goals linked with lines
9.   a text on an arrow or on a line represents the propagation function behind the relationship
10.  a text outside an oval or not on a line or arrow represents a note or comment that may contain, e.g., full description of a goal

Appendix D Value Models of Real-life Businesses

1. Cigarbid	281
2. eBay	282
3. Electronic Courthouse, the.....	283
4. ePier	284
5. Inspire	285
6. MySimon	286
7. Online Resolution	287
8. PriceLine.....	288
9. SellXS.....	289
10. SmartSettle.....	290
11. SquareTrade - California Association of Realtors business model for buyers and sellers	291
12. SquareTrade - California Association of Realtors business model for realtors	292
13. SquareTrade - eBay business model	293
14. SquareTrade - econsumer.gov business model	294
15. SquareTrade - eLance business model.....	295
16. SquareTrade - General business model.....	296
17. SquareTrade - Sony business model	297
18. TEAM.....	298
19. TeleTrade.....	299

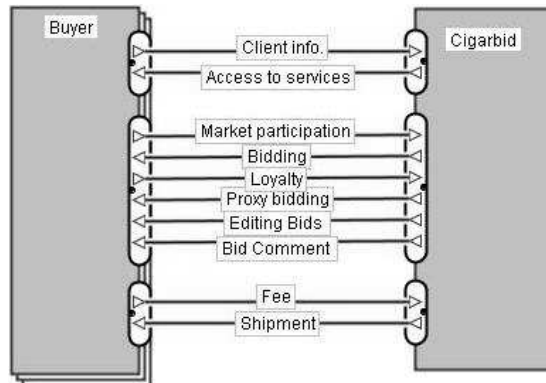
1. Cigarbid

NAME: Cigarbid

SOURCE: <http://www.cigarbid.com/auction/>, visited June 2004

DESCRIPTION: CigarBid.com is a cigar auction warehouse. Every day, it provides live auctions on a variety of products, including cigars, merchandise, 5-packs, samplers, humidors and more. All these products are available to bid on 24 hours per day in several types of auction. Auctions allow for proxy bidding and commenting on every bit in natural language messages.

value model:



2. eBay

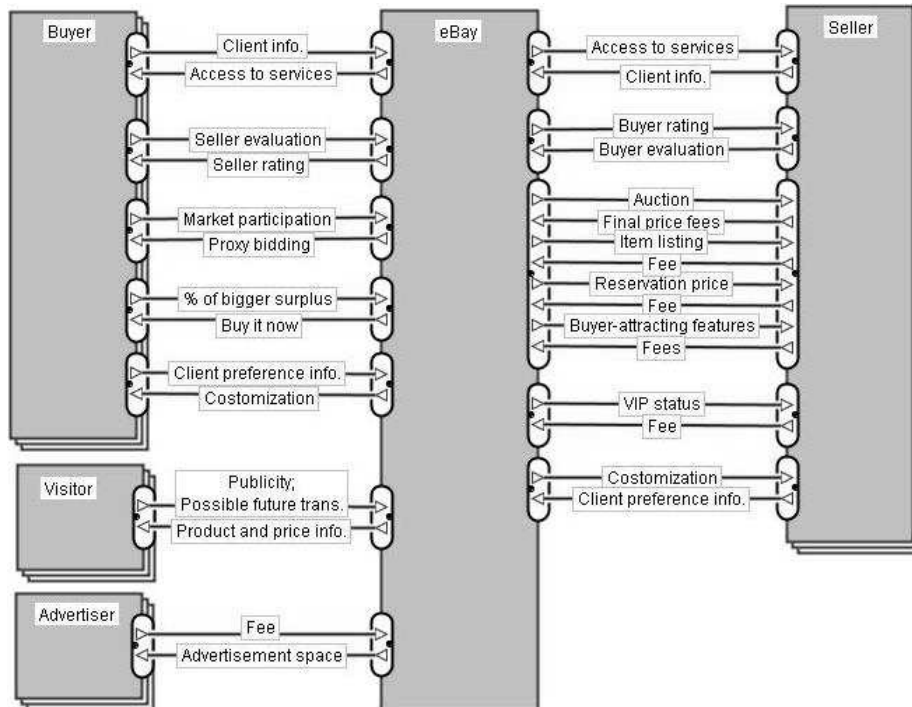
NAME: eBay

SOURCE: <http://www.ebay.com/>, visited June 2004

DESCRIPTION: eBay offers a wide variety of features and services that enable members to buy and sell on the site quickly and conveniently. Buyers have the option to purchase items in auction-style format or items can be purchased at fixed price through a feature called Buy-It-Now. In addition, items at fixed price are also available Half.com, an eBay company.

eBay has numerous services which enhance the trading experience, including marketplace services such as: online payments by PayPal; a wide array of Buyer and Seller tools; and the Developers Program for community members who would like to develop their own solutions. eBay's mission is to provide a global trading platform where practically anyone can trade practically anything.

value model:



3. Electronic Courthouse, the

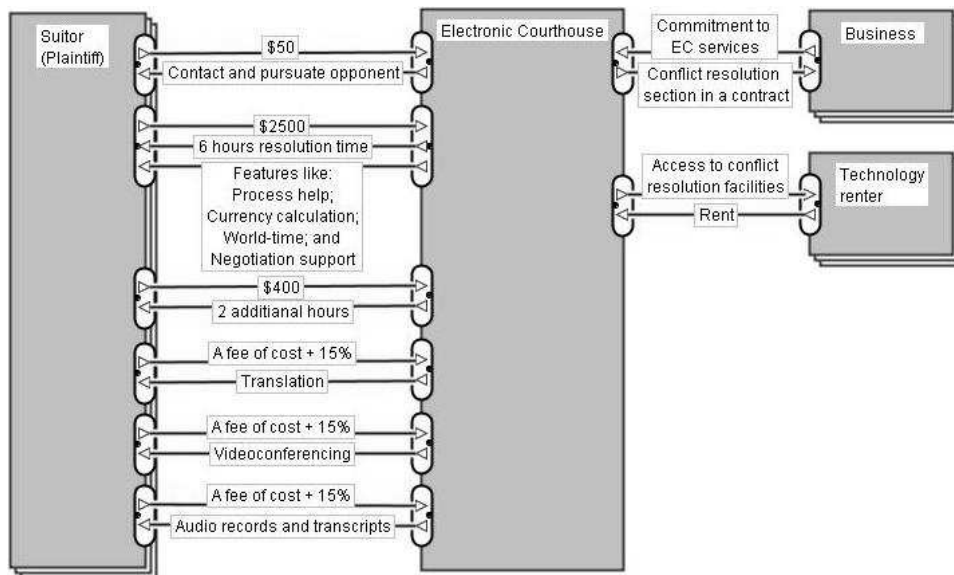
NAME: The Electronic Courthouse

SOURCE: <http://www.electroniccourthouse.com/>, visited June 2004

DESCRIPTION: The Electronic Courthouse provides an online forum for businesses as an alternative to expensive litigation. It allows parties to resolve their commercial disputes fast and remotely. It combines automated information systems and a team of resolution professionals, specializing in, but not limited to, real-estate, insurance, supply chain disputes, breach of contract, sports and entertainment law.

Parties may complete their submissions online, search online legal data base for answers to their legal questions, use translation services, and meet with the other party and their resolution professional in a secure Web-based meeting facility supported by voice conferencing. Parties can choose among several dispute resolution methodologies, including mediation, neutral evaluation, arbitration, or a process agreed to by the parties in a dispute resolution clause in their contract.

value model:



4. ePier

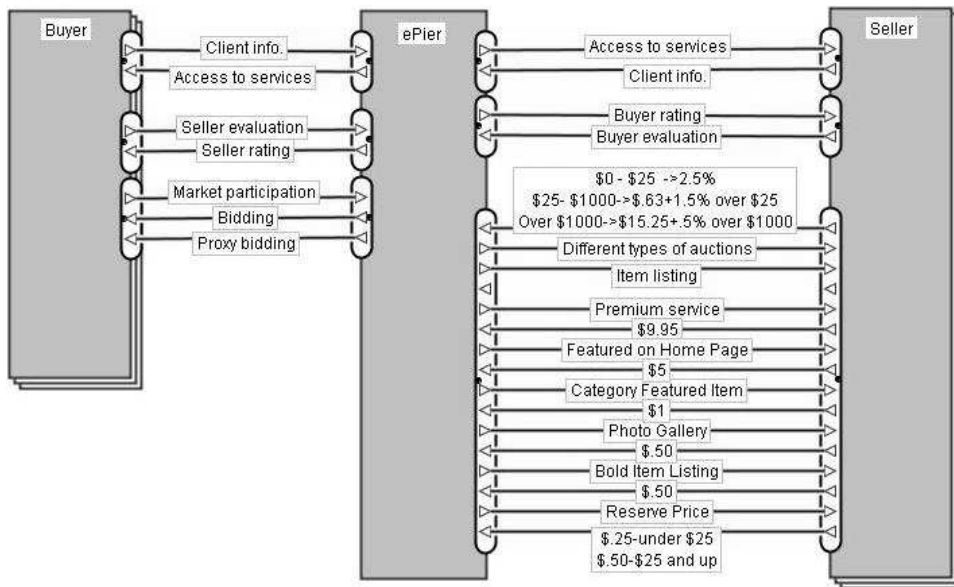
NAME: ePier

SOURCE: <http://www.epier.com/>, visited June 2004

DESCRIPTION: ePier makes it quick and easy for individuals and businesses to buy and sell their products and services, all without expensive marketing costs. At ePier individuals can buy and sell anything from personal items to automobiles, businesses can sell their products to the global community by setting up their own free business site; and service providers can advertise their services and watch the business roll in.

ePier places the competitive focus on quality, price, and speed of delivery. Buyers and sellers alike use our site to find each other, negotiate the deal, and even as a platform for the delivery of goods and services.

value model:



5. Inspire

NAME: Inspire

SOURCE: <http://www.interneg.org/inspire/>, visited June 2004

DESCRIPTION: Inspire has been developed for teaching and research purposes. It provides a detailed description of a negotiation case, which gives information about the importance and alternatives of each issue. Inspire provides services by which offers are constructed and text messages typed. It further offers a rating (score) beside each offer based on preference information and a graph plotting with the history of both sides of the negotiation. Based on the preference information provided, Inspire determines whether an agreement is an optimal one in the sense that no party can improve it without loss to the other side. Inspire can suggest better packages and give the option to continue the negotiation.

value model:



6. MySimon

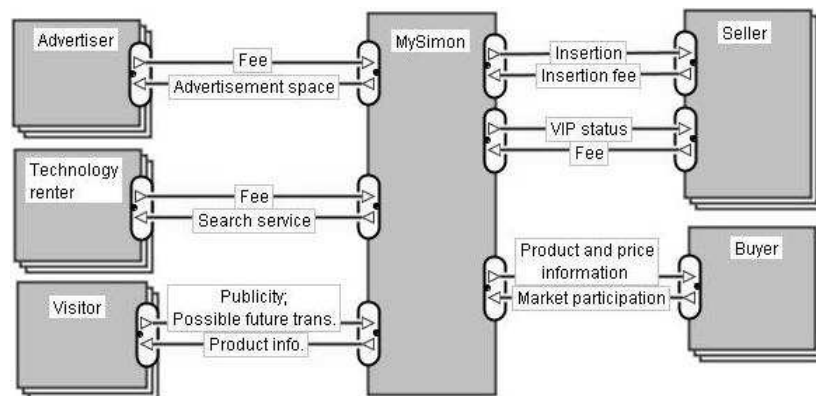
NAME: MySimon

SOURCE: <http://www.mysimon.com/>, visited June 2004

DESCRIPTION: mySimon is a comparison shopping service on the Internet for products and services. It searches thousands of merchants and lists millions of products so that its consumers can compare selections before making a purchase from one of its online merchants.

mySimon is a comparison shopping service. It does not sell or ship anything.

value model:



7. Online Resolution

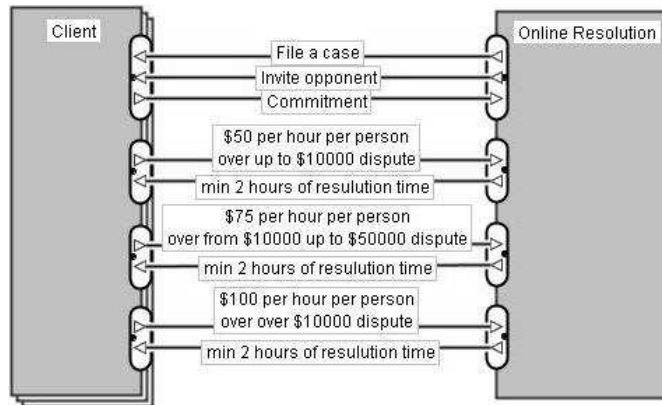
NAME: Online Resolution

SOURCE: <http://www.onlineresolution.com/>, visited June 2004

DESCRIPTION: Online Resolution brings together an extensive panel of professionals with expertise handling e-commerce disputes, insurance claims, business negotiations, and family conflict. Online Resolution provides online four methods for dealing with disputes: negotiation; mediation; expert evaluation; and arbitration.

Online Resolution takes the traditional claims adjusting process and combines it with the dynamic and consensual features of alternate dispute resolution to create a unique resolution option for insurance disputes. Online Resolution brings independent expert adjusters to online. Moreover, Online Resolution adjusters are independent neutrals: they do not represent the insurance company or any party to the dispute. They work with all the parties to uncover the facts and provide a well-reasoned opinion on resolving the case.

value model:



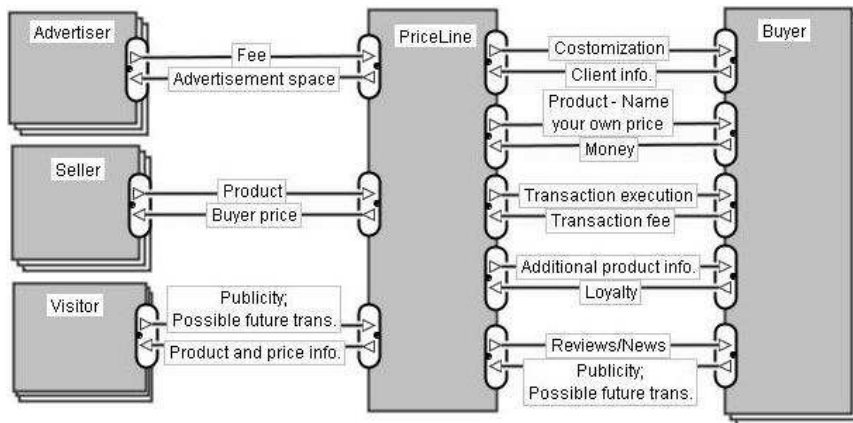
8. PriceLine

NAME: PriceLine

SOURCE: <http://www.priceline.com/>, visited June 2004

DESCRIPTION: Priceline.com offers a broad range of travel services including airline tickets, hotels, rental cars, vacation packages, cruises and more. Priceline.com is a buying service where you can save money by naming your own price. Priceline will take an offer for, for example, name-brand hotels and search to see if any will agree to the price asked. Since thousands of hotel rooms go unsold every night, companies would be willing to consider low prices: it makes sense for them.

value model:



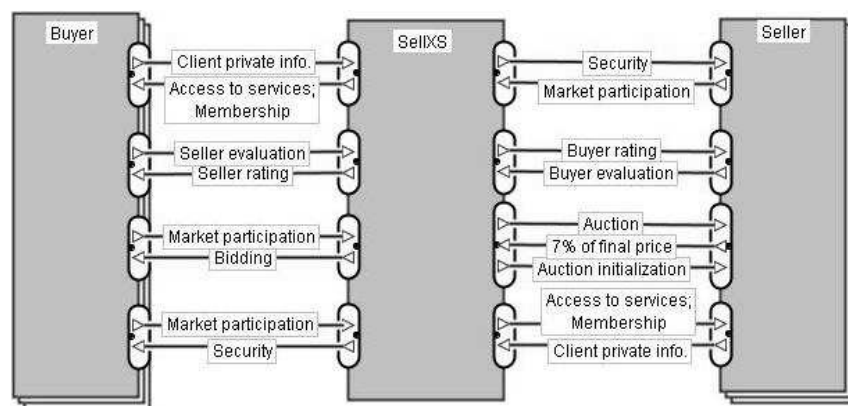
9. SellXS

NAME: SellXS

SOURCE: <http://www.sellxs.com/>, visited June 2004

DESCRIPTION: SellXS.com is a business-to-business (B2B) marketplace for auctioning excess inventory in the semiconductor industry. Liquidating this inventory the traditional way is a daunting challenge that requires extensive research and negotiating with multiple suppliers. SellXS.com enables high-tech companies to buy and sell excess inventory on the open market by leveraging dynamic pricing and a highly scalable eBay-style direct trade model to create greater market efficiencies and cost savings than is possible through traditional or online broker models. SellXS.com facilitates direct trade between buyers and sellers.

value model:



10. SmartSettle

NAME: SmartSettle

SOURCE: <http://www.smartsettle.com/>, visited June 2004

DESCRIPTION: SmartSettle aims to accelerate the negotiation process for any type of case and put decision-makers in control of a process that finds the best possible solutions. SmartSettle offers services to businesses, which have difficulty reaching a good agreement about certain matters.

After registration, parties may engage a single SmartSettle facilitator or each have their own private facilitator. The facilitator helps parties work together to express their interests and identify issues without making specific demands. The facilitator works independently with each party to elicit their initial confidential preferences. Based on party preferences and concessions made by each party, packages are generated for parties to consider. Preferences are refined as needed with help from the facilitator. Optimization is used to generate improvements to the current tentative solution.

value model:



11. SquareTrade - California Association of Realtors business model for buyers and sellers

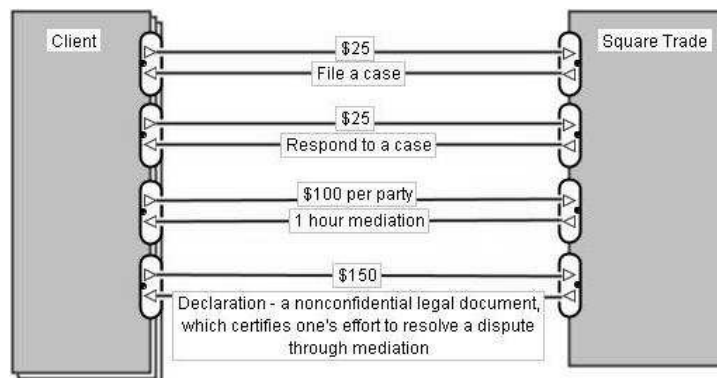
NAME: SquareTrade Realtors f or buyers and sellers

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for California Association of Realtors, a buyer and a seller of a real estate resolve their conflict through SquareTrade's services. This business model is designed for buyer and sellers. The difference with respect to the General business model is in the fees paid by the clients and in one additional service. Fees are assigned to every service, including filing a case and responding to a case. The additional service offers a legal document certifying the effort of one of the sides to resolve the conflict out side of a courthouse.

value model:



12. SquareTrade - California Association of Realtors business model for realtors

NAME: SquareTrade Realtors for realtors

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for California Association of Realtors, real-estate agents resolve their conflict through SquareTrade's services. This business model is designed for real-estate agents. The difference with respect to the General business model is in the fees paid by the clients and in the possibility to prolong the process. Another difference is that the not mediated negotiation service is changed to a second type of mediation.

value model:



13. SquareTrade - eBay business model

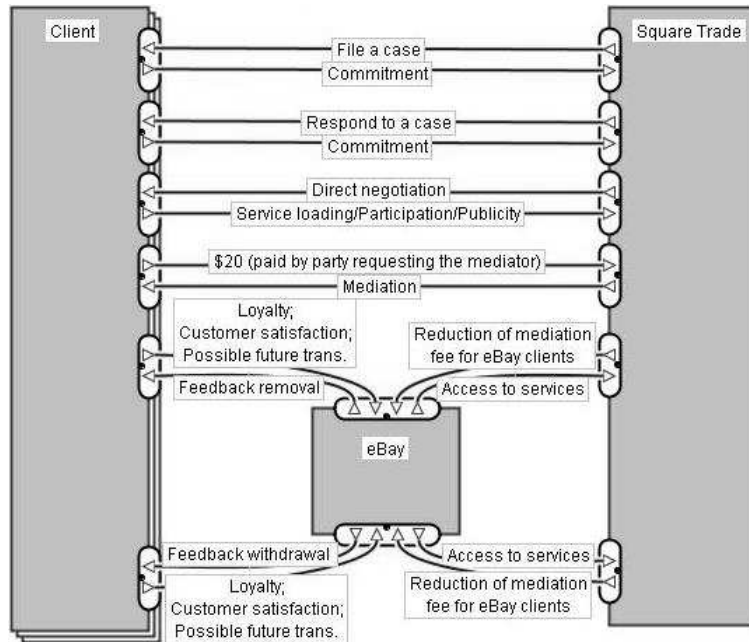
NAME: SquareTrade eBay

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for eBay, clients of eBay resolve their conflict interacting with SquareTrade. In the eBay business model the fee is reduced and fixed per conflict resolution. Additionally, two special features give the possibility to remove or withdraw eBay feedbacks.

value model:



14. SquareTrade - econsumer.gov business model

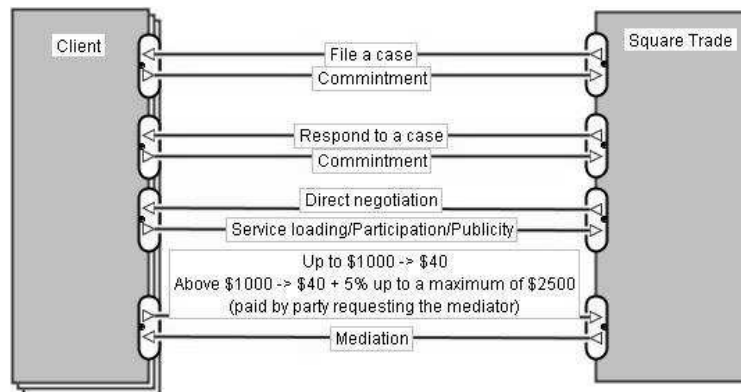
NAME: SquareTrade econsumer.gov

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

The econsumer.gov business model is identical with the General business model, which includes four value exchanges, which correspond to four intermediation services, namely: File a case, Respond to a case, Direct negotiation and Mediation.

value model:



15. SquareTrade - eLance business model

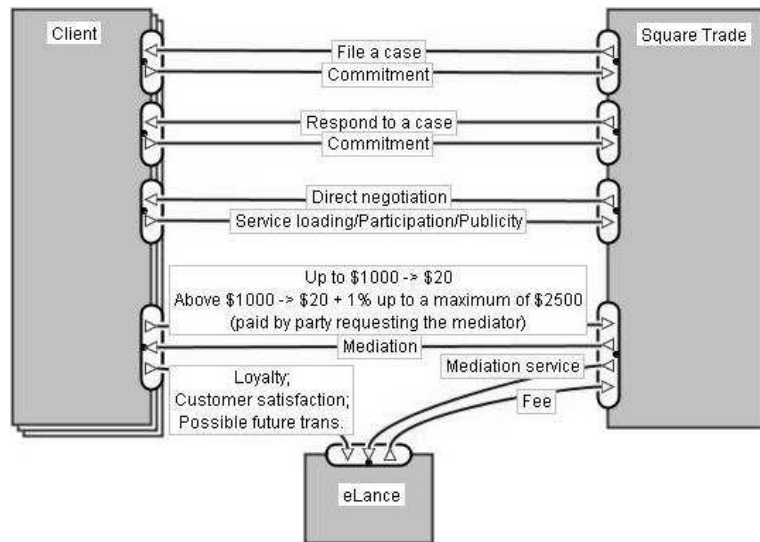
NAME: SquareTrade eLance

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for eLance, clients of eLance resolve their conflict interacting with SquareTrade. eLance does not take part in the process; it is transparent for the clients. The conflict resolution is the same as in the General business model with the exception that parties pay less. The difference in price is covered by eLance.

value model:



16. SquareTrade - General business model

NAME: SquareTrade

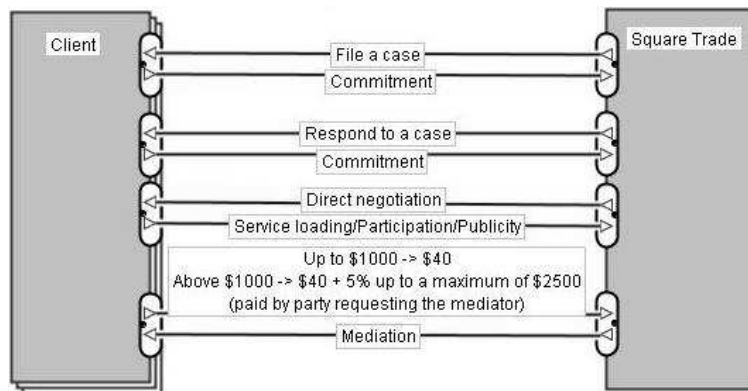
SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

SquareTrade uses a worldwide network of over 250 professional mediators, and is advised by experts in the fields of consumer protection, cyberlaw, and dispute resolution.

SquareTrade has one general business model, which provides a skeleton for derived business models. This model includes four value exchanges, which correspond to four intermediation services, namely: File a case, Respond to a case, Direct negotiation and Mediation.

value model:



17. SquareTrade - Sony business model

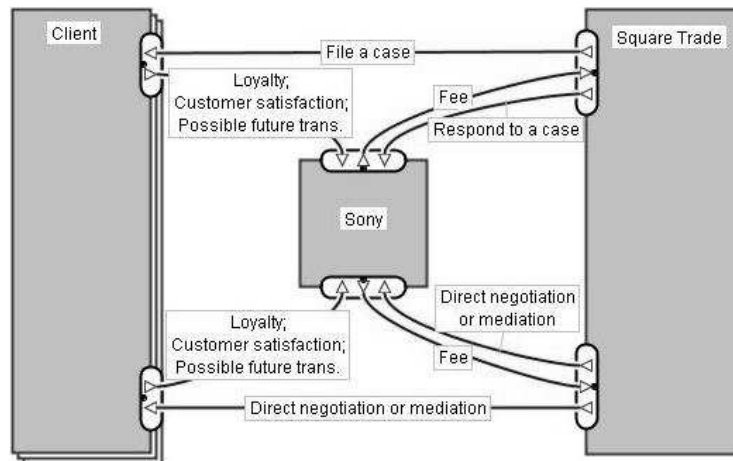
NAME: SquareTrade Sony

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for Sony, all services are free of charge for the client. Moreover, the negotiation and mediation services are not time-restricted. Sony has outsourced its business process of client's claims handling. Sony pays fee to SquareTrade to manage the conflicts with its clients.

value model:



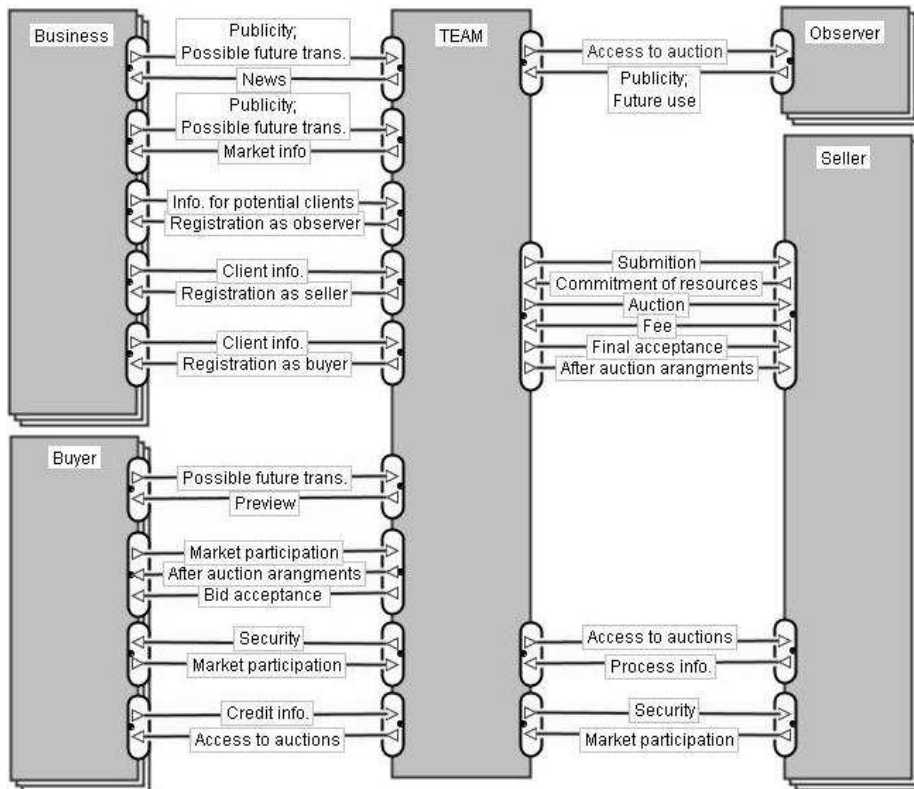
18. TEAM

NAME: TEAM

SOURCE: <http://www.teamauctionsales.com/>, visited June 2004

DESCRIPTION: The Electronic Auction Market (TEAM) is an online, interactive marketplace that brings cattle buyers and sellers together through the power of the Internet. TEAM is a real-time cattle auction with multiple sales weekly. Auctions are held both online, and in conjunction with live sales. TEAM provides buyers with the opportunity to bid on quality strings of cattle they would normally not have access to. TEAM provides maximum exposure to cattle being sold, resulting in maximum dollars for the cattle.

value model:



19. TeleTrade

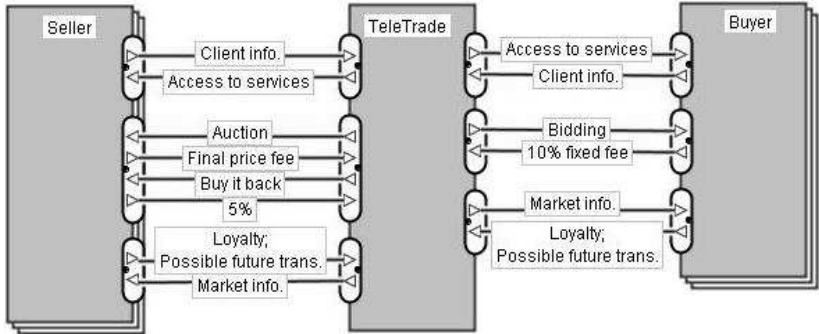
NAME: TeleTrade

SOURCE: <http://www.teletrade.com/>, visited June 2004

DESCRIPTION: Teletrade features Internet bidding and bidding over our toll-free 800 lines using a touch-tone phone from anywhere in North America. All participants are brought together into one auction. Teletrade auctions do not go on for days; they always end the day they start.

Teletrade possesses all lots offered in our auctions, describes each lot accurately, packages, insures and ships all winning bids. All services at Teletrade's web site are free. One only pays a fee when one buys or sells an item.

value model:



Appendix E Library of Value Patterns

1. Advertising	301
2. Client Connection	303
3. Core Service Extension.....	305
4. Insourcing	307
5. Market Segmentation.....	309
6. Rating	311
7. Registration.....	313
8. Reservation	315
9. Risk Reduction	317
10. Screening	319
11. Technology Renting.....	322

1. Advertising

NAME: Advertising

CODE: ADV

HEADLINE: An intermediary offers to broadcast a message to many receivers.

CONTEXT: A business, i.e. an advertiser, has a new or unknown product. It is interested in advertising the product to a great number of potential buyers. An intermediary has access to many clients. Moreover, it has a database with client information.

DESCRIPTION: An Intermediary offers a product¹ to a number of clients for free: i.e., the clients do not have to pay money to consume the product. What the intermediary gets in return is the attention of the clients. The intermediary makes money by bundling its product with an advertisement of a third party seeking exposure to potential clients. The Intermediary is able to target the advertisements as it has information about its clients' preferences.

PROBLEM:

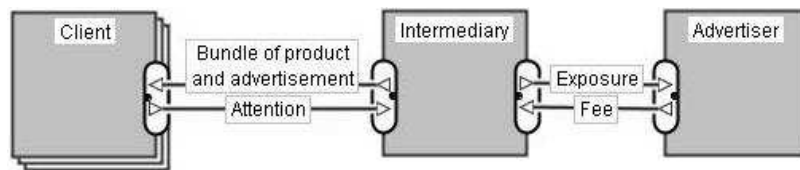
- roles:
- The pattern includes three roles played by three different businesses:
 - *Client* is the role played by the consumers of products provided by the *Intermediary*;
 - *Intermediary* is the role played by the business that provides a product bundled with an advertisement; and
 - *Advertiser* is the role played by the business seeking publicity.

forces:

Role	Force	
Client	F1	Consume products for free
Intermediary	F2	Profit from access to and knowledge about clients
Advertiser	F3	Reach potential clients

SOLUTION: An intermediary broadcasts targeted advertisements to its clients in return for a fee.

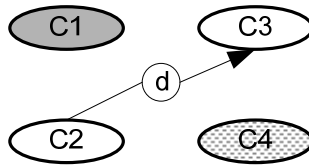
value model:



¹ The term product includes also services.

CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Receive products for free	Client	F1
C2 Bundle products and advertisements	Intermediary	F2
C3 Receive money for exposing Advertiser to Clients	Intermediary	F2
C4 Get access to potential clients	Advertiser	F3

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Receive products for free		
a1	Receive products	Boolean	true
C2	Bundle products and advertisements		
a2	Bundled product and advertisement	Boolean	true
C3	Receive money for exposing Advertiser to Clients		
a3	Received money	Boolean	true
C4	Get access to potential clients		
a4	Access to clients	Boolean	true

OCCURRED IN:

- eBay
- PriceLine
- MySimon

2. Client Connection

NAME: Client connection

CODE: CLC

HEADLINE: An intermediary develops relationship with its clients.

CONTEXT: An intermediary is situated in a market with many competitors and loosely connected clients with buying power.

DESCRIPTION: To win the loyalty of its clients, an intermediary offers additional services or products or both. By doing that, it differentiates from the competitors and increases the switching costs for its clients. The clients remain loyal because of the extra value they receive.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

- *Intermediary* is the role played by a business that mediates certain transactions;
- *Client* is the role played by businesses using the mediation services of *Intermediary*. *Clients* have buying power.

forces:

Role	Force	
Intermediary	F1	Keep existing clients
	F2	Influence clients decisions
Client	F3	Get additional value for its buying power

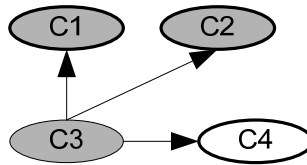
SOLUTION: An intermediary gives its clients useful information about itself, product reviews, market news or recommendations in return for possible future transactions, loyalty and publicity.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Keep existing clients	Intermediary	F1
C2 Influence clients decisions	Intermediary	F2
C3 Provide clients with information about itself, products, services, and market analysis	Intermediary	
C4 Receive additional value	Client	F3

operationalization:

<i>Capability code</i>		<i>Capability</i>	
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Keep existing clients		
a1	Returning new clients	Percentage per month	76
C2	Influence clients decisions		
a2	Follow-ups from newsletters	Percentage sent newsletters	3
C3	Provide clients with information about itself, products, services, and market analysis		
a3	Newsletters	Natural numbers / per month	2
C4	Receive additional value		
a4	Perceived value	{spam, no value, useful}	useful

OCCURRED IN:

- TEAM
- PriceLine
- TeleTrade

3. Core Service Extension

NAME: Core service extension

CODE: CSE

HEADLINE: A Service provider offers additional features to a core service in order to achieve better customer satisfaction.

CONTEXT: A Service provider is situated in a market with many competitors that offer the same service. Differentiation is needed to get a competitive advantage.

DESCRIPTION: The pattern describes a service provider that offers complementary services (or products) to its core product (or service). The additional services are optional and are delivered for an additional fee. The product and the services are bundled. The benefits for the service provider are the better satisfaction of client's need; whereas the client gets the additional service at discount prices.

PROBLEM:

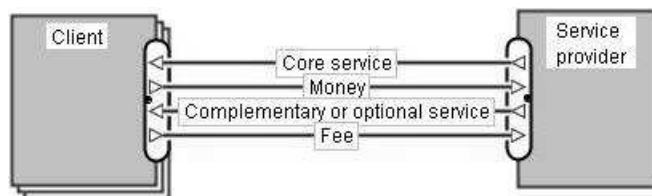
- roles:
- The pattern includes two roles played by two different businesses:
- *Service provider* is the role played by the business that offers the services;
 - *Client* is the role played by a business that consumes the services.

forces:

Role	Force	
Service provider	F1	Achieve better customer satisfaction
	F2	Differentiate from competitors
Client	F3	Experience customized services

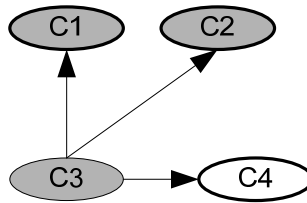
SOLUTION: A Service provider offers its core service in return of money. In addition, it bundles the core service with complementary or optional services for which the Client pays with additional fees.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Achieve better customer satisfaction	Service provider	F1
C2 Differentiate	Service provider	F2
C3 Offer complementary and optional services	Service provider	
C4 Receive customized services	Client	F3

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Achieve better customer satisfaction		
a1	Customer satisfaction	{ dissatisfied, neutral, satisfied }	satisfied
C2	Differentiate		
a2	Differentiated	Boolean	true
C3	Offer complementary and optional services		
a3	Relevance of services	{ useless, desired, valuable }	desired
C4	Receive customized services		
a4	Customized services	Boolean	true

OCCURRED IN:

- Inspire
- Cigarbid
- eBay
- ePier
- SellXS
- TEAM
- TeleTrade
- SmartSettle
- The Electronic Courthouse
- SquareTrade – General business model
- SquareTrade – eBay business model
- SquareTrade – econsumer.gov business model
- SquareTrade – eLance business model

4. Insourcing

NAME: Insourcing

CODE: INS

HEADLINE: An intermediary offers its core service to be executed on behalf of another business.

CONTEXT: An intermediary offers a specialized service which requires an interaction with the clients of the outsourcing business.

DESCRIPTION: The pattern describes an intermediary that delivers a product¹ on behalf of another business. The product is consumed by a client of the business. The benefit for the intermediary is the access to clients and the fees paid by the business. The business is willing to give away part of its business in return for better provision which will lead to better satisfaction of its clients.

PROBLEM:

roles: The pattern includes three roles played by three different businesses:

- *Intermediary* is the role played by the business that insources the provision of a product;
- *Business* is the role played by the business that wants to outsource the provision of a product.
- *Client* is the role played by a client of *Business* which needs the product delivered by *Intermediary*.

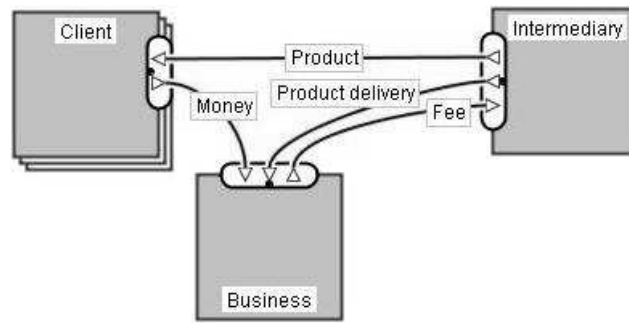
forces:

Role	Force	
Intermediary	F1	Increase usage of its core services to better utilize its special knowledge
Business	F2	Satisfy better clients' needs by better delivery of a product
Client	F3	Satisfy needs for a high quality product

SOLUTION: An intermediary offers its core service (delivery of a particular product) to the client(s) of a business. In return, it receives fees from the business. The intermediary serves directly the clients of the business but it is contracted by the business. The business sells the delivered product directly to its clients.

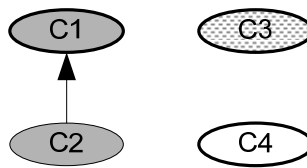
¹ The term product includes also services.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Increase business	Intermediary	F1
C2 Insource the provision of a product	Intermediary	F1
C3 Outsource the provision of a product	Business	F2
C4 Receive a product	Client	F3

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Increase business		
a1	Insource business	Boolean	true
C2	Insource the provision of a product		
a2	Insource product delivery	Boolean	true
C3	Outsource the provision of a product		
a3	Outsource product delivery	Boolean	true
C4	Receive a product		
a4	Received product	Boolean	true

OCCURRED IN:

- SquareTrade – Sony business model
- SquareTrade – eLance business model

5. Market Segmentation

NAME: Market segmentation

CODE: SEG

HEADLINE: A provider offers two version of its product for consumers with different needs.

CONTEXT: The market of a particular product consists of two segments of consumers. A product provider wants to customize its product for the particular needs.

DESCRIPTION: The pattern describes a provider that offers the delivery of a product¹ in two different versions. Each version is tailored to the particular needs of the clients. Versions differ in price and complements to the main product. The clients of the two versions form two segments.

PROBLEM:

- roles:
- The pattern includes three roles played by three different businesses:
- *Provider* is the role played by the business that provider the product;
 - *Basic client* is the role played by a business with limited needs from or no experience with the product.
 - *Experienced client* is the role played by a business which demands more from the product and is willing to pay a premium to get extra features.

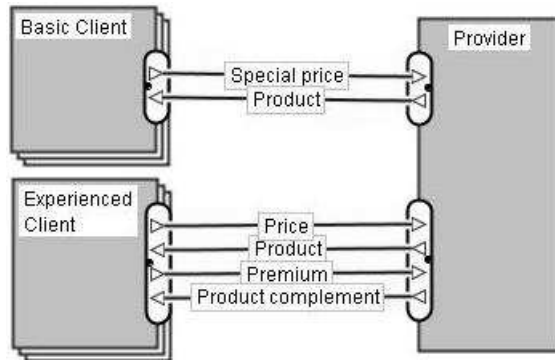
forces:

Role	Force	
Provider	F1	Charge both basic and experienced clients
	F2	Justify the higher price for the complete product
Basic client	F3	Pay less and get the basic product
Experienced client	F4	Utilize the complete set of possibilities offered with the product

SOLUTION: A provider offers its product in two different versions to two market segments consisting of clients with different needs. The exchanged value object with the various segment differ in number and quality.

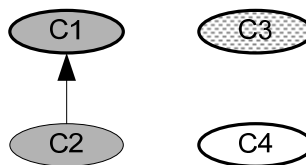
¹ The term product includes also services.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Fragment clients	Provider	F1
C2 Version products	Provider	F2
C3 Pay the lowest price	Basic client	F3
C4 Receive optimal utility	Experienced client	F4

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Fragment clients		
a1	Fragmented market	Boolean	true
C2	Version products		
a2	Versions	Natural numbers	2
C3	Pay the lowest price		
a3	Receive basic product	Boolean	true
C4	Receive optimal utility		
a4	Customized product	Boolean	true

OCCURRED IN: models not listed in Appendix D Value Models of Real-life Businesses.

6. Rating

NAME: Rating

CODE: RAT

HEADLINE: An intermediary offers means to its clients to evaluate each other.

CONTEXT: An Intermediary facilitates deals between clients that do not know each other. The clients are conditionally divided into two types: opportunistic clients and insecure clients. In one transaction a client can be both types. The opportunistic client takes advantage of its partner and do not fulfil its contractual obligations; the insecure client fears transacting with a partner with unknown reputation.

DESCRIPTION: An Intermediary provides the insecure clients (see CONTEXT for definition of insecure client) with a rating of their potential business partners. The rating is calculated from the feedback given by every client at the end of every transaction mediated by the Intermediary.

PROBLEM:

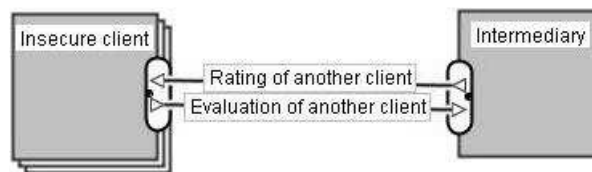
- roles: The pattern includes two roles played by two different businesses:
- *Intermediary* is the role played by the business that provides the contacts of the transacting businesses;
 - *Insecure client* is the role played by one of the transacting businesses that worries about fraud.

forces:

Role	Force	
Intermediary	F1	Reduce anonymity of clients
	F2	Stay independent and neutral
Insecure client	F3	Know more about prospective partners prior to transacting
	F4	Have means to counter fraud

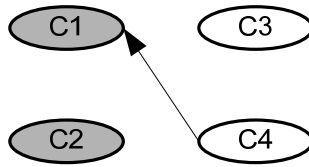
SOLUTION: An intermediary offers a possibility to clients to rate each other after a particular transaction. In return, clients receive an accumulated rating number of past evaluations.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1	Reduce anonymity	Intermediary
C2	Do not evaluate clients	Intermediary
C3	Receive rating of partners	Insecure client
C4	Evaluate transacting partners	Insecure client

operationalization:

<i>Capability code</i>	<i>Capability</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Reduce anonymity		
a1	Ratings based on more than 10 evaluations	Percentage of all clients	48
C2	Do not evaluate clients		
a2	Evaluation neutral	Boolean	true
C3	Receive rating of partners		
a3	Received rating	Boolean	true
C4	Evaluate transacting partners		
a4	Evaluated transactions	Percentage evaluated transaction	57

OCCURRED IN:

- eBay
- ePier
- SellXS

7. Registration

NAME: Registration

CODE: REG

HEADLINE: An intermediary registers its clients to be able to differentiate them later.

CONTEXT: An intermediary is situated in a market with anonymous buyers and sellers. The intermediary needs to know its clients to provide better services.

DESCRIPTION: An Intermediary requires from its clients identification information to be able to keep track of them and, in that way, to offers them better-targeted services. Every client provides enough information to be classified in different market segment and its buying behaviour to be monitored.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

- *Intermediary* is the role played by the market creator;
- *Client* is the role played by businesses using the services of *Intermediary*. *Clients* have buying power.

forces:

Role	Force	
Intermediary	F1	Identify clients to know them better
Client	F2	Get access to services

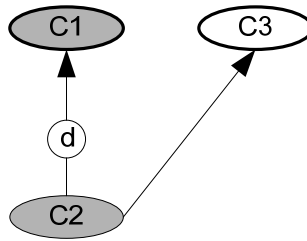
SOLUTION: An intermediary requires a registration from every new client and identification prior to the use of its services. The registration requires client information.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Collect client information	Intermediary	F1
C2 Require registration before allowing access to services	Intermediary	F1
C3 Receive access to services	Client	F2

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Collect client information		
a1	Registered clients	Percentage registered clients of all	75
C2	Require registration before allowing access to services		
a2	Required registration	Boolean	true
C3	Receive access to services		
a3	Clients willing to register	Percentage of all potential clients	78

OCCURRED IN:

- Cigarbid
- eBay
- ePier
- SellXS
- TEAM
- TeleTrade

8. Reservation

NAME: Reservation

CODE: RES

HEADLINE: A business requires a reservation in advance of a particular product in order to secure resources needed for the delivery

CONTEXT: A business offers a product or service which is expensive to produce or not durable. The number of consumers cancelling orders is significant.

DESCRIPTION: A business offers a product which it does not keep in stock. In case the consumer cancels the order during the production of the product, the resources put into the process are lost. The business wants to secure that the ordered products are paid; therefore, it requires payment in advance.

PROBLEM:

roles: The pattern includes two roles played by two different businesses.

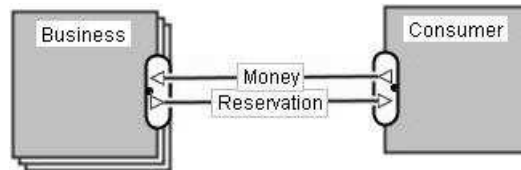
- *Business* is the role played by the business that offers the product;
- *Consumer* is the role played by the consumer of the product.

forces:

Role	Force
Business	F1 Secure resource used to produce the product
Consumer	F2 Receive guarantees that the product will be delivered

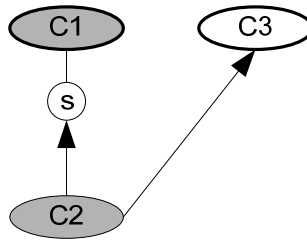
SOLUTION: A business requires payment in advance, which reserves the production resource and the final product for a particular consumer.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Secure resources	Business	F1
C2 Pay in advance	Business	
C3 Reserve product	Consumer	F2

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Secure resources		
a1	Secured resources	Boolean	true
C2	Pay in advance		
a2	Paid in advance	Boolean	true
C3	Reserve product		
a3	Reserved product	Boolean	true

OCCURRED IN: models not listed in Appendix D Value Models of Real-life Businesses.

9. Risk Reduction

NAME: Risk reduction

CODE: RSK

HEADLINE: A business requires a reservation in advance of a particular product in order to secure resources needed for the delivery.

CONTEXT: A business offers a product which is not kept in stock. If a buyer withdrew from the exchange after ordering the product then the resource used to produce the product would be lost. The business accept guaranties from third parties that the ordered products will be paid.

DESCRIPTION: A business offers a product which it does not keep in stock. In case the consumer cancels the order during the production of the product, the resources put into the process are lost. The business wants to secure that the ordered products are paid; therefore, it accepts an assurance in a form of a credit letter from a party that it trusts.

PROBLEM:

roles: The pattern includes three roles played by three different businesses.

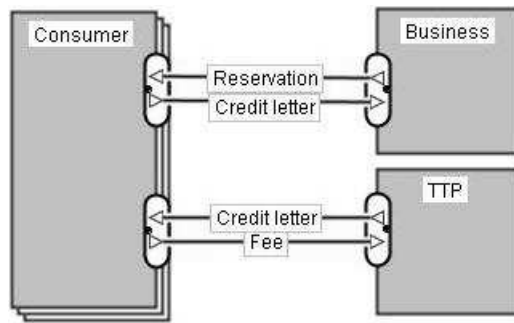
- *Business* is the role played by the business that offers the product.
- *Consumer* is the role played by the consumer of the product.
- *Trusted third party (TTP)* is the role played by a bank or a certification authority which guarantee that certain aspects of a business transaction are secure, e.g. *Consumer's* payment.

forces:

Role	Force	
Business	F1	Secure resource used to produce the product
Consumer	F2	Receive guarantees that the product will be delivered
	F3	Pay as late as possible
TTP	F4	Reduce risk

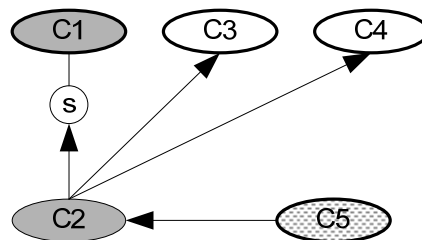
SOLUTION: A trusted third party (TTP) issues a statement (a credit letter) that a consumer has money to pay and will pay after a product is delivered. The consumer pays a fee to the TTP which, in return, issues a credit letter. The consumer gives the credit letter to the business to make a reservation. The business accepts the credit letters of the TTP and reserves the production capacity for the consumer.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Secure resources	Business	F1
C2 Accept guaranties from a third party	Business	
C3 Reserve product	Consumer	F2
C4 Pay after product delivery	Consumer	F3
C5 Issue guaranties	TTP	F4

operationalization:

Capability code	Capability	Variable code	Variable	Variable domain	Value
C1	Secure resources	a1	Secured resources	Boolean	true
C2	Accept guaranties from a third party	a2	Guaranties accepted	Boolean	true
C3	Reserve product	a3	Reserved product	Boolean	true
C4	Pay after product delivery	a4	Paid after delivery	Boolean	true
C5	Issue guaranties	a5	Issued guaranties	Boolean	true

OCCURRED IN: models not listed in Appendix D Value Models of Real-life Businesses.

10. Screening

NAME: Screening

CODE: SCR

HEADLINE: An intermediary filters market participants to guarantee some level of security.

CONTEXT: An Intermediary facilitated deals between businesses that do not know each other. The businesses are conditionally divided into two types: opportunistic businesses and insecure businesses. (In one transaction a business can be both types). The opportunistic business takes advantage of its partner and do not fulfil its contractual obligations; the insecure business fears transacting with a partner with unknown reputation.

DESCRIPTION: An Intermediary requires from potential market participants private information about their credibility. On the basis of this information, it decides if they are eligible for transacting with other, already registered, businesses. The intermediary ensures the credibility of businesses: this gives security and guarantees to all market participants.

PROBLEM:

roles: The pattern includes three roles played by three different businesses:

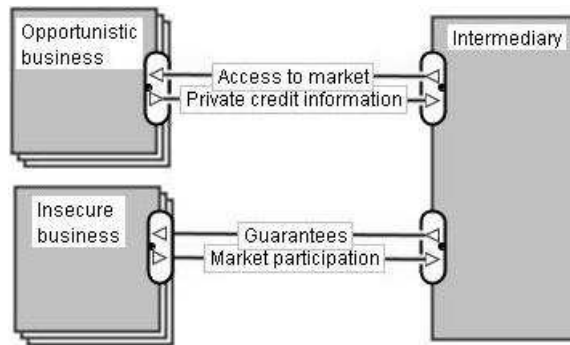
- *Intermediary* is the role played by the market creator;
- *Opportunistic business* is the role played by a business that acts opportunistically.
- *Insecure business* is the role played by a business that worries about fraud.

forces:

Role	Force	
Intermediary	F1	Increase market size
	F2	Provide trustworthy environment
	F3	Protect the interests of all participants
Opportunistic business	F4	Protect private information
	F5	Get access to markets
Insecure business	F6	Receive security guarantees about the credibility of prospective partners

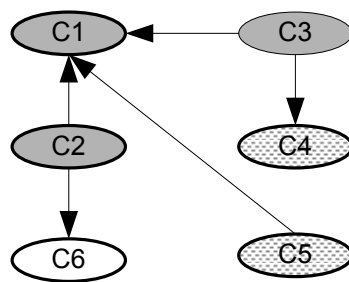
SOLUTION: An intermediary screens the potential market participants and allows access only to the credible ones. The intermediary requires from businesses private credit information. In return, businesses receive guarantees for trustworthiness of market participants. The intermediary receives participation in transactions from businesses.

value model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Get more member businesses	Intermediary	F1
C2 Screen businesses	Intermediary	F2
C3 Keep private credit information secret	Intermediary	F3
C4 Give private credit information only to Intermediary	Opportunistic business	F4
C5 Satisfy the requirements of the intermediary	Opportunistic business	F5
C6 Receive guarantees of partner's credibility	Insecure business	F6

operationalization:

<i>Capability code</i>	<i>Capability</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Get more member businesses		
a1	Clients approving the screening	Percentage	67
C2	Screen businesses		
a2	Screened businesses	Boolean	true
C3	Keep private credit information secret		
a3	Secret credit information	Boolean	true
C4	Give private credit information only to Intermediary		
a4	Clients willing to register	Percentage of all potential clients	91
C5	Satisfy the requirements of the intermediary		
a5	Compliant clients	Percentage of registered clients	98
C6	Receive guarantees of partner's credibility		
a6	Received guarantees	Boolean	true

OCCURRED IN:

- SellXS
- TEAM

11. Technology Renting

NAME: Technology renting

CODE: TCH

HEADLINE: A business offers for rent a technology which it uses to produce its core service.

CONTEXT: A company possesses a technology for the delivery of a service which has a bigger market that the company can serve. The technology has a short life-span; therefore, it is beneficial to make it available to competitor.

DESCRIPTION: A business owns a special or patented technology which it uses to produce its core services. The market is potentially big and other businesses want to offer similar services. The technology owner is willing to rent the technology to other businesses which possibly include direct competitors. Businesses pay royalties per each use of the technology.

PROBLEM:

roles: The pattern includes two roles played by two different businesses.

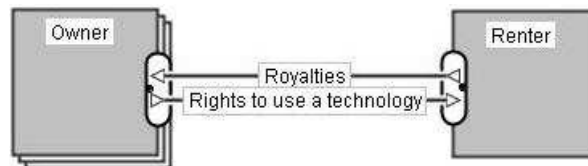
- *Owner* is the role played by the business that possesses the technology;
- *Renter* is the role played by the business that rents the technology to produce its own services.

forces:

Role	Force	
Owner	F1	Better utilization of a technology
	F2	Fast return on investment
	F3	Control over the intellectual property of the technology
Renter	F4	Risk-free entry to a new market

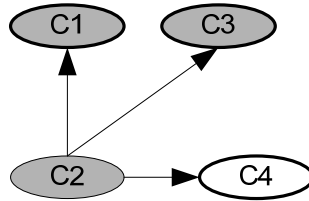
SOLUTION: The owner of a technology gives access for use to other businesses (the renters) in return for royalties. The renters of the technology are free to produce even competing products.

value model:



CAPABILITIES:

capability model:



Capability		Benefiting role	Resolving force
C1	Execute more transactions	Owner	F1
C2	Rent technology	Owner	F2
C3	Keep the technology in secret	Owner	F3
C4	Do not invest in development of technology	Renter	F4

operationalization:

<i>Capability code</i>	<i>Capability</i>		
<i>Variable code</i>	<i>Variable</i>	<i>Variable domain</i>	<i>Value</i>
C1	Execute more transactions		
a1	Renter's transactions	Percentage	85
C2	Rent technology		
a2	Rented technology	Boolean	true
C3	Keep the technology in secret		
a3	Secret technology	Boolean	true
C4	Do not invest in development of technology		
a4	No investment	Boolean	true

OCCURRED IN:

- The Electronic Courthouse
- MySimon

Appendix F Process Models of Real-life Businesses

1. Aalsmeer Flower Auction.....	325
2. APX	327
3. Cigarbid	330
4. eBay	332
5. Electronic Courthouse, the.....	334
6. MySimon	337
7. Online Resolution	339
8. PriceLine.....	341
9. SellXS.....	343
10. SquareTrade - Sony business model.....	345
11. TEAM.....	347
12. TeleTrade.....	350

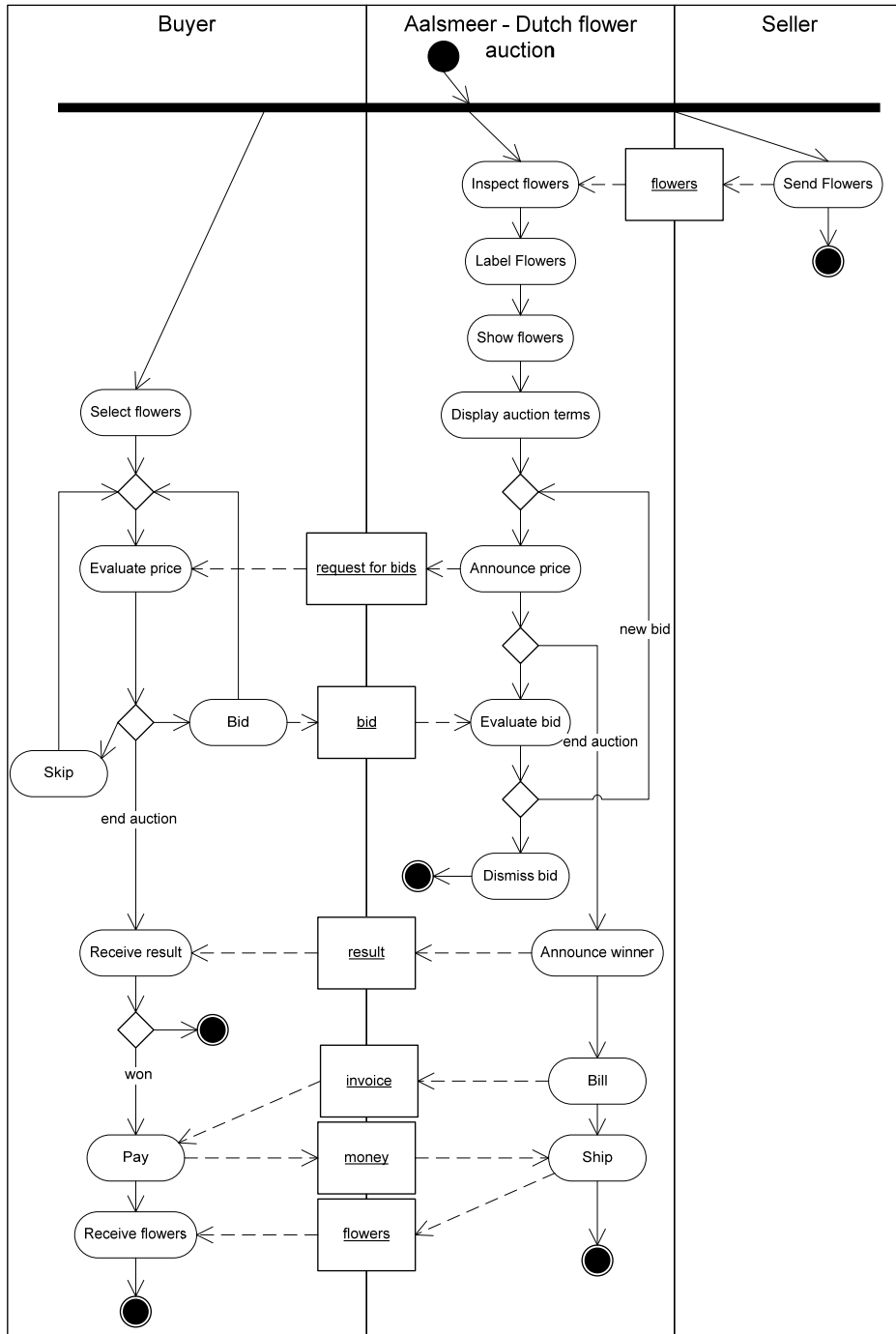
1. Aalsmeer Flower Auction

NAME: Aalsmeer Flower Auction

SOURCE: <http://www.vba-aalsmeer.nl/>, visited June 2005

DESCRIPTION: Aalsmeer Flower Auction offers globally producing growers and globally active wholesalers and exporters a total concept: a central marketplace for the buying and selling of floricultural products with a balanced range of marketing channels, good facilities for growers and buyers and effective logistics. Aalsmeer, the most prominent auction in the world, thus contributes significantly to processes of distributing and pricing flowers and plants.

process model: Auctioning



2. APX

NAME: APX

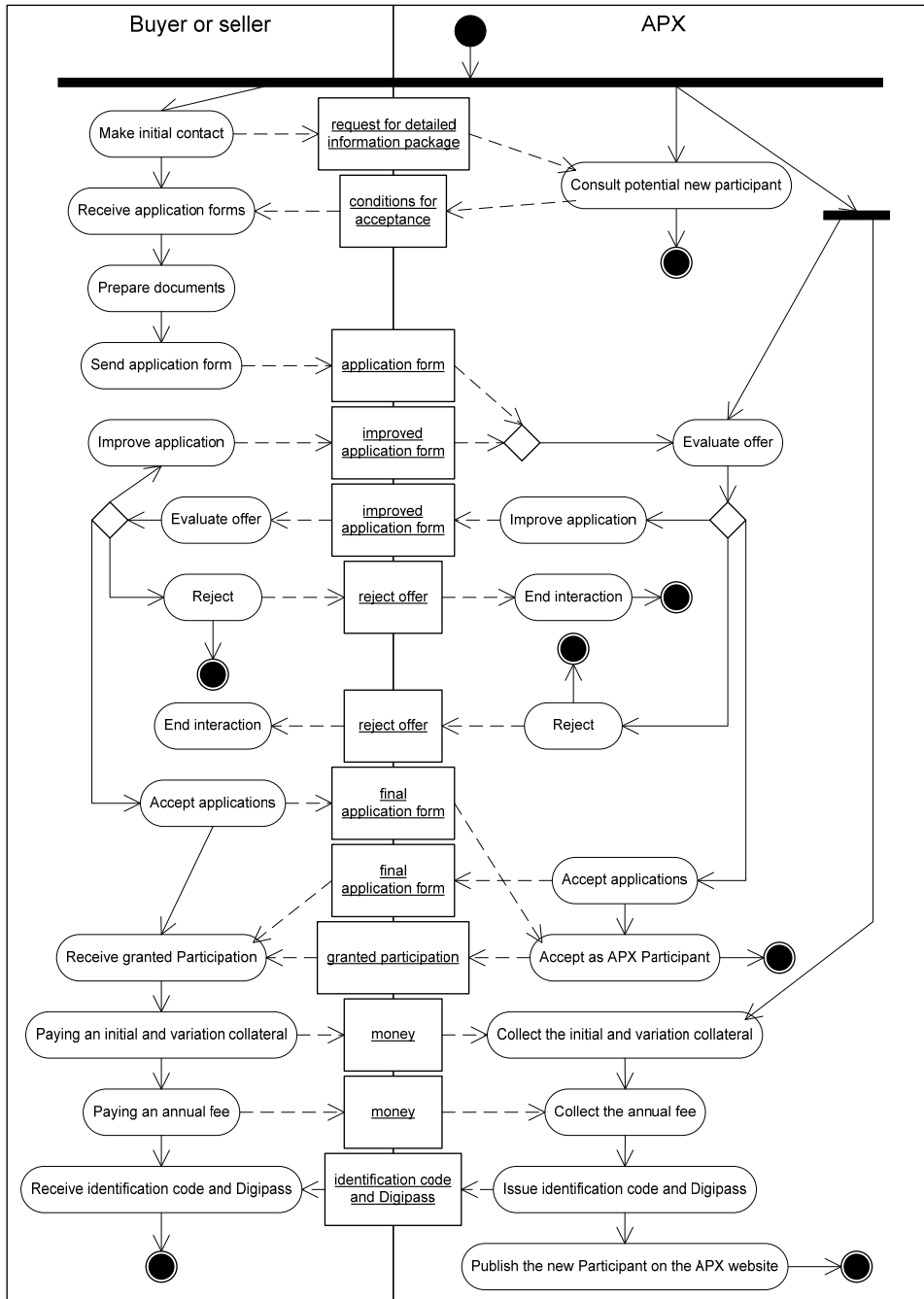
SOURCE: <http://www.apxgroup.com/>, visited June 2004

DESCRIPTION: APX is Europe's premier provider of power and gas exchanges, operating markets in the Netherlands, the UK and Belgium.

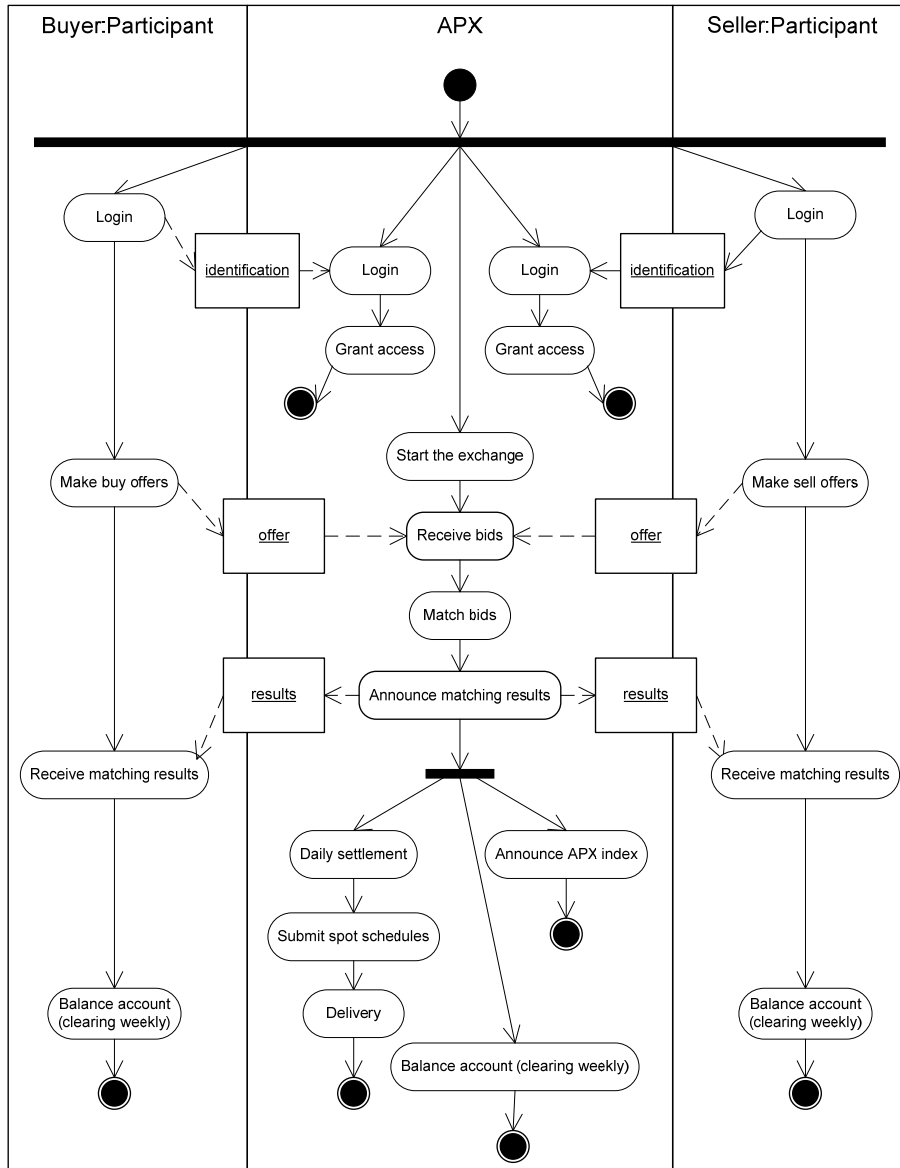
APX Group facilitates the development of the liberalized integrated energy markets in North West Europe by providing an efficient, transparent and secure electronic trading environment for gas and power, by calculating indices and by delivery of external services

APX B.V. is an independent fully electronic exchange for anonymous trading on the spot market. The spot market has been operational since May 1999 offering distributors, producers, traders, brokers and industrial end-users a spot market trading platform in the form of day-ahead transactions: trading today for delivery of electricity tomorrow. APX provides its members standardized products to sell and purchase and is the central counter party in all electricity trades.

process model: Registration



process model: Trading



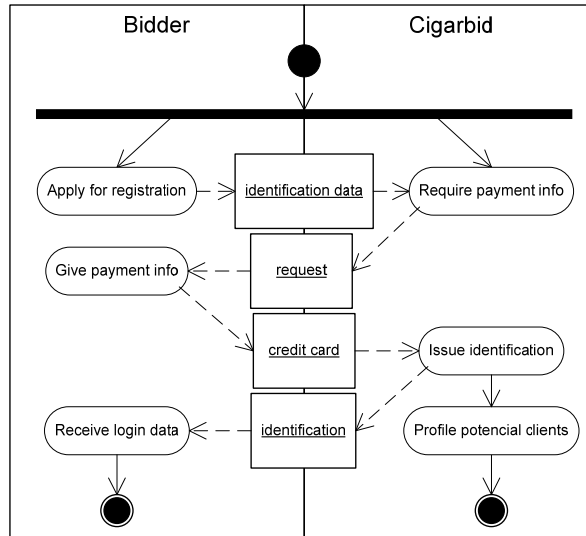
3. Cigarbid

NAME: Cigarbid

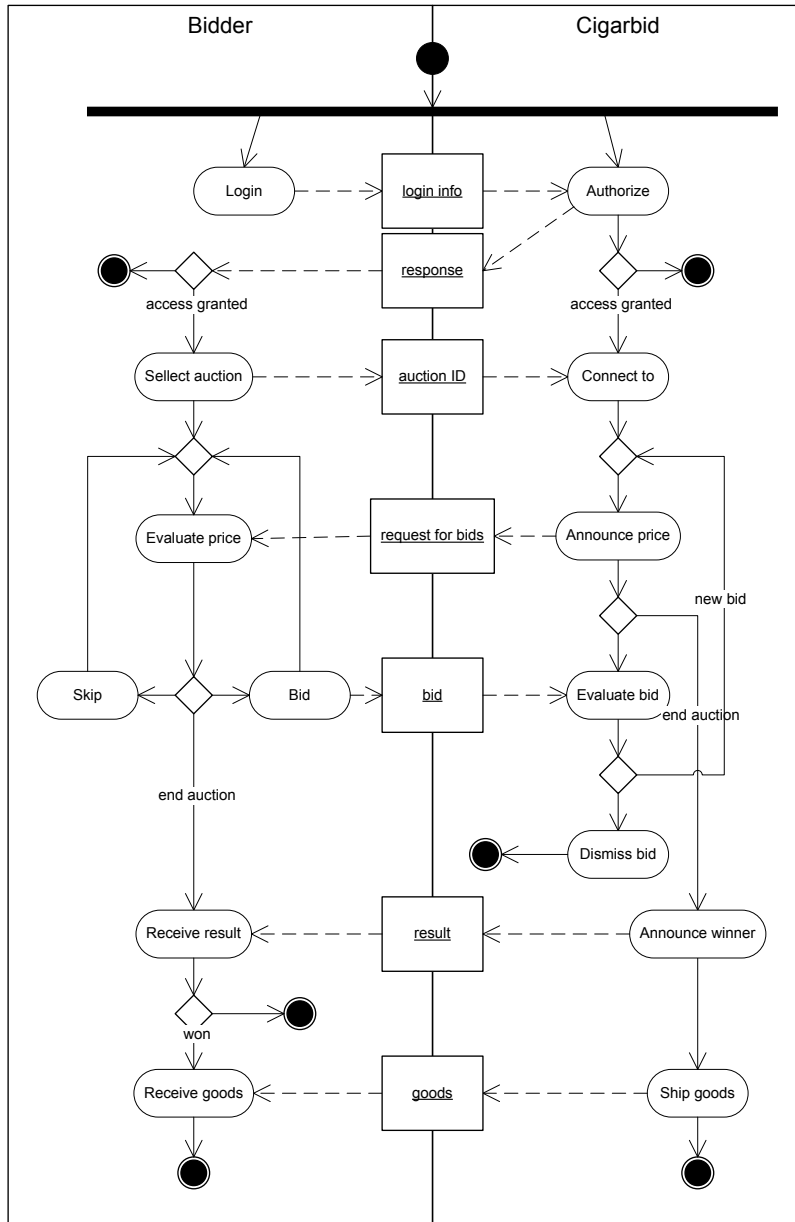
SOURCE: <http://www.cigarbid.com/auction/>, visited June 2004

DESCRIPTION: CigarBid.com is a cigar auction warehouse. Every day, it provides live auctions on a variety of products, including cigars, merchandise, 5-packs, samplers, humidors and more. All these products are available to bid on 24 hours per day in several types of auction. Auctions allow for proxy bidding and commenting on every bit in natural language messages.

process model: Registration



process model: **Auctioning**



4. eBay

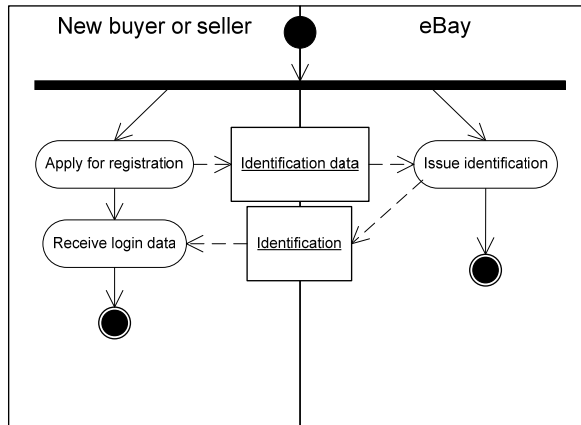
NAME: eBay

SOURCE: <http://www.ebay.com/>, visited June 2004

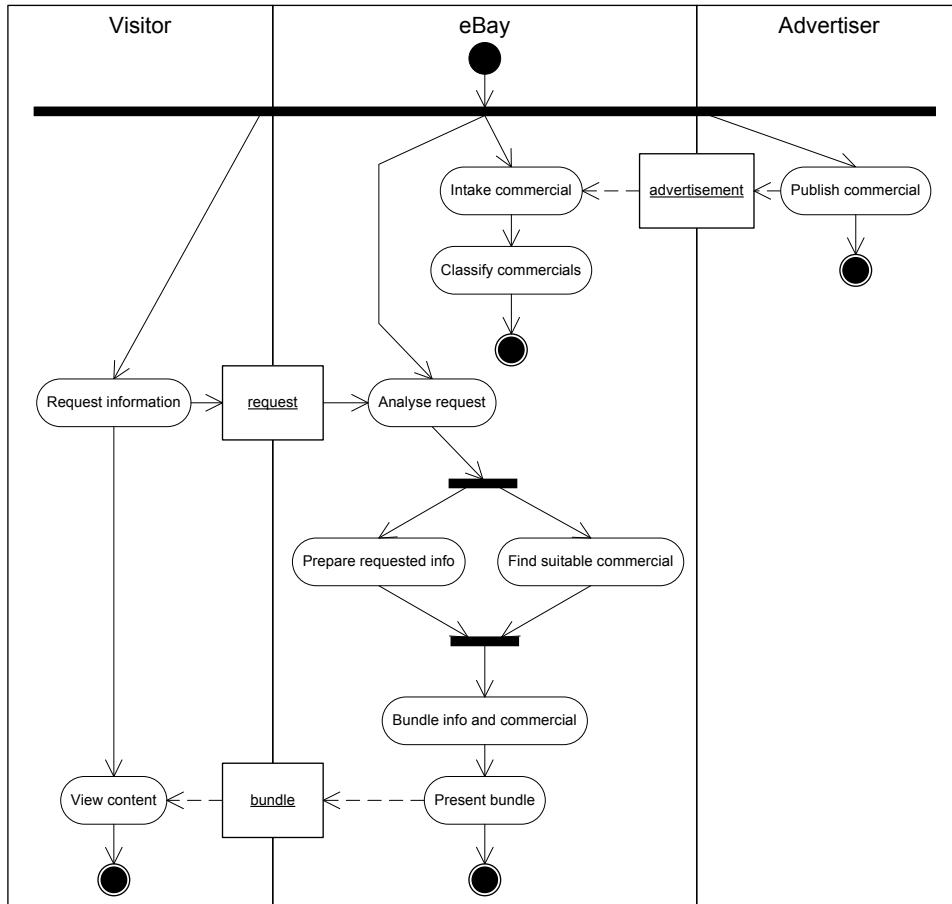
DESCRIPTION: eBay offers a wide variety of features and services that enable members to buy and sell on the site quickly and conveniently. Buyers have the option to purchase items in auction-style format or items can be purchased at fixed price through a feature called Buy-It-Now. In addition, items at fixed price are also available Half.com, an eBay company.

eBay has numerous services which enhance the trading experience, including marketplace services such as: online payments by PayPal; a wide array of Buyer and Seller tools; and the Developers Program for community members who would like to develop their own solutions. eBay's mission is to provide a global trading platform where practically anyone can trade practically anything.

process model: Registration



process model: Advertising



5. Electronic Courthouse, the

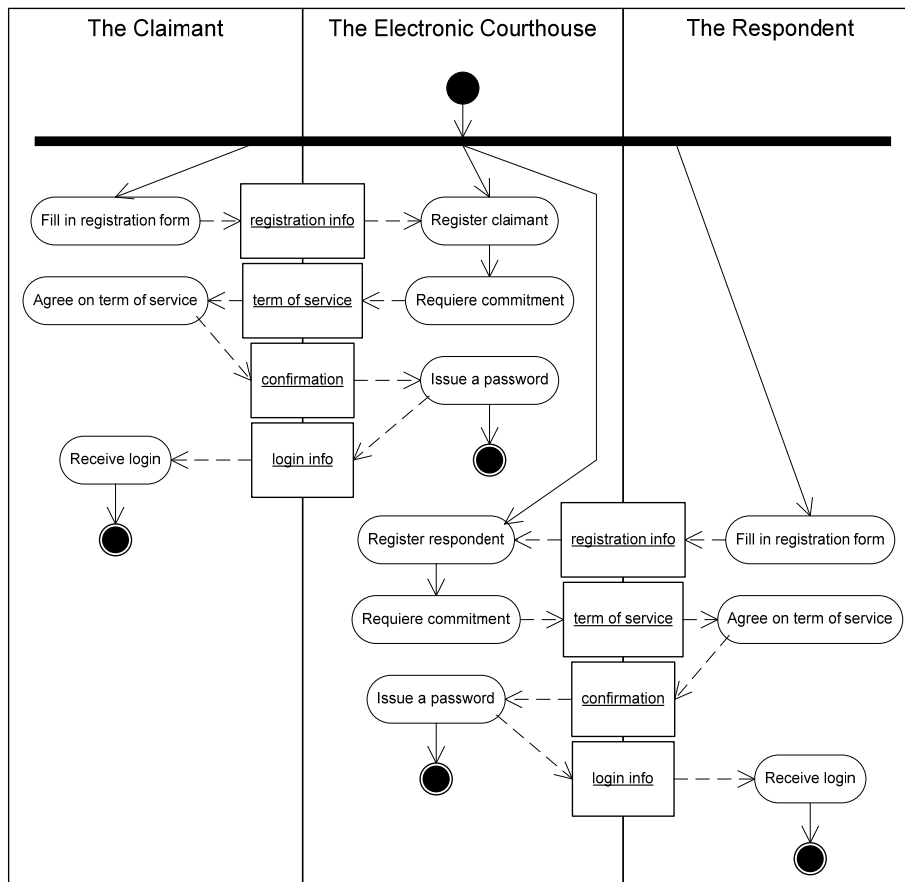
NAME: The Electronic Courthouse

SOURCE: <http://www.electroniccourthouse.com/>, visited June 2004

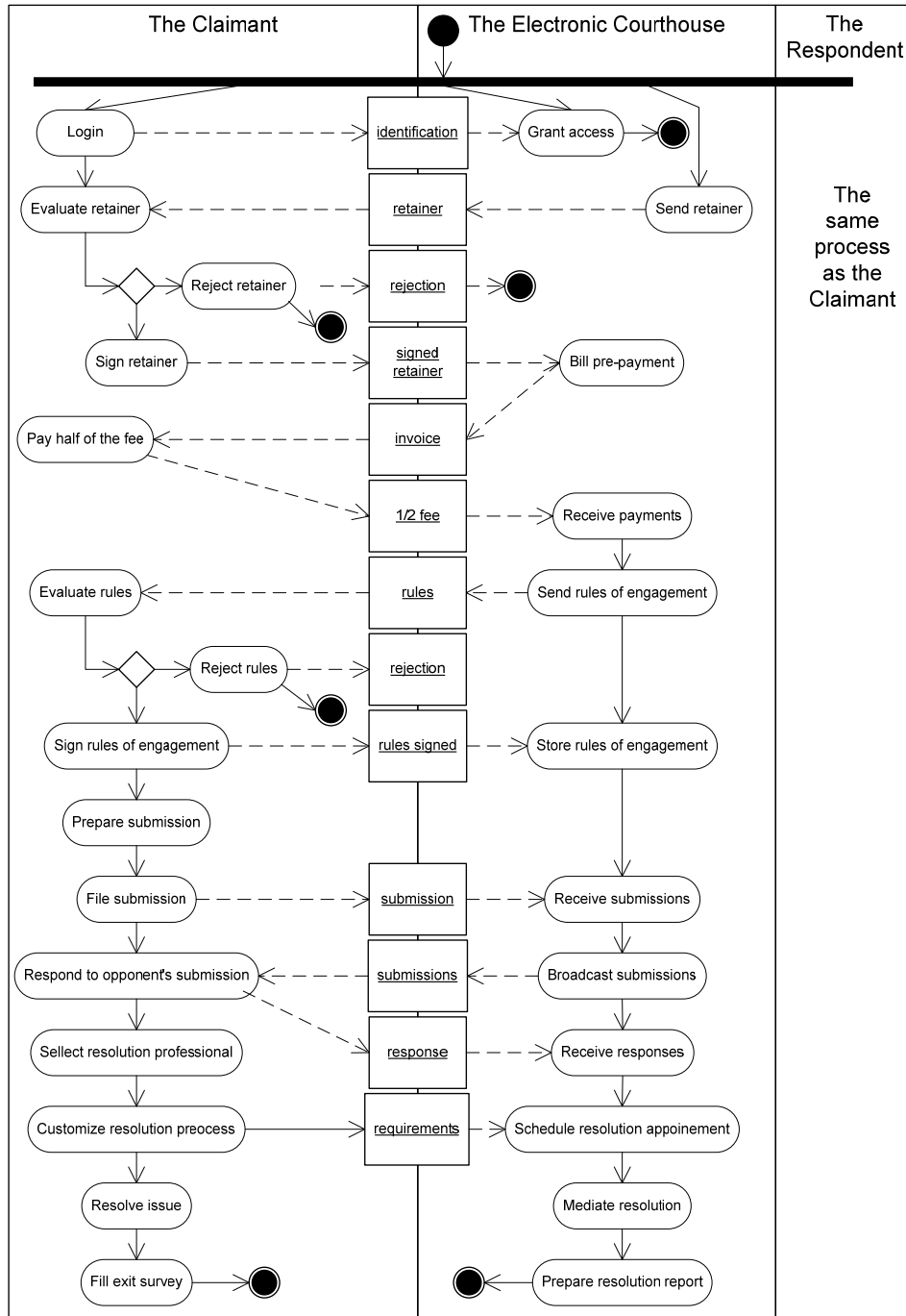
DESCRIPTION: The Electronic Courthouse provides an online forum for businesses as an alternative to expensive litigation. It allows parties to resolve their commercial disputes fast and remotely. It combines automated information systems and a team of resolution professionals, specializing in, but not limited to, real-estate, insurance, supply chain disputes, breach of contract, sports and entertainment law.

Parties may complete their submissions online, search online legal data base for answers to their legal questions, use translation services, and meet with the other party and their resolution professional in a secure Web-based meeting facility supported by voice conferencing. Parties can choose among several dispute resolution methodologies, including mediation, neutral evaluation, arbitration, or a process agreed to by the parties in a dispute resolution clause in their contract.

process model: Registration



process model: Resolution



6. MySimon

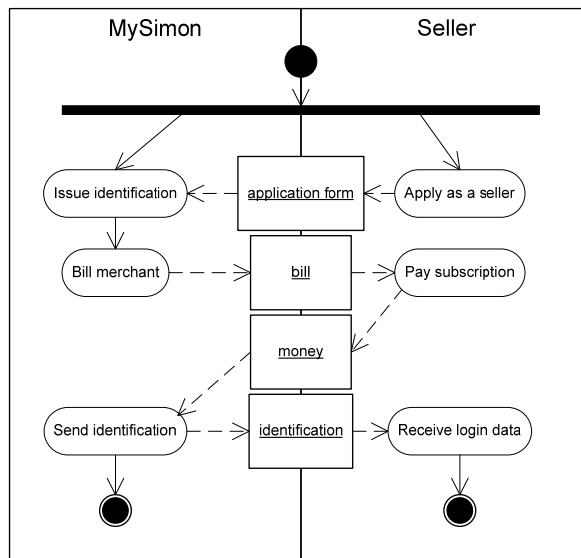
NAME: MySimon

SOURCE: <http://www.mysimon.com/>, visited June 2004

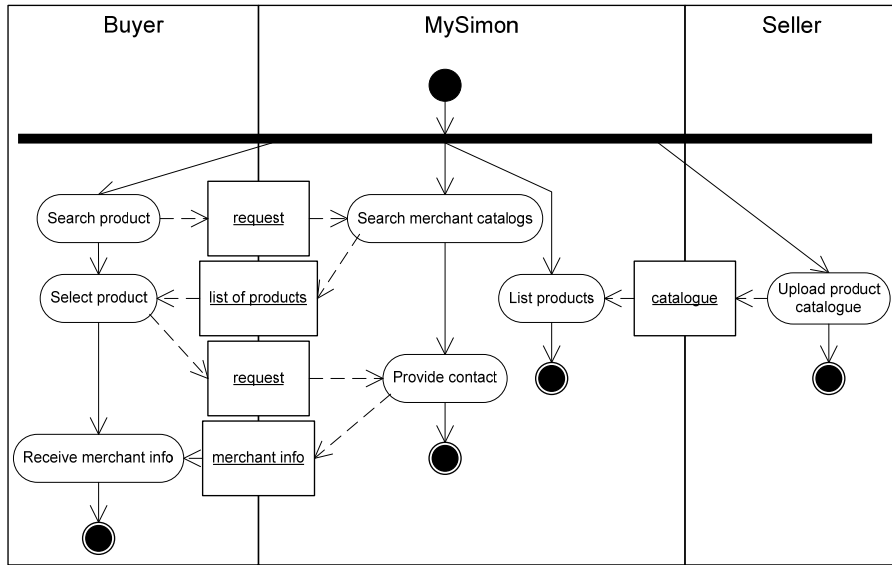
DESCRIPTION: mySimon is a comparison shopping service on the Internet for products and services. It searches thousands of merchants and lists millions of products so that its consumers can compare selections before making a purchase from one of its online merchants.

mySimon is a comparison shopping service. It does not sell or ship anything.

process model: Seller registration



process model: Comparison



7. Online Resolution

NAME: Online Resolution

SOURCE: <http://www.onlineresolution.com/>, visited June 2004

DESCRIPTION: Online Resolution brings together an extensive panel of professionals with expertise handling e-commerce disputes, insurance claims, business negotiations, and family conflict. Online Resolution provides online four methods for dealing with disputes: negotiation; mediation; expert evaluation; and arbitration.

Online Resolution takes the traditional claims adjusting process and combines it with the dynamic and consensual features of alternate dispute resolution to create a unique resolution option for insurance disputes. Online Resolution brings independent expert adjusters to online. Moreover, Online Resolution adjusters are independent neutrals: they do not represent the insurance company or any party to the dispute. They work with all the parties to uncover the facts and provide a well-reasoned opinion on resolving the case.

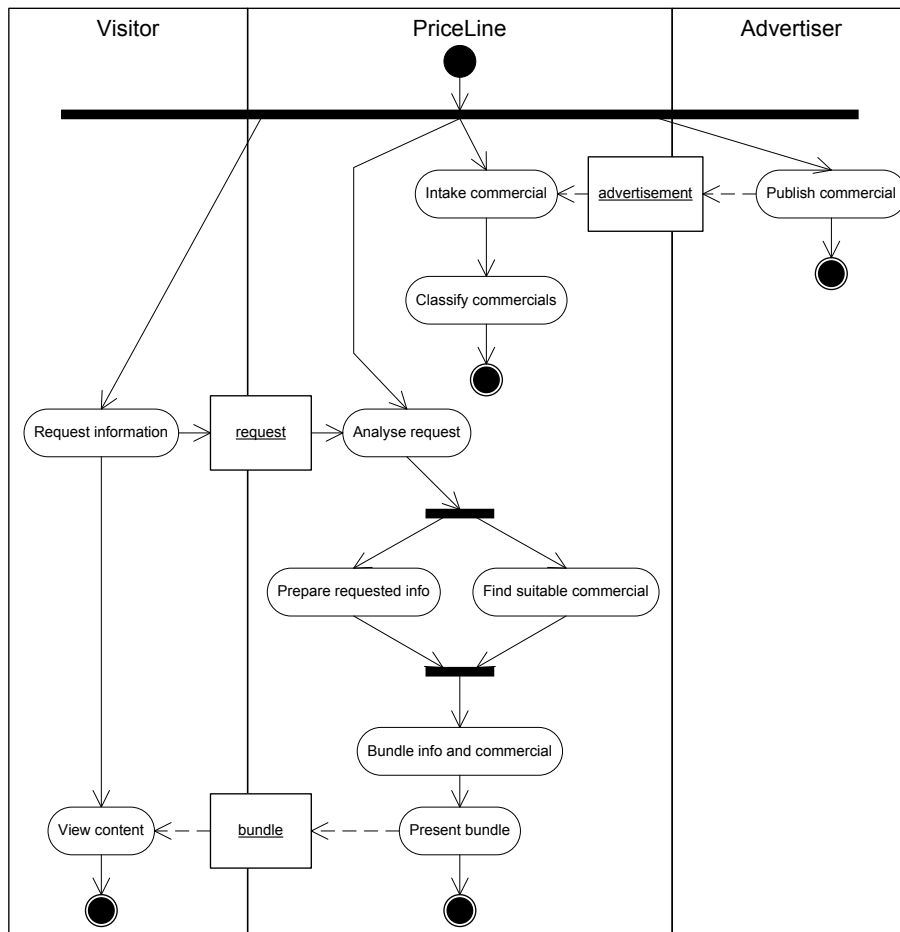
8. PriceLine

NAME: PriceLine

SOURCE: <http://www.priceline.com/>, visited June 2004

DESCRIPTION: Priceline.com offers a broad range of travel services including airline tickets, hotels, rental cars, vacation packages, cruises and more. Priceline.com is a buying service where you can save money by naming your own price. Priceline will take an offer for, for example, name-brand hotels and search to see if any will agree to the price asked. Since thousands of hotel rooms go unsold every night, companies would be willing to consider low prices: it makes sense for them.

process model: Advertising



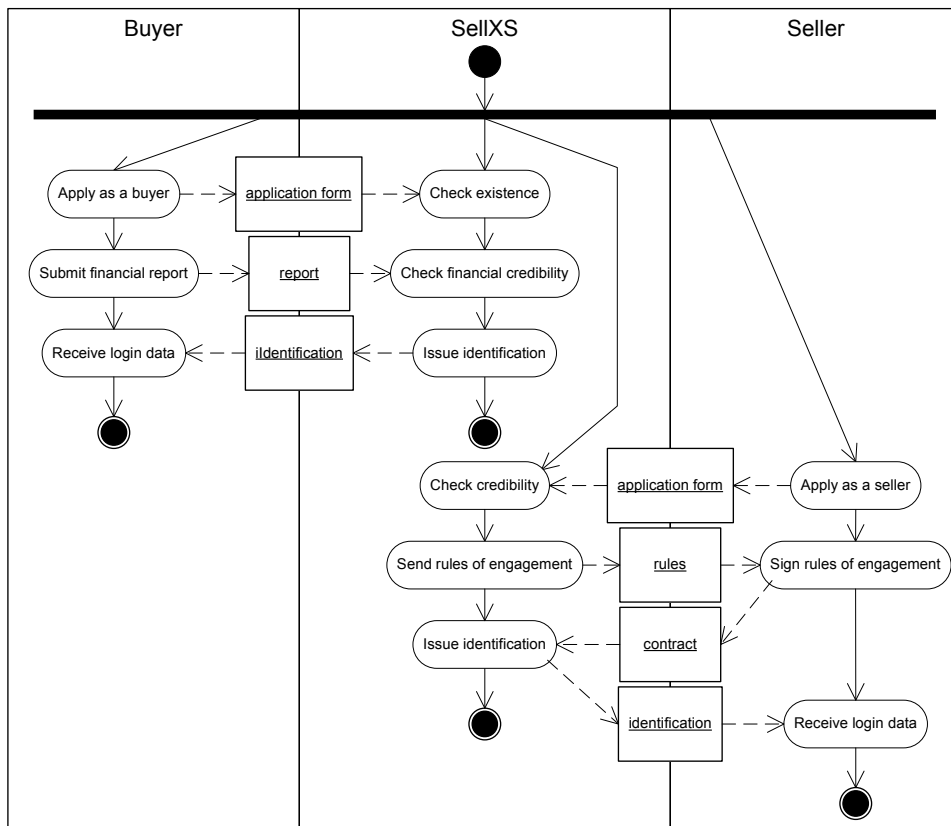
9. SellXS

NAME: SellXS

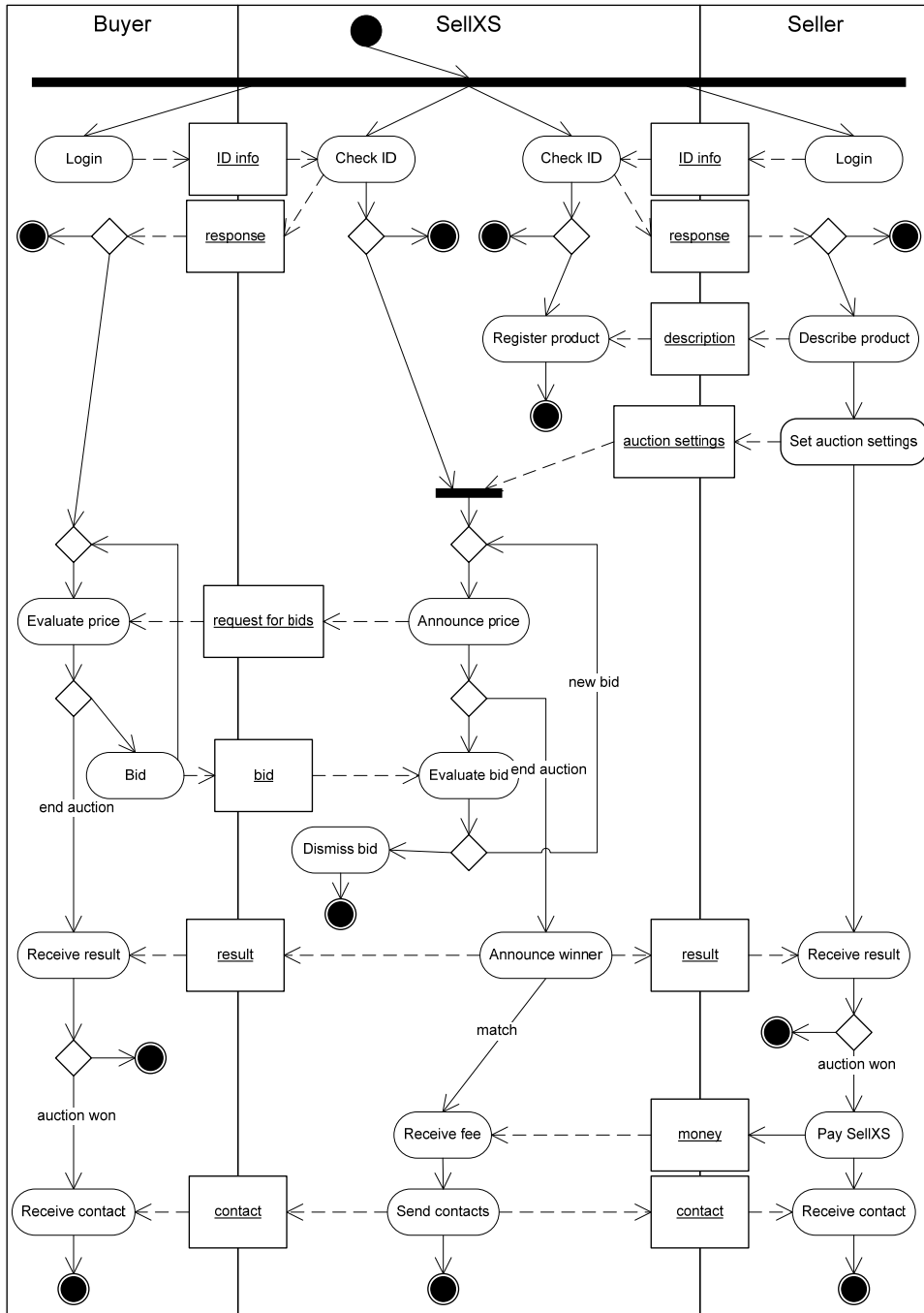
SOURCE: <http://www.sellxs.com/>, visited June 2004

DESCRIPTION: SellXS.com is a business-to-business (B2B) marketplace for auctioning excess inventory in the semiconductor industry. Liquidating this inventory the traditional way is a daunting challenge that requires extensive research and negotiating with multiple suppliers. SellXS.com enables high-tech companies to buy and sell excess inventory on the open market by leveraging dynamic pricing and a highly scalable eBay-style direct trade model to create greater market efficiencies and cost savings than is possible through traditional or online broker models. SellXS.com facilitates direct trade between buyers and sellers.

process model: Registration



process model: Trading



10. SquareTrade - Sony business model

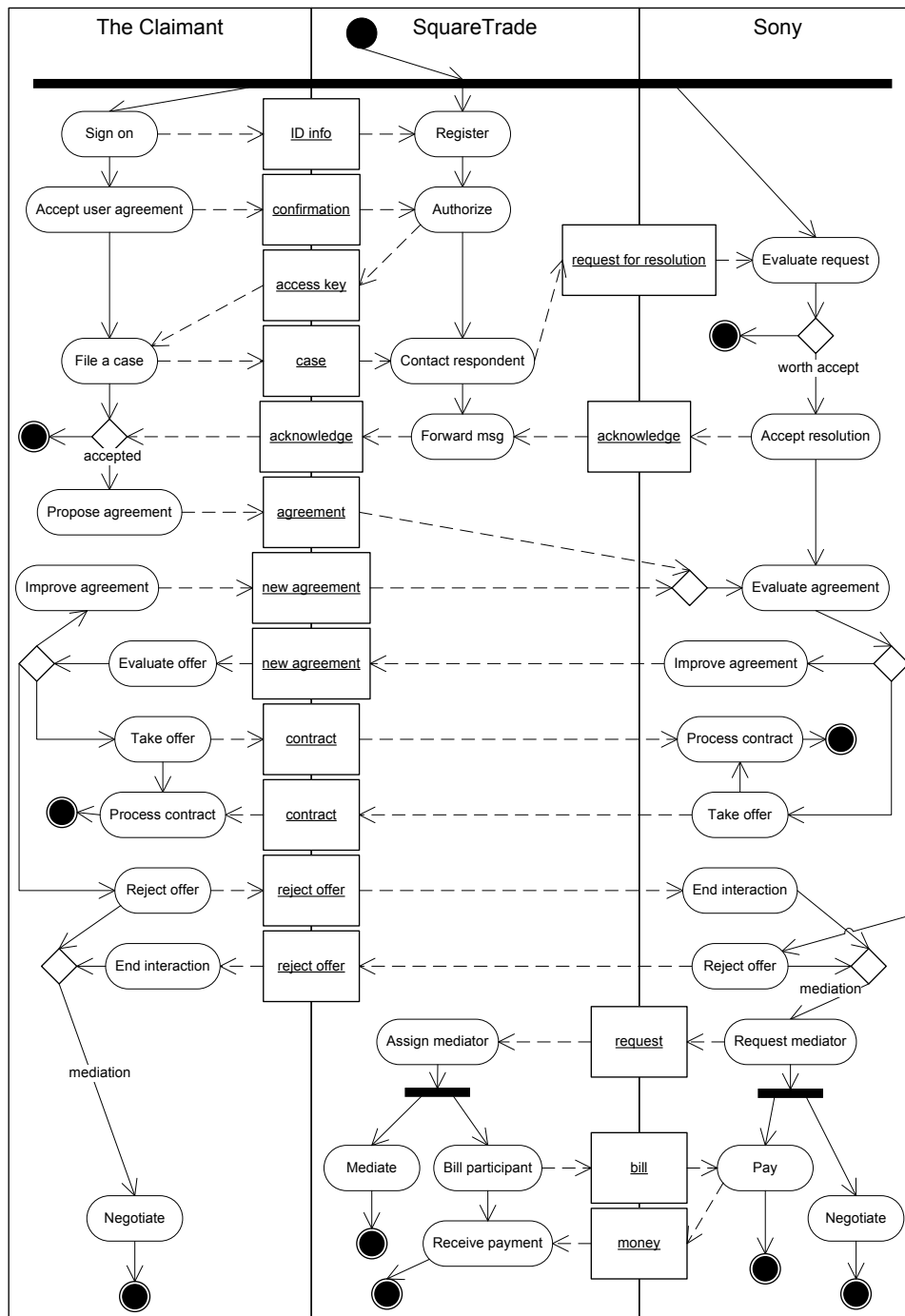
NAME: SquareTrade Sony

SOURCE: <https://www.squaretrade.com/>, visited June 2004

DESCRIPTION: SquareTrade provides Web-based dispute resolution services that give fast and convenient way for parties anywhere in the world to resolve issues that have arisen over online transactions. During the process of resolution, parties work together to resolve problems within the SquareTrade system, either independently using a direct negotiation service or through mediation. When a case is filed, the other party is notified and given the opportunity to respond. If the other party responds to the case, the case automatically goes into direct negotiation, and the filing party can request a mediator at any time.

In the customized business model for Sony, all services are free of charge for the client. Moreover, the negotiation and mediation services are not time-restricted. Sony has outsourced its business process of client's claims handling. Sony pays fee to SquareTrade to manage the conflicts with its clients.

process model: Resolution



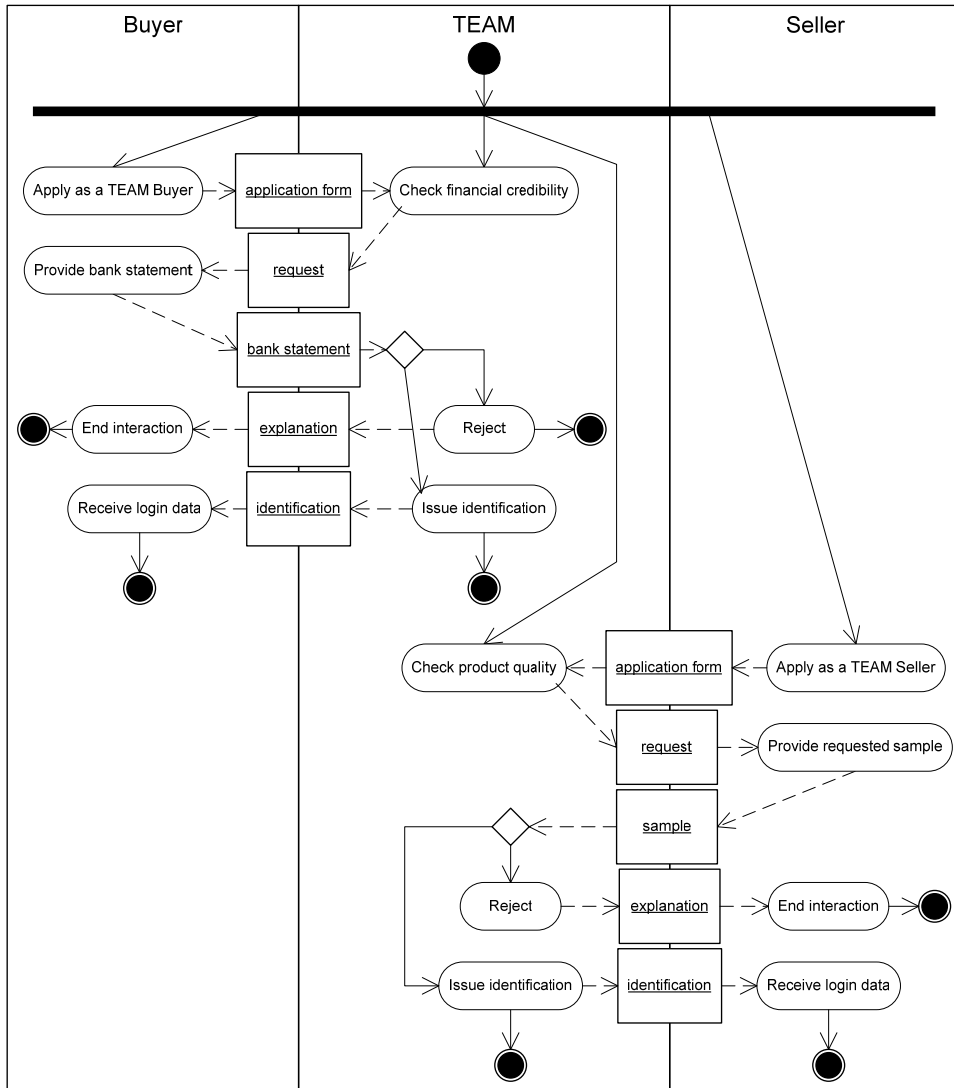
11. TEAM

NAME: TEAM

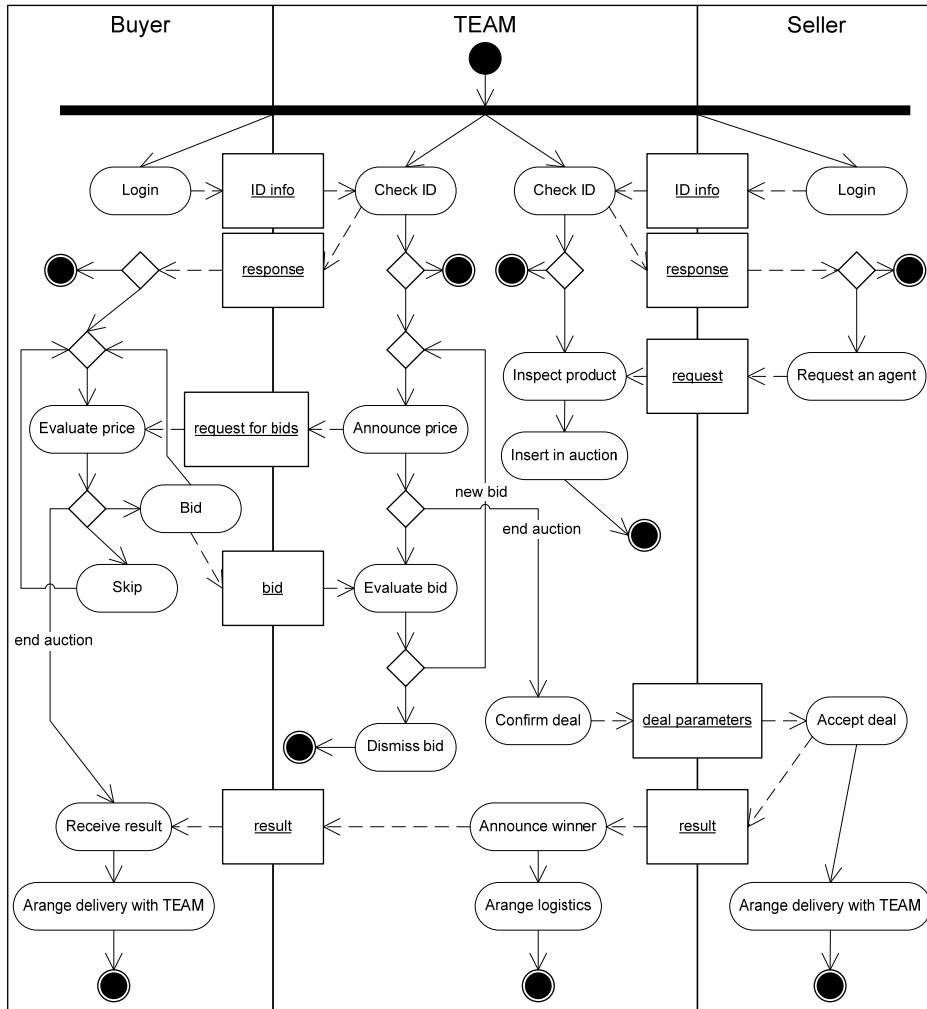
SOURCE: <http://www.teamauctionsales.com/>, visited June 2004

DESCRIPTION: The Electronic Auction Market (TEAM) is an online, interactive marketplace that brings cattle buyers and sellers together through the power of the Internet. TEAM is a real-time cattle auction with multiple sales weekly. Auctions are held both online, and in conjunction with live sales. TEAM provides buyers the opportunity to bid on quality strings of cattle they would normally not have access to. TEAM provides maximum exposure to cattle being sold, resulting in maximum dollars for the cattle.

process model: Registration



process model: **Auctioning**



12. TeleTrade

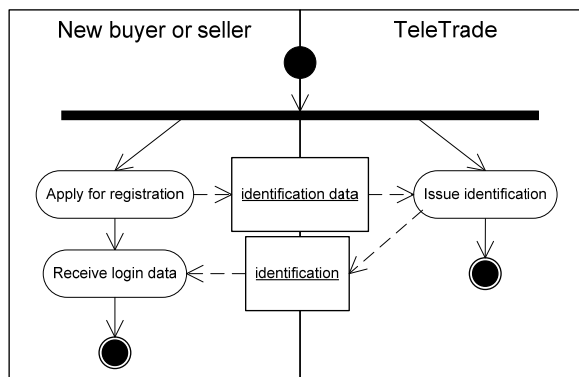
NAME: TeleTrade

SOURCE: <http://www.teletrade.com/>, visited June 2004

DESCRIPTION: Teletrade features Internet bidding and bidding over our toll-free 800 lines using a touch-tone phone from anywhere in North America. All participants are brought together into one auction. Teletrade auctions do not go on for days; they always end the day they start.

Teletrade possesses all lots offered in our auctions, describes each lot accurately, packages, insures and ships all winning bids. All services at Teletrade's web site are free. One only pays a fee when one buys or sells an item.

process model: Registration



Appendix G Library of Process Patterns

1. Advertising	353
2. Auction	355
3. Binding registration	357
4. Identification.....	359
5. Offer counter-offer.....	361
6. Outsourcing	363
7. Paid registration	365
8. Payment	367
9. Payment in terms	369
10. Registration.....	371
11. Restricted registration.....	373
12. Take it or leave it.....	375

1. Advertising

NAME: Advertising

CODE: ADV

HEADLINE: An intermediary offers to broadcast a message to many receivers.

CONTEXT: A business, i.e. an advertiser, has a new or unknown product. It is interested in advertising the product to a great number of potential buyers. An intermediary has access to many clients.

DESCRIPTION: An intermediary offers its product¹ bundled with a commercial from an advertising company. To consume the product, the client of the intermediary has to unbundled the product, which means to get exposed to the commercial.

PROBLEM:

roles: The pattern includes three roles played by three different businesses:

- *Advertiser* is the role played by the business seeking publicity;
- *Intermediary* is the role played by the business that provides a product bundled with an advertisement; and
- *Client* is the role played by the consumers of products provided by the *Intermediary*.

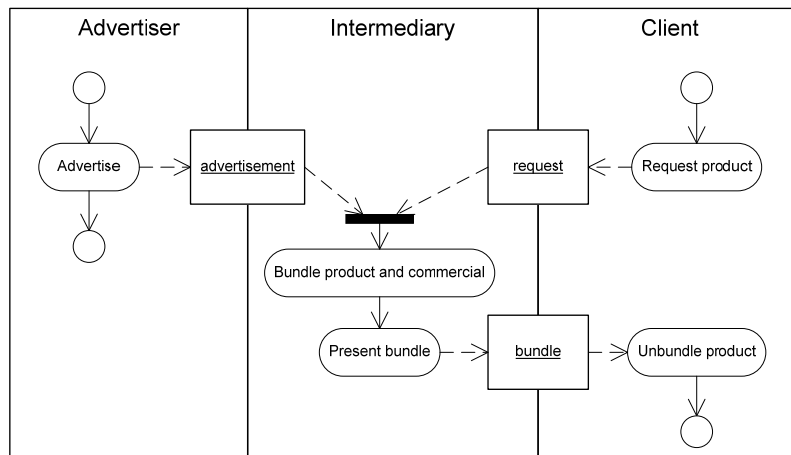
forces:

Role	Force	
Advertiser	F1	Reach potential clients
Intermediary	F2	Utilize access to buyers
Client	F3	Consume products

SOLUTION: An intermediary delivers its products bundled with commercials from advertisers.

¹ The term product includes also services.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Access potential clients	Advertiser	F1
C2 Bundle products and advertisements	Intermediary	F2
C3 Receive products	Client	F3

operationalization:

Capability code	Capability	Variable domain	Value
C1	Deliver message to potential clients		
a1	Informed potential clients	Boolean	true
C2	Deliver bundle of product and advertisement		
a2	Delivered bundled product and advertisement	Boolean	true
C3	Complete request for products		
a3	Completed request for products	Boolean	true

OCCURRED IN:

- eBay
- PriceLine

2. Auction

NAME: Auction

CODE: AUC

HEADLINE: Auction mechanism for price determination.

CONTEXT: A business environment in which there is great demand for a product of limited supply.

DESCRIPTION: A business performs price determination activity for a product of limited supply. During a time interval, it continuously accepts bids for the product. At the end of the interval, the best bid is announced as a winning bid.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

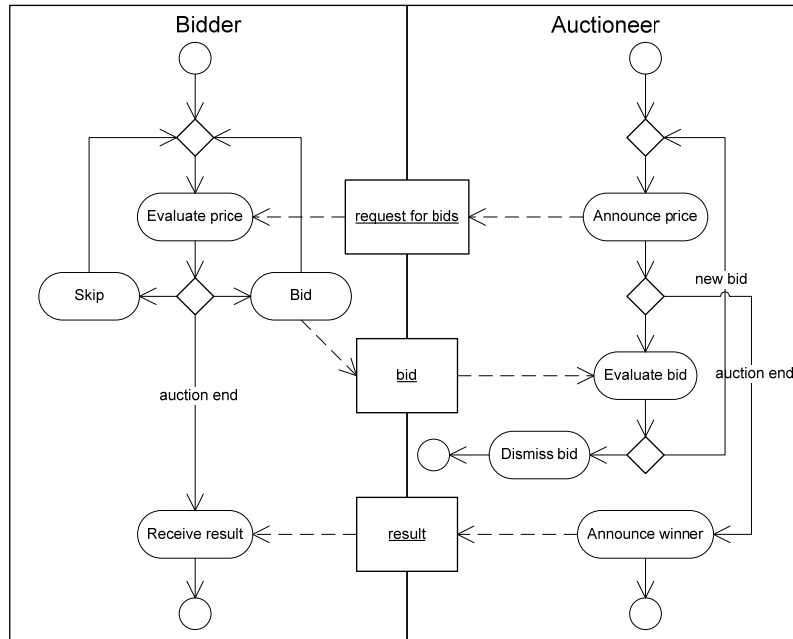
- *Bidder* is the role played by a business competing for the product;
- *Auctioneer* is the role played by a business that runs a auction to determine the best price for a product.

forces:

Role	Force	
Bidder	F1	Get a product at the lowers possible price
	F2	Get a product of limited supply
Auctioneer	F3	Determine the best price

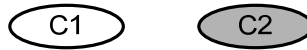
SOLUTION: Run an auction.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Get the optimal price	Bidder	F1, F2
C2 Determine the best price	Auctioneer	F3

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Get the optimal price		
a1	Optimal price	Boolean	true
C2	Determine the best price		
a2	Determined best price	Boolean	true

OCCURRED IN:

- Aalsmeer Flower Auction
- Cigarbid
- SellXS
- TEAM
- TeleTrade

3. Binding registration

NAME: Binding registration

CODE: BRG

HEADLINE: A business requires commitment from clients upon registration.

CONTEXT: A business environment in which businesses need strong commitment from their clients.

DESCRIPTION: A client applies for a registration during which it sends its identification information to a business. In addition, the business requires billing information to guarantee the commitment of the client. The business registers the client and subsequently issues a pass.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

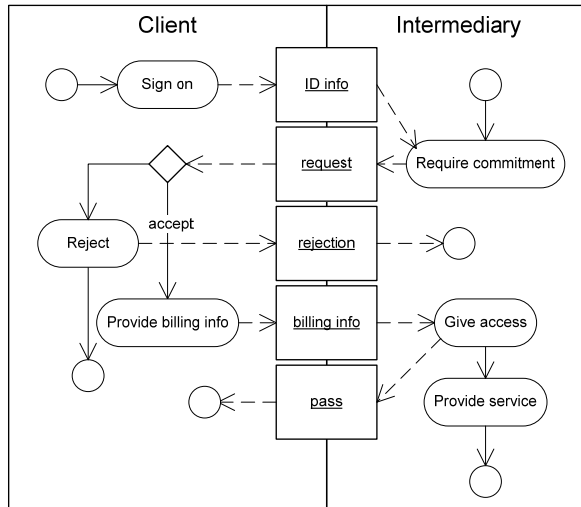
- *Intermediary* is the role played by the business that requires the commitment and issues the identification;
- *Client* is the role played by a business that provides the private and billing information, and receives the pass.

forces:

Role	Force	
Intermediary	F1	Discriminate clients
	F2	Guarantee client's commitment
	F3	Subsequent identification of clients
Client	F4	Subsequent access to services
	F5	Subsequent profile building

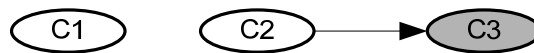
SOLUTION: Issue a pass only after commitment is made that guarantees participation of future transactions.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Obtain client's commitment	Intermediary	F2
C2 Register clients	Intermediary	F1, F3
C3 Obtain alias	Client	F4, F5

operationalization:

Capability code		Capability	
Variable code	Variable	Variable domain	Value
C1	Obtain client's commitment		
a1	Obtained commitment	Boolean	true
C2	Register clients		
a2	Registered clients	Boolean	true
C3	Obtain alias		
a3	Obtained alias	Boolean	true

OCCURRED IN:

- Cigarbid
- SellXS
- TeleTrade
- SquareTrade - Sony business model
- PriceLine
- The Electronic courthouse

4. Identification

NAME: Identification

CODE: IDN

HEADLINE: A business grants access only to known clients.

CONTEXT: An open environment with anonymous clients. The transaction handling requires participants to be known; the business process runs differently for different participants; or there are restricted areas where only privileged clients are allowed.

DESCRIPTION: The client performs a login activity during which it send its identification information to the intermediary. The intermediary examines the data and eventually grants access to the client.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

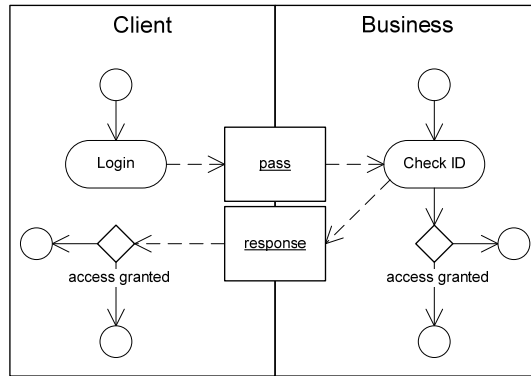
- *Intermediary* is the role played by the business performing the identification;
- *Client* is the role played by a business that identifies it-self in front of the *Intermediary*.

forces:

Role	Force	
Intermediary	F1	Customize behaviour according to clients
Client	F2	Access to services
	F3	Profile building

SOLUTION: Each client must identify it-self prior to engaging in any interaction with the intermediary.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Identify clients	Intermediary	F1
C2 Ensure identity	Client	F2, F3

operationalization:

Capability		Capability	
Capability code	Variable code	Variable	Value
C1	a1	Identified clients	true
C2	a2	Ensured identity	true

OCCURRED IN:

- APX
- Cigarbid
- SellXS
- TEAM
- TeleTrade
- The Electronic courthouse

5. Offer counter-offer

NAME: Offer counter-offer

CODE: OCO

HEADLINE: Negotiation process of exchange of offers and counter-offers.

CONTEXT: An agreement has to be reached on an issue with high stakes for both participants.

DESCRIPTION: A business (in our case a supplier) opens a negotiation process by sending an offer to its partner (in our case consumer.) The response is a new and improved offer, counter-offer. The iterative exchange of counter offers continues until one of the parties either accept or rejects the offer.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

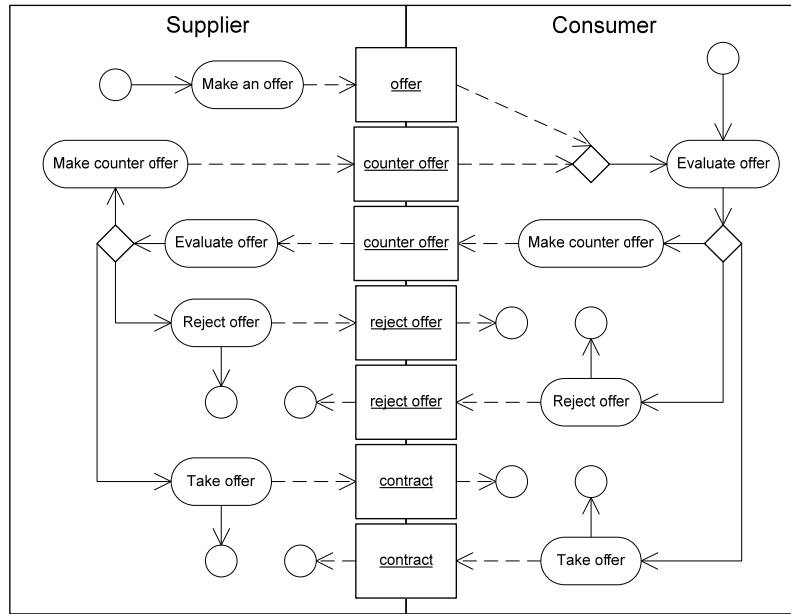
- *Intermediary* is the role played by the market creator;
- *Client* is the role played by businesses using the services of *Intermediary*. *Clients* have buying power.

forces:

Role	Force	
Supplier	F1	Achieve optimal agreement
Consumer	F2	Have equal influence on the contract

SOLUTION: Businesses engage in an iterative exchange of offers until one party accepts or reject the last submitted offer.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Iterative exchange of offers	Supplier	F1
C2 Ability to change the contract	Consumer	F2

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Iterative exchange of offers		
a1	Negotiable agreement	Boolean	true
C2	Ability to change the contract		
a2	Flexible contract	Boolean	true

OCCURRED IN:

- APX
- SquareTrade - Sony business model

6. Outsourcing

NAME: Outsourcing

CODE: OUT

HEADLINE: An intermediary offers its core service to be executed on behalf of another business.

CONTEXT: An intermediary offers a specialized service which requires an interaction with the clients of the outsourcing business.

DESCRIPTION: An intermediary provides the delivery of a product¹ on behalf of another business. The intermediary makes a fixed offer that the business has to accept or reject. In case of accepting the offer, the business is billed and the product is provided to a client of the business.

PROBLEM:

roles: The pattern includes three roles played by three different businesses:

- *Intermediary* is the role played by the business that insources the provision of a product;
- *Business* is the role played by the business that wants to outsource the provision of a product.
- *Client* is the role played by a client of *Business* which needs the product delivered by *Intermediary*.

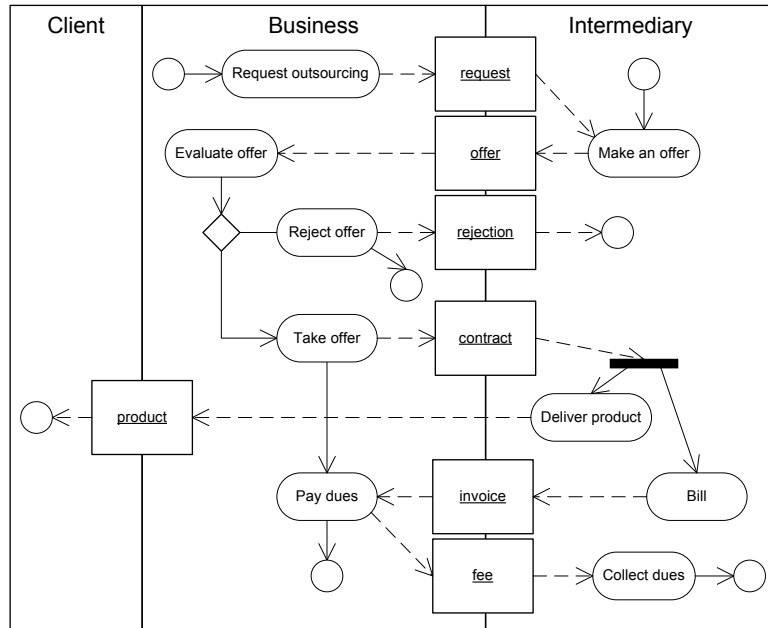
forces:

Role	Force	
Intermediary	F1	Quick contracting of its products including the payment of the products
Business	F2	Outsource non-core activities
Client	F3	Experience product

SOLUTION: An intermediary accepts requests for certain products from a business. The offers it makes are not negotiable. (The business has to take or reject the offer.) Then, the intermediary in parallel bills the business and delivers the product to the client of the business.

¹ The term product includes also services.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Contract and bill insourcing of products	Intermediary	F1
C2 Contract outsourcing of products	Business	F2
C3 Receive products	Client	F3

operationalization:

Capability code	Capability	Variable code	Variable	Variable domain	Value
C1	Contract and bill insourcing of products				
a1	Contracted and billed insourcing			Boolean	true
C2	Contract outsourcing of products				
a2	Contracted outsourcing of a product			Boolean	true
C3	Receive products				
a3	Received products			Boolean	true

OCCURRED IN:

- TeleTrade

7. Paid registration

NAME: Paid registration

CODE: PRG

HEADLINE: An intermediary admits clients in return of registration fee.

CONTEXT: An open environment in which businesses can free-ride some of the services of another business.

DESCRIPTION: A client applies for registration during which it sends its identification information to the intermediary. To prevent free-riders, the intermediary requires that clients pay a fee. The intermediary registers clients and subsequently issues identification passes.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

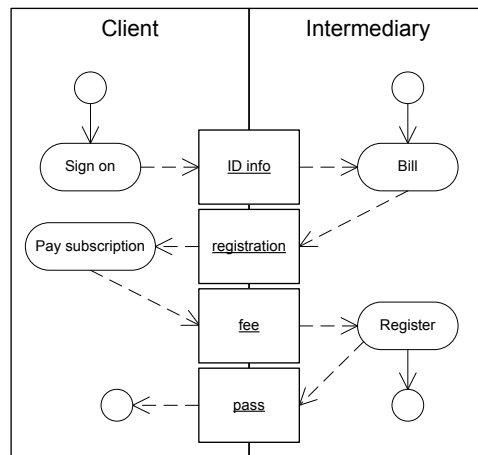
- *Intermediary* is the role played by the business that collects the fees and issues the IDs;
- *Client* is the role played by a business that provides the private information and receives the ID.

forces:

Role	Force	
Intermediary	F1	Discourage free-riders
	F2	Discriminate clients
	F3	Collect money for offered service
	F4	Subsequent identification of clients
Client	F5	Subsequent access to services
	F6	Subsequent profile building

SOLUTION: Collect fees for every registration.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Register clients	Intermediary	F2, F4
C2 Prepaid use of services	Intermediary	F3
C3 Discourage free-riders	Intermediary	F1
C4 Obtain alias	Client	F5, F6

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Register clients		
a1	Registered clients	Boolean	true
C2	Prepay use of services		
a2	Prepaid use of services	Boolean	true
C3	Discourage free-riders		
a3	Discouraged free-riders	Boolean	true
C4	Obtain alias		
a4	Obtained alias	Boolean	true

OCCURRED IN:

- APX
- MySimon

8. Payment

NAME: Payment

CODE: PAY

HEADLINE: A provider bills a consumer.

CONTEXT: Before or after a delivery of a product, payment has to occur.

DESCRIPTION: A provider sends an invoice to a consumer to which it delivered or is going to deliver a product¹. The provider receives money as a response to the invoice.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

- *Provider* is the role played by a business that delivers certain product;
- *Consumer* is the role played by a business experiencing the product and paying the money.

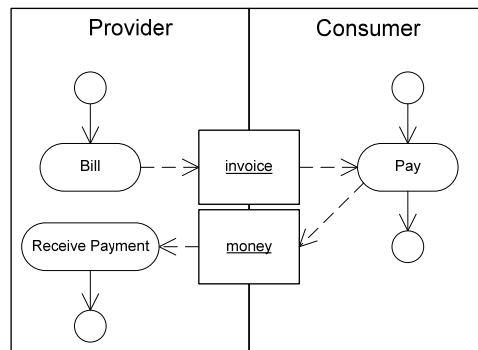
forces:

Role	Force	
Provider	F1	Receive due money for product
Consumer	F2	Fulfil contractual obligations

SOLUTION: The provider of a product sends an invoice to the consumer to which the consumer has to respond with money transfer.

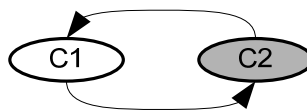
¹ The term product includes also services.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Receive payment	Provider	F1
C2 Pay dues	Consumer	F2

operationalization:

Capability code		Capability	
Variable code	Variable	Variable domain	Value
C1	Receive payment		
a1	Received payment	Boolean	true
C2	Pay dues		
a2	Paid dues	Boolean	true

OCCURRED IN:

- Aalsmeer Flower Auction
- SellXS
- TeleTrade
- OnlineResolution
- SquareTrade - Sony business model
- The Electronic courthouse

9. Payment in terms

NAME: Payment in terms

CODE: PIT

HEADLINE: A provider bills a consumer in terms.

CONTEXT: Consumer pays in terms in parallel with the delivery of the product.

DESCRIPTION: A provider sends invoices to a consumer to which it delivers a product. The invoices are sent and the payments are received in predetermined time intervals.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

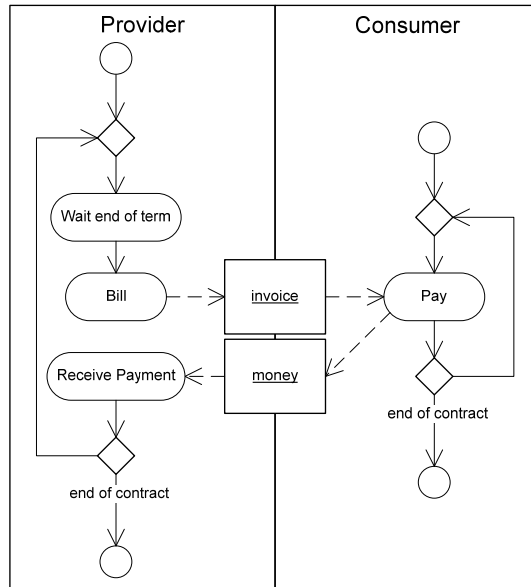
- *Provider* is the role played by a business that delivers certain product;
- *Consumer* is the role played by a business experiencing the product and paying the money.

forces:

Role	Force	
Provider	F1	Receive due money for delivered service
	F2	Balance money flow are product delivery
Consumer	F3	Fulfil contractual obligations

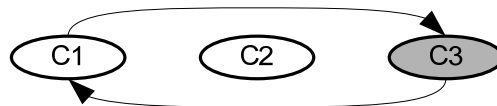
SOLUTION: The provider of a product sends invoices to the consumer in predetermined terms, to which the consumer has to respond with money transfers.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Receive payment	Provider	F1
C2 Keep steady money flow	Provider	F2
C3 Pay dues	Consumer	F3

operationalization:

Capability code	Capability	Variable code	Variable	Variable domain	Value
C1	Receive payment	a1	Received payment	Boolean	true
C2	Keep steady money flow	a2	Payment in terms	Boolean	true
C3	Pay dues	a3	Paid dues	Boolean	true

OCCURRED IN: models not listed in Appendix F Process Models of Real-life Businesses.

10. Registration

NAME: Registration

CODE: REG

HEADLINE: A business identifies its clients.

CONTEXT: An open environment in which the handling of transactions requires knowledge about the ID of the participants.

DESCRIPTION: A client applies for registration during which it sends its identification information to the intermediary. The intermediary registers the client and subsequently issues a pass that uniquely identifies the client.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

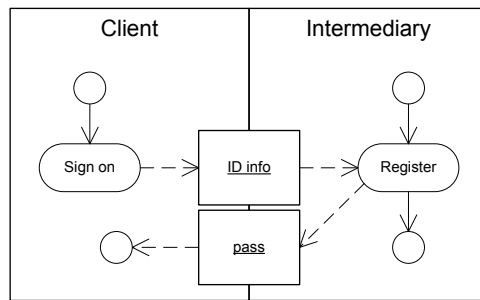
- *Intermediary* is the role played by the business that issues the identification;
- *Client* is the role played by a business that provides the identification information and receives the pass.

forces:

Role	Force	
Intermediary	F1	Discriminate clients
	F2	Subsequent identification of clients
Client	F3	Subsequent access to services
	F4	Subsequent profile building

SOLUTION: Issue a unique pair of username and access key to every new client that has not registered yet.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Register clients	Intermediary	F1, F2
C2 Obtain alias	Client	F3, F4

operationalization:

Capability code	Capability		
Variable code	Variable	Variable domain	Value
C1	Register clients		
a1	Registered client	Percentage	85
C2	Obtain alias		
a2	Obtained alias	Percentage	85

OCCURRED IN:

- eBay
- TeleTrade

11. Restricted registration

NAME: Restricted registration

CODE: RRG

HEADLINE: An intermediary screens clients.

CONTEXT: A business environment in which participants are required to fulfil certain credibility requirements. The non-credible businesses must not be given access.

DESCRIPTION: A client applies for registration during which it sends its identification information to the intermediary. The intermediary request evidence that the client is worth registering. After validating the credibility of client, the Intermediary registers the client and subsequently issues a pass that uniquely identifies the client. If issuing a pass is refused then explanation is provided.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

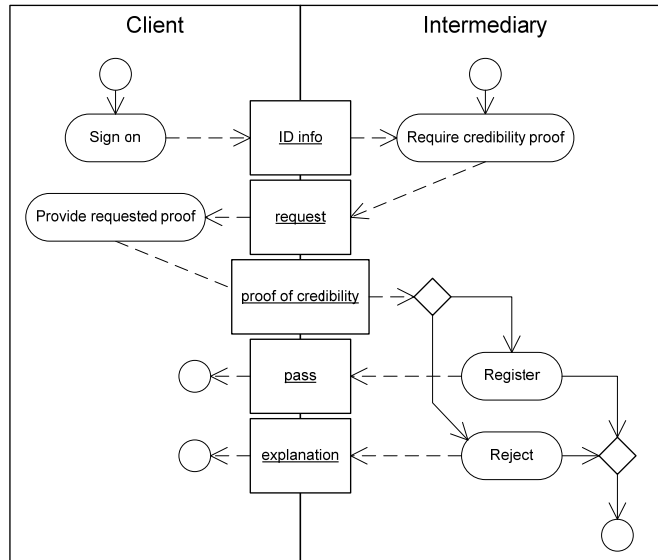
- *Intermediary* is the role played by the business that checks the credibility and issues the identification;
- *Client* is the role played by a business that provides the identification information, proof of credibility, and receives the pass.

forces:

Role	Force	
Intermediary	F1	Restrict access to credible clients
	F2	Discriminate clients
	F3	Subsequent identification of clients
Client	F4	Subsequent access to services
	F5	Subsequent profile building

SOLUTION: Issue a unique pair of username and access key to every new client that has not registered yet.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Screen clients	Intermediary	F1
C2 Register clients	Intermediary	F2, F3
C3 Obtain alias	Client	F4, F5

operationalization:

Capability code		Capability	
Variable code	Variable	Variable domain	Value
C1	Screen clients		
a1	Screened clients	Boolean	true
C2	Register clients		
a2	Registered clients	Boolean	true
C3	Obtain alias		
a3	Obtained alias	Boolean	true

OCCURRED IN:

- SelIXS
- TEAM

12. Take it or leave it

NAME: Take it or leave it

CODE: TOL

HEADLINE: A supplier makes a non-negotiable offer to a consumer.

CONTEXT: A business environment in which suppliers have market power and can dictate the rules of encounter.

DESCRIPTION: A supplier of a certain product makes an offer to a consumer. The offer contains the conditions on delivery of the product. The consumer has to accept or reject the offer.

PROBLEM:

roles: The pattern includes two roles played by two different businesses:

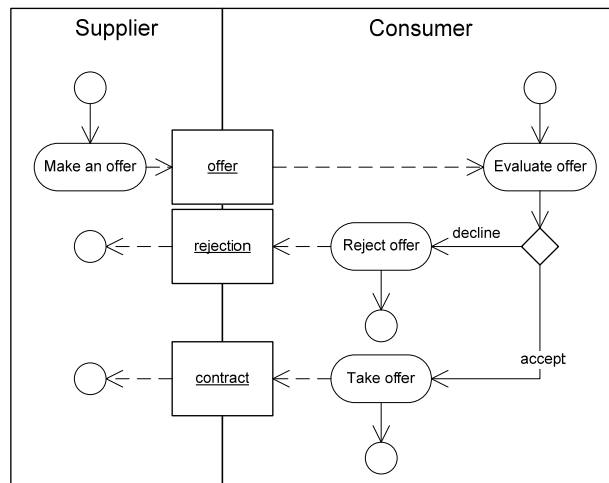
- *Supplier* is the role played by a business with market power which is making the offer;
- *Consumer* is the role played by a business seeking delivery of *Supplier*'s product.

forces:

Role	Force	
Supplier	F1	Determine contract conditions
Consumer	F2	Efficient contracting

SOLUTION: A supplier sends an offer that a consumer can only accept or reject.

process model:



CAPABILITIES:

capability model:



Capability	Benefiting role	Resolving force
C1 Dictate contract conditions	Supplier	F1
C2 Efficient contracting	Consumer	F2

operationalization:

Capability code	Capability	Variable code	Variable	Variable domain	Value
C1	Dictate contract conditions	a1	Fixed contract	Boolean	true
C2	Efficient contracting	a2	Proposals made	Natural numbers	1

OCCURRED IN:

- The Electronic courthouse

References

1. Aalst van der, W.M.P. and Weske, M.: The P2P approach to Interorganizational Workflows. Proceedings of 13th International Conference on Advanced Information Systems Engineering (CAISE). Interlaken, Switzerland (2001)
2. Aalst van der, W.M.P., Hofstede ter, A.H.M., Kiepuszewski, B. and Barros, A.P.: Workflow Patterns. *Distributed and Parallel Databases*, 14(3) (2003) 5—51
3. Aalst van der, W.M.P.: Patterns. Available at: <http://is.tm.tue.nl/research/patterns/patterns.htm> (March 2007)
4. Adams, J., Koushik, S., Vasudeva, G. and Galambos, G.: *Patterns for e-business: A Strategy for Reuse*. IBM Press (2001)
5. Afuah, A. and Tucci, C.L.: *Internet Business Models and Strategies: Text and Cases*. McGraw-Hill Boston (2001)
6. Alexander, C., Ishikawa, S., and Silverstein M.: *A Pattern Language: Towns, Buildings, Construction*. The Oxford University Press New York (1977)
7. Amit, R. and Zott, C.: Value creation in e-business. *Strategic Management Journal*, 22 (2001) 493—520
8. Bower, J. L. and Christensen, C. M.: Disruptive technologies: catching the wave. *Harvard Business Review*, 73, 1 (January-February 1995) 43—53
9. Bowman, H., Steen, M.W.A., Boiten, E.A. and Derrick, J.: A formal framework for viewpoint consistency. *Formal Methods in System Design*, 21 (September 2002) 111—166
10. Braatz, B., Klein, M. and Schröter, G.: Semantical Integration of Object-Oriented Viewpoint Specification Techniques. In: *Integration of Software Specification Techniques for Applications in Engineering*. Lecture Notes in Computer Science, Springer (2004)
11. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A.: TROPOS: An Agent-Oriented Software Development Methodology. In *Journal of Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers (May 2004)
12. Bruijn, J. d. and Polleres, A.: Towards an Ontology Mapping Specification Language for the Semantic Web. DERI - Digital Enterprise Research Institute, DERI TR 2004-06-30 (2004)
13. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M.: *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons (1996)
14. Cooper, D. R. and Schindler, P. S.: *Business Research Methods*. Irwin/McGraw-Hill, Singapore (1998)
15. Dijkman, R.M., Quartel, D.A.C., Pires, L.F. and Sinderen van M.J.: An Approach to Relate Viewpoints and Modeling Languages. In: *Proceedings of the 7th IEEE Enterprise Distributed Object Computing (EDOC) Conference*, Brisbane, Australia (2003) 14—27
16. Doan A. and Halevy, A. Y.: Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, Special Issue on Semantic Integration (2005)
17. Easterbrook, S. and Nuseibeh, B.: Using ViewPoints for inconsistency management. *Software Engineering Journal*, 11(2) (1996) 132—132
18. Easterbrook, S., Finkelstein, A., Kramer, J. and Nuseibeh, B.: Coordinating Distributed ViewPoints: The Anatomy of a Consistency Check. *International Journal on Concurrent Engineering: Research & Applications*, 2(3) (1994)
19. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal on Software Engineering and Knowledge Engineering*, Special issue on Trends and Research Directions in Software Engineering Environments, 2(1) (1992) 31—58
20. Fowler, M.: *Analysis Patterns: Reusable Object Models*. Addison-Wesley Professional (1996)
21. Fu, X., Bultan, T. and Su, J.: Realizability of Conversation Protocols with Message Contents. *Proceedings IEEE International Conference on Web Services (ICWS)* (2004) 96—103

22. Gamma, E., Helm, R. Johnson R. and Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
23. Genomics and Disease Prevention: Genetic Testing Glossary. Available at : <http://www.cdc.gov/genomics/gtesting/ACCE/FBR/CF/CFGlossary2.htm> (November 2005)
24. Giorgini, P., Mylopoulos, J., Nicchiarelli, E. and Sebastiani, R.: Formal Reasoning Techniques for Goal Models. In Journal of Data Semantics (2004)
25. Gordijn, J. and Akkermans, J.M.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Engineering Journal, Springer Verlag, 8(2) (2003) 114—134
26. Gordijn, J., Akkermans, J.M. and Vliet van, J.C.: Business Modelling is not Process Modelling. In: Conceptual Modeling for E-Business and the Web, LNCS 1921. Salt Lake City, USA (October 2000) 40—51
27. Gordijn, J.: Value-based Requirements Engineering: Exploring Innovative e- Commerce Ideas. PhD thesis, Vrije Universiteit, Amsterdam (2002)
28. Gross, D. and Yu, E.: From Non-Functional Requirements to Design through Patterns. Sixth International Workshop on Requirements Engineering: Foundation for Software Quality, Stockholm (2000)
29. Hammer, M. and Stanton, S.: How Process Enterprises Really Work. Harvard Business Review (Nov-Dec 1999) 108—118
30. Hedman, J. and Kalling, T.: The Business Model: A Means to Understand the Business Context of Information and Communication Technology. Institute of Economic Research, Working Paper Series, 9 (2001), available at: <http://www.lri.lu.se/pdf/wp/2001-9.pdf>
31. IBM: IBM Patterns for e-business. Available at: <http://www-106.ibm.com/developerworks/patterns/> (December 2004)
32. ISO/IEC 10746-1 | ITU-T Recommendation X.901.: Open Distributed Processing - Reference Model. OMG (1995-98)
33. Kalfoglou, Y. and Schorlemmer, M.: Ontology Mapping: The State of the Art. In: Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S. and Uschold, M. (eds.): 04391 - Semantic Interoperability and Integration. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany (2005)
34. Kaplan, A.: The conduct of inquiry: Methodology for behaviour science. Transaction publishers, New Brunswick (USA) and London (UK) (1998)
35. Kindler, E., Martens, A. and Reising, W.: Inter-operability of Workflow Applications: Local Criteria for Global Soundness. Business Process Management, Models, Techniques, and Empirical Studies (2000) 235—253
36. Lamsweerde van, A., Dardenne, A., Delcourt, B. and Dubisy, F.: The KAOS Project: Knowledge Acquisition in Automated Specification of Software. Proceedings AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence (March 1991) 59—62
37. Lamsweerde van, A.: From System Goals to Software Architecture. In: Bernardo, M. and Inverardi, P. (eds.): Formal Methods for Software Architectures. LNCS 2804. Springer-Verlag (2003) 25—43
38. Leite, J.C.S.P. and Freeman, P.A.: Requirements validation through viewpoint resolution. IEEE transactions on Software Engineering, 17 (12) (1991) 1253—1269
39. Letier, E. and Lamsweerde van, A.: Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. Proceedings FSE 2004 - 12th International Symposium on the Foundation of Software Engineering. ACM Press, Newport Beach, CA (November 2004) 53—62
40. MEMO research projects: MEMO: Multi Perspective Enterprise Modelling. Available at: <http://www.uni-koblenz.de/~iwi/EM/MEMO/index.html> (August 2005)
41. Merriam-Webster: Merriam-Webster online Dictionary. Available at: <http://www.m-w.com/home.htm> (August 2005)

42. Mollaghasemi, M. and Pet-Edwards, J.: Technical briefing: making multiple-objective decisions. IEEE Computer Society Press, Los Alamitos, CA (1997)
43. Mylopoulos, J., Chung L., and Nixon, B.: Representing and using non-functional requirements: a process-oriented approach. IEEE Transactions on Software Engineering (June 1992)
44. Nuseibeh, B., Kramer, J. and Finkelstein, A.: A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications. IEEE Transactions on Software Engineering, 20 (10) (1994)
45. OMG: OMG Unified Modeling Language Specification, version 1.5. Available at: <http://www.omg.org/cgi-bin/doc?formal/03-03-01> (2003)
46. Porter, M. E.: Strategy and the Internet. Harvard Business Review, 79 (March 2001)
47. Rahm E. and Bernstein, P. A.: A survey of approaches to automatic schema matching. VLDB Journal, 10 (2001) 334—350
48. Rappa, M.: Business models on the web. Available at: <http://digitalenterprise.org/models/models.html> (2004)
49. Reenskaug, T., Wold, P. and Lehne, O. A.: Working with Objects: The OOram Software Engineering Method. Manning/Prentice Hall (1996)
50. Reijswoud, V.E. and Dietz, J.L.G.: DEMO Modeling Handbook. www.demo.tudelft.nl (1999) available at: <http://www.demo.nl/documents/handbook.pdf>
51. RosettaNet: RosettaNet Partner Interface Processes (PIPs). Available at: <http://www.rosettanet.org/> (April 2006)
52. Shvaiko, P.: Ontology Matching. Available at: <http://www.ontologymatching.org/> (2005)
53. Sommerville, I., and Sawyer, P.: Viewpoints: principles, problems and a practical approach to requirements engineering. Annals of Software Engineering, 3 (1997) 101—130, available at: <http://citeseer.csail.mit.edu/article/sommerville97viewpoints.html>
54. Sunetnanta, T. and Finkelstein, A.: Automated Consistency Checking for Multiperspective Software Specifications. Workshop on Advanced Separation of Concerns, The 23rd International Conference on Software Engineering (ICSE2001), Toronto, Ontario, Canada (May 2001)
55. Sunetnanta, T.: Multiperspective Development Environment for Configurable Distributed Applications. Ph.D. Thesis, Department of Computing, Imperial College (February 1999)
56. Takeda, H., Veerkamp, P., Tomiyama, T., and Yoshikawa, H.: Modeling design processes. AI Mag. 11 (4) (October 1990) 37—48
57. Telematica Institute: ArchiMate. Available at: <http://www.telin.nl/NetworkedBusiness/ArchiMate/ART/index.html> (March 2005)
58. Timmers, P.: Business models for electronic markets. Electronic Markets, 8(2) (1998) 2—8
59. Weill, P. and Vitale, M.R.: Place to Space: Migrating to Ebusiness Models. Harvard Business School Press, Boston (2001)
60. Weiss, M.: Pattern-Driven Design of Agent Systems: Approach and Case Study. Conference on Advanced Information Systems Engineering (CAiSE), LNCS 2681, Springer (2003)
61. Weiss, M.: Pattern-Driven Design of Agent Systems: Approach and Case Study. Conference on Advanced Information Systems Engineering (CAiSE), LNCS 2681, Springer (2003)
62. Wieringa, R.J. and Gordijn, J.: Value-Oriented Design of Service Coordination Processes: Correctness and Trust. ACM Symposium on Applied Computing (2005)
63. Wieringa, R.J., Blanken, H.M., Fokkinga, M.M. and Grefen, P.W.P.J.: Aligning application architecture to the business context. In: Proceedings of the Conference on Advanced Information System Engineering (CAiSE) (2003)
64. Wieringa, R.J.: Design Methods for Reactive Systems. Morgan Kaufmann (2002)
65. Wikipedia: Breadth-first search. Available at: http://en.wikipedia.org/wiki/Breadth-first_search (March 2007)

66. Wodtke, D. and Weikum, G.: A Formal Foundation for Distributed Workflow Execution Based on State Charts. In: Afrati, F.N. and Kolaitis, P. (eds.): Proceedings of the 6th International Conference on Database Theory (ICDT) (1997) 230—246
67. Wolfram MathWorld: Number theory. Wolfram Research. Available at: <http://mathworld.wolfram.com/PascalsTriangle.html> (January 2006)
68. Wombacher, A., Fankhauser, P. and Aberer, K.: Overview on Decentralized Establishment of Consistent Multi-Lateral Collaborations Based on Asynchronous Communication. Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE) (2005) 164—170
69. WordNet: A lexical database for the English language. Available at: <http://wordnet.princeton.edu/> (August 2005)
70. Yacoub, S. and Ammar, H.: Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems. 1st edition. Addison-Wesley Professional (2003)
71. Yi, X. and Kochut, K.J.: Process Composition of Web Services with Complex Conversation Protocols: a Colored Petri Nets Based Approach. Proceedings of the Design, Analysis, and Simulation of Distributed Systems (2004) 141—148
72. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE), Washington D.C., USA (1997) 226—235
73. Zlatev, Z. and Wombacher, A.: Consistency between e3-value Models and Activity Diagrams in a Multi-Perspective Development Method. In: Meersman, R., Tari, Z., et al. (eds.): Proceedings Part I LNCS 3760/2005: On the Move to Meaningful Internet Systems 2005. CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, Agia Napa, Cyprus (October - November 2005) 520—538
74. Zlatev, Z., Daneva, M. and Wieringa, R.J.: Multi-Perspective Requirements Engineering for Networked Business Systems: A Framework for Pattern Composition. In: 8th Workshop on Requirements Engineering (WER05). Porto (June 2005) 26—37
75. Zlatev, Z., Eck van, P. and Wieringa, R.: Value-exchange patterns in business models of intermediaries that offer negotiation services. Technical Report TR-CTIT-04-26. Centre for Telematics and Information Technology, University of Twente, The Netherlands (June 2004)
76. Zlatev, Z., Eck van, P., Wieringa, R. and Gordijn, J.: Goal-Oriented RE for e-services. In: International Workshop on Service-oriented Requirements Engineering (SORE), Kyoto, Japan (September 2004)

Summary

This thesis defines a design framework and a method for modelling networked businesses. The intended application domain is electronic businesses that extensively use information and communication technology to coordinate work. The key property of the proposed approach is the reuse of design knowledge in the form of design patterns.

Design patterns are extracted from models of existing electronic intermediaries considered successful. These businesses have been reverse-engineered to two types of models: economic value exchange models and business process models. The identified patterns comprise two libraries of value exchange and business process patterns, respectively. Patterns are catalogued with, among others, their context, solved problem, and proposed solution. Most importantly, they are annotated with a machine-readable capability model used as a search key in the library.

Capability models are part of the goal-modelling technique for business requirements proposed here. Our goal-modelling technique operationalizes each business goal with a variable and an evaluation function: the evaluation function determines when a measured variable value satisfies the goal. A goal model represents requirements if goals are assigned evaluation functions but the variable values are unknown. In such a case, the goal model specifies what is desired to happen. If, on the other hand, variable values are known, the goal model documents the capabilities of a pattern.

The proposed design framework structures the development process into: (1) available design knowledge in libraries of value and process patterns, (2) business requirements captured in a goal model, and (3) economic value and business process perspectives to look at a business system. The design method prescribes steps to transform patterns and requirements into a system specification. These include: (i) identification of relevant pattern based on matching capability and requirements goal models; (ii) synthesis of value and process patterns into value and process models, respectively; and (iii) consistency check procedure for value and process model.

The usefulness of the approach is demonstrated in a real-life example, which shows that the framework and method exhibit a predefined set of desired properties.

Samenvatting

Dit proefschrift beschrijft een ontwerp raamwerk en een methode om interactie tussen bedrijven te modeleren. Het beoogde toepassingsdomein omvat bedrijven waarbij veelvuldig en intensief gebruik wordt gemaakt van informatie en communicatie technologie voor het coördineren van werk. De belangrijkste eigenschap van de voorgestelde benadering is de herbruikbaarheid van ontwerp kennis door middel van ontwerp patronen.

De ontwerp patronen zijn ontleend aan modellen van bestaande, succesvolle elektronische tussenpersonen. Deze bedrijven werden ontleed tot twee typen modellen: modellen die de uitwisseling van producten en diensten beschrijven en modellen die de bedrijfsprocessen beschrijven. Patronen zijn onderverdeeld naar, onder andere, hun context, het probleem dat ze oplossen en de gesuggereerde oplossing. Belangrijk is dat de eigenschappen van elk patroon beschreven worden door middel van een gecodeerd toepasselijkheids model dat dienst doet als zoek sleutel in de volledige bibliotheek van patronen.

Toepasselijkheidsmodellen vormen een integraal deel van de hier voorgestelde doelstellingsmodellering techniek. Onze techniek operationaliseert elke economische doelstelling met een variabele en een evaluatie functie: de evaluatiefunctie bepaalt wanneer een verkregen meetwaarde voor een variabele een bepaalde doelstelling vervult. Een doelmodel representeert de gestelde eisen in het geval kwantitatieve drempelwaardes voor de variabelen niet bekend zijn. In dit geval beschrijft het doelmodel het verlangde gedrag. Echter wanneer de drempelwaardes wel bekend zijn legt het doelmodel de toepasselijkheid van het patroon vast.

Het voorgestelde ontwerp raamwerk structureert het ontwerpproces als volgt: (1) beschikbare ontwerp kennis in bibliotheken van waarde- en procespatronen, (2) vereisten en doelstellingen vervat in een doelmodel, en (3) economische waarde en bedrijfsproces modellen om een gegeven economisch systeem te beschrijven. De ontwerp methode schrijft stappen voor om patronen en vereisten in een systeem specificatie om te zetten. Deze omvatten: (i) het identificeren van relevante patronen gebaseerd op het paren van toepasselijkheids- en doelmodellen; (ii) synthese van waarde- en procespatronen in waarde- en procesmodellen; en (iii) een procedure om de consistentie van waarde- en procesmodellen te toetsen.

De waarde van deze aanpak wordt aangetoond met een voorbeeld uit de praktijk, waaruit blijkt dat het raamwerk en de methode voldoen aan een aantal, van tevoren vastgelegde, gewenste eigenschappen.

SIKS Dissertation Series

- 1998-1 Johan van den Akker (CWI), DEGAS - An Active, Temporal Database of Autonomous Objects
1998-2 Floris Wiesman (UM), Information Retrieval by Graphically Browsing Meta-Information
1998-3 Ans Steuten (TUD), A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
1998-4 Dennis Breuker (UM), Memory versus Search in Games
1998-5 E.W.Oskamp (RUL), Computerondersteuning bij Straftoemeting
1999-1 Mark Sloof (VU), Physiology of Quality Change Modelling: Automated modelling of Quality Change of Agricultural Products
1999-2 Rob Potharst (EUR), Classification using decision trees and neural nets
1999-3 Don Beal (UM), The Nature of Minimax Search
1999-4 Jacques Penders (UM), The practical Art of Moving Physical Objects
1999-5 Aldo de Moor (KUB), Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
1999-6 Niek J.E. Wijngaards (VU), Re-design of compositional systems
1999-7 David Spelt (UT), Verification support for object database design
1999-8 Jacques H.J. Lenting (UM), Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.
2000-1 Frank Niessink (VU), Perspectives on Improving Software Maintenance
2000-2 Koen Holtman (TUE), Prototyping of CMS Storage Management
2000-3 Carolien M.T. Metselaar (UVA), Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.
2000-4 Geert de Haan (VU), ETAG, A Formal Model of Competence Knowledge for User Interface Design
2000-5 Ruud van der Pol (UM), Knowledge-based Query Formulation in Information Retrieval.
2000-6 Rogier van Eijk (UU), Programming Languages for Agent Communication
2000-7 Niels Peek (UU), Decision-theoretic Planning of Clinical Patient Management
2000-8 Veerle Coup, (EUR), Sensitivity Analysis of Decision-Theoretic Networks
2000-9 Florian Waas (CWI), Principles of Probabilistic Query Optimization
2000-10 Niels Nes (CWI), Image Database Management System Design Considerations, Algorithms and Architecture
2000-11 Jonas Karlsson (CWI), Scalable Distributed Data Structures for Database Management
2001-1 Silja Renooij (UU), Qualitative Approaches to Quantifying Probabilistic Networks
2001-2 Koen Hindriks (UU), Agent Programming Languages: Programming with Mental Models
2001-3 Maarten van Someren (UvA), Learning as problem solving
2001-4 Evgueni Smirnov (UM), Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
2001-5 Jacco van Ossenbruggen (VU), Processing Structured Hypermedia: A Matter of Style
2001-6 Martijn van Welie (VU), Task-based User Interface Design
2001-7 Bastiaan Schonhage (VU), Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU), A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
2001-9 Pieter Jan 't Hoen (RUL), Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
2001-10 Maarten Sierhuis (UvA), Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
2001-11 Tom M. van Engers (VUA), Knowledge Management: The Role of Mental Models in Business Systems Design
2002-01 Nico Lassing (VU), Architecture-Level Modifiability Analysis
2002-02 Roelof van Zwol (UT), Modelling and searching web-based document collections
2002-03 Henk Ernst Blok (UT), Database Optimization Aspects for Information Retrieval
2002-04 Juan Roberto Castelo Valdueza (UU), The Discrete Acyclic Digraph Markov Model in Data Mining
2002-05 Radu Serban (VU), The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
2002-06 Laurens Mommers (UL), Applied legal epistemology; Building a knowledge-based ontology of the legal domain
2002-07 Peter Boncz (CWI), Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

- 2002-08 Jaap Gordijn (VU), Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
- 2002-09 Willem-Jan van den Heuvel(KUB), Integrating Modern Business Applications with Objectified Legacy Systems
- 2002-10 Brian Sheppard (UM), Towards Perfect Play of Scrabble
- 2002-11 Wouter C.A. Wijngaards (VU), Agent Based Modelling of Dynamics: Biological and Organisational Applications
- 2002-12 Albrecht Schmidt (Uva), Processing XML in Database Systems
- 2002-13 Hongjing Wu (TUE), A Reference Architecture for Adaptive Hypermedia Applications
- 2002-14 Wieke de Vries (UU), Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
- 2002-15 Rik Eshuis (UT), Semantics and Verification of UML Activity Diagrams for Workflow Modelling
- 2002-16 Pieter van Langen (VU), The Anatomy of Design: Foundations, Models and Applications
- 2002-17 Stefan Manegold (UVA), Understanding, Modeling, and Improving Main-Memory Database Performance
- 2003-01 Heiner Stuckenschmidt (VU), Ontology-Based Information Sharing in Weakly Structured Environments
- 2003-02 Jan Broersen (VU), Modal Action Logics for Reasoning About Reactive Systems
- 2003-03 Martijn Schuemie (TUD), Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
- 2003-04 Milan Petkovic (UT), Content-Based Video Retrieval Supported by Database Technology
- 2003-05 Jos Lehmann (UVA), Causation in Artificial Intelligence and Law - A modelling approach
- 2003-06 Boris van Schooten (UT), Development and specification of virtual environments
- 2003-07 Machiel Jansen (UvA), Formal Explorations of Knowledge Intensive Tasks
- 2003-08 Yongping Ran (UM), Repair Based Scheduling
- 2003-09 Rens Kortmann (UM), The resolution of visually guided behaviour
- 2003-10 Andreas Lincke (UvT), Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
- 2003-11 Simon Keizer (UT), Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
- 2003-12 Roeland Ordelman (UT), Dutch speech recognition in multimedia information retrieval
- 2003-13 Jeroen Donkers (UM), Nosce Hostem - Searching with Opponent Models
- 2003-14 Stijn Hoppenbrouwers (KUN), Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
- 2003-15 Mathijs de Weerd (TUD), Plan Merging in Multi-Agent Systems
- 2003-16 Menzo Windhouwer (CWI), Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
- 2003-17 David Jansen (UT), Extensions of Statecharts with Probability, Time, and Stochastic Timing
- 2003-18 Levente Kocsis (UM), Learning Search Decisions
- 2004-01 Virginia Dignum (UU), A Model for Organizational Interaction: Based on Agents, Founded in Logic
- 2004-02 Lai Xu (UvT), Monitoring Multi-party Contracts for E-business
- 2004-03 Perry Groot (VU), A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
- 2004-04 Chris van Aart (UVA), Organizational Principles for Multi-Agent Architectures
- 2004-05 Viara Popova (EUR), Knowledge discovery and monotonicity
- 2004-06 Bart-Jan Hommes (TUD), The Evaluation of Business Process Modeling Techniques
- 2004-07 Elise Boltjes (UM), Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
- 2004-08 Joop Verbeek(UM), Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiegegevensuitwisseling en digitale expertise
- 2004-09 Martin Caminada (VU), For the Sake of the Argument; explorations into argument-based reasoning
- 2004-10 Suzanne Kabel (UVA), Knowledge-rich indexing of learning-objects
- 2004-11 Michel Klein (VU), Change Management for Distributed Ontologies
- 2004-12 The Duy Bui (UT), Creating emotions and facial expressions for embodied agents
- 2004-13 Wojciech Jamroga (UT), Using Multiple Models of Reality: On Agents who Know how to Play
- 2004-14 Paul Harrenstein (UU), Logic in Conflict. Logical Explorations in Strategic Equilibrium
- 2004-15 Arno Knobbe (UU), Multi-Relational Data Mining
- 2004-16 Federico Divina (VU), Hybrid Genetic Relational Search for Inductive Learning

- 2004-17 Mark Winands (UM), Informed Search in Complex Games
- 2004-18 Vania Bessa Machado (UvA), Supporting the Construction of Qualitative Knowledge Models
- 2004-19 Thijs Westerveld (UT), Using generative probabilistic models for multimedia retrieval
- 2004-20 Madelon Evers (Nyenrode), Learning from Design: facilitating multidisciplinary design teams
- 2005-01 Floor Verdenius (UVA), Methodological Aspects of Designing Induction-Based Applications
- 2005-02 Erik van der Werf (UM), AI techniques for the game of Go
- 2005-03 Franc Grootjen (RUN), A Pragmatic Approach to the Conceptualisation of Language
- 2005-04 Nirvana Meratnia (UT), Towards Database Support for Moving Object data
- 2005-05 Gabriel Infante-Lopez (UVA), Two-Level Probabilistic Grammars for Natural Language Parsing
- 2005-06 Pieter Spronck (UM), Adaptive Game AI
- 2005-07 Flavius Frasinca (TUE), Hypermedia Presentation Generation for Semantic Web Information Systems
- 2005-08 Richard Vdovjak (TUE), A Model-driven Approach for Building Distributed Ontology-based Web Applications
- 2005-09 Jeen Broekstra (VU), Storage, Querying and Inferencing for Semantic Web Languages
- 2005-10 Anders Bouwer (UVA), Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
- 2005-11 Elth Ogston (VU), Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
- 2005-12 Csaba Boer (EUR), Distributed Simulation in Industry
- 2005-13 Fred Hamburg (UL), Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
- 2005-14 Borys Omelayenko (VU), Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
- 2005-15 Tibor Bosse (VU), Analysis of the Dynamics of Cognitive Processes
- 2005-16 Joris Graaumanns (UU), Usability of XML Query Languages
- 2005-17 Boris Shishkov (TUD), Software Specification Based on Re-usable Business Components
- 2005-18 Danielle Sent (UU), Test-selection strategies for probabilistic networks
- 2005-19 Michel van Dartel (UM), Situated Representation
- 2005-20 Cristina Coteanu (UL), Cyber Consumer Law, State of the Art and Perspectives
- 2005-21 Wijnand Derks (UT), Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics
- 2006-01 Samuil Angelov (TUE), Foundations of B2B Electronic Contracting
- 2006-02 Cristina Chisalita (VU), Contextual issues in the design and use of information technology in organizations
- 2006-03 Noor Christoph (UVA), The role of metacognitive skills in learning to solve problems
- 2006-04 Marta Sabou (VU), Building Web Service Ontologies
- 2006-05 Cees Pierik (UU), Validation Techniques for Object-Oriented Proof Outlines
- 2006-06 Ziv Baida (VU), Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
- 2006-07 Marko Smiljanic (UT), XML schema matching -- balancing efficiency and effectiveness by means of clustering
- 2006-08 Eelco Herder (UT), Forward, Back and Home Again - Analyzing User Behavior on the Web
- 2006-09 Mohamed Wahdan (UM), Automatic Formulation of the Auditor's Opinion
- 2006-10 Ronny Siebes (VU), Semantic Routing in Peer-to-Peer Systems
- 2006-11 Joeri van Ruth (UT), Flattening Queries over Nested Data Types
- 2006-12 Bert Bongers (VU), Interactivation - Towards an e-cology of people, our technological environment, and the arts
- 2006-13 Henk-Jan Lebbink (UU), Dialogue and Decision Games for Information Exchanging Agents
- 2006-14 Johan Hoorn (VU), Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
- 2006-15 Rainer Malik (UU), CONAN: Text Mining in the Biomedical Domain
- 2006-16 Carsten Riggelsen (UU), Approximation Methods for Efficient Learning of Bayesian Networks
- 2006-17 Stacey Nagata (UU), User Assistance for Multitasking with Interruptions on a Mobile Device
- 2006-18 Valentin Zhizhkin (UVA), Graph transformation for Natural Language Processing
- 2006-19 Birna van Riemsdijk (UU), Cognitive Agent Programming: A Semantic Approach
- 2006-20 Marina Velikova (UvT), Monotone models for prediction in data mining
- 2006-21 Bas van Gils (RUN), Aptness on the Web
- 2006-22 Paul de Vrieze (RUN), Fundamentals of Adaptive Personalisation
- 2006-23 Ion Juvina (UU), Development of Cognitive Model for Navigating on the Web

- 2006-24 Laura Hollink (VU), Semantic Annotation for Retrieval of Visual Resources
- 2006-25 Madalina Drugan (UU), Conditional log-likelihood MDL and Evolutionary MCMC
- 2006-26 Vojkan Mihajlovic (UT), Score Region Algebra: A Flexible Framework for Structured Information Retrieval
- 2006-27 Stefano Bocconi (CWI), Vox Populi: generating video documentaries from semantically annotated media repositories
- 2006-28 Borkur Sigurbjornsson (UVA), Focused Information Access using XML Element Retrieval
- 2007-01 Kees Leune (UvT), Access Control and Service-Oriented Architectures
- 2007-02 Wouter Teepe (RUG), Reconciling Information Exchange and Confidentiality: A Formal Approach
- 2007-03 Peter Mika (VU), Social Networks and the Semantic Web
- 2007-04 Jurriaan van Diggelen (UU), Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
- 2007-05 Bart Schermer (UL), Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance
- 2007-06 Gilad Mishne (UVA), Applied Text Analytics for Blogs
- 2007-07 Natasa Jovanovic (UT), To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
- 2007-08 Mark Hoogendoorn (VU), Modeling of Change in Multi-Agent Organizations
- 2007-09 David Mobach (VU), Agent-Based Mediated Service Negotiation
- 2007-10 Huib Aldewereld (UU), Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols
- 2007-11 Natalia Stash (TUE), Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
- 2007-12 Marcel van Gerven (RUN), Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
- 2007-13 Rutger Rienks (UT), Meetings in Smart Environments; Implications of Progressing Technology
- 2007-14 Niek Bergboer (UM), Context-Based Image Analysis
- 2007-15 Joyca Lacroix (UM), NIM: a Situated Computational Memory Model
- 2007-16 Davide Grossi (UU), Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
- 2007-17 Theodore Charitos (UU), Reasoning with Dynamic Networks in Practice
- 2007-18 Bart Orriens (UvT), On the development and management of adaptive business collaborations
- 2007-19 David Levy (UM), Intimate relationships with artificial partners
- 2007-20 Slinger Jansen (UU), Customer Configuration Updating in a Software Supply Network
- 2007-21 Karianne Vermaas (UU), Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
- 2007-22 Zlatko Zlatev (UT), Goal-oriented design of value and process models from patterns