

EPC To BPEL Transformations

Lucas O. Meertens
epctobpel@lmeertens.nl

EPC To BPEL Transformations

January 2009

Graduation thesis of:

Lucas O. Meertens
Student number 0037141
Business Information Technology
University of Twente
epctobpel@lmeertens.nl

On behalf of:

Sogeti Nederland B.V.
Lange Dreef 17
4131 NJ Vianen

Under supervision of:

Dr. M.E. Iacob (University of Twente, faculty of MB)
S.M. Eckartz, MSc. (University of Twente, faculty of EEMCS)
M. van Es (Sogeti Nederland B.V., division DSE)

Abstract

Companies looking to improve their business processes can choose from several approaches. Almost all of these include modeling the processes. Implementing the modeled processes is the next step. Usually, developers create the required software, based on the models. However, often the resulting code does not meet the demands of the business. In order to improve the transition from business process models to code, Model-Driven Engineering (MDE) provides a promise by using model transformation. This promise consists of the ability to change business models into code automatically.

This research investigates one of the possible model transformations. Namely, the transformation from Event-driven Process Chains (EPC) models to Business Process Execution Language (BPEL) specifications. Business modelers use EPC to create process models of the control flow. IT developers can use the resulting BPEL specifications as executable code, which contains the control flow.

A conceptual model provides a method for evaluating model transformation. Ontology (the BWW model) and workflow patterns form its basis. According to the conceptual model, it is possible to transform most patterns and constructs from EPC to BPEL. However, one pattern is impossible to transform, and several constructs cause ambiguities. A conceptual mapping from EPC to BPEL offers one way to deal with these issues.

To verify the conceptual model, and test what is possible in practice, several diagrams are transformed. The Oracle BPA Suite serves as the environment for these experiments. It is able to transform most of the workflow patterns and constructs, which the conceptual model predicts. It does this in a different, yet correct, way from the conceptual mapping. The only unexpected issue that arose was the inability to transform the OR-connector. Applying a case from practice illustrates more difficulties. It supplies several extra limitations, such as the incorrect transformation of a while-loop within a parallel structure.

Eleven guidelines present a route to EPC models that are transformable by the Oracle BPA Suite. Creating structured diagrams is a means to work around most of the limitations. If

structuring is impossible, decomposing the model into smaller diagrams is a solution. This can lead to irrational diagrams from a modeler's point of view, though.

This research contributes to theory and practice in three ways. First, it contributes to theory by expanding the knowledge of model transformation with the conceptual model. The second contribution is the validation the conceptual model for the specific case of EPC to BPEL transformation as done by the Oracle BPA Suite. The discovered limitations show what to expect during model transformation in practice. The third contribution is a list of guidelines, which modelers can apply to improve the feasibility of EPC to BPEL transformations.

Samenvatting

Organisaties hebben meerdere mogelijkheden om bedrijfsprocessen te verbeteren. De meeste methoden omvatten het modeleren van de processen. Implementeren van de gemodelleerde processen is het volgende doel. Op basis van de modellen bouwen ontwikkelaars meestal de benodigde applicaties. Helaas voldoet de gecreëerde code vaak niet aan de eisen van de organisatie. De belofte van Model-Driven Engineering (MDE) om de ontwikkeling van model naar code te verbeteren is gebaseerd op modeltransformatie. Hierbij worden bedrijfsprocesmodellen automatisch omgezet naar code.

Dit onderzoek bekijkt een van de vormen van modeltransformatie, namelijk de transformatie van modellen in Event-driven Process Chains (EPC) naar Business Process Execution Language (BPEL) specificaties. Vanaf de bedrijfskant gebruiken modelleers EPC om procesmodellen in kaart te brengen, die het gedrag binnen het proces beschrijven. Ontwikkelaars aan de ICT kant kunnen de resulterende BPEL specificaties, die het gedrag bevatten, gebruiken als uitvoerbare code.

Het conceptueel raamwerk biedt een manier om modeltransformaties te analyseren. Ontologie (het BWW model) en procespatronen vormen de basis van het raamwerk. De meeste bouwstenen en patronen zijn transformeerbaar van EPC naar BPEL volgens het raamwerk. Eén patroon is niet transformeerbaar, en meerdere bouwstenen veroorzaken onduidelijkheid voor transformatie. Om met deze problemen om te gaan verschaft het raamwerk één mogelijke vertaling.

Verscheidene diagrammen dienen als invoer voor transformatie, om het raamwerk te verifiëren en na te gaan wat in de praktijk mogelijk is. De Oracle BPA Suite dient als platform voor dit experiment. Transformatie van de meeste bouwstenen en patronen verloopt zoals het raamwerk voorspelt. Dat het platform de “OR-connector” (OF-verbinding) niet kan transformeren, is het enige onverwachte. Verder gebruikt het platform een andere, maar ook juiste, vertaling. In tegenstelling tot de onderdelen uit het raamwerk, veroorzaakt een voorbeeldstudie uit de praktijk meer problemen. Transformatie van de voorbeeldstudie levert meer beperkingen op, zoals de verkeerde transformatie van een lus binnen een parallelle samenstelling.

Toepassing van elf voorschriften levert EPC modellen die de Oracle BPA Suite kan transformeren. De meeste beperkingen zijn omzeilbaar, door de EPC diagrammen te structureren volgens het principe, dat de laatst geopende verbinding als eerst gesloten wordt. Ontleden van het model in kleinere diagrammen is een oplossing, als structureren onmogelijk blijkt. Deze oplossing leidt mogelijk tot modellen, die onlogisch zijn vanuit het standpunt van een modelleur.

Dit onderzoek draagt op drie manieren bij aan theorievorming en de praktijk. Ten eerste breidt het de kennis over modeltransformatie uit, in de vorm van het conceptuele raamwerk voor het analyseren van modeltransformaties. Ten tweede valideert het onderzoek dit raamwerk voor het geval van EPC naar BPEL transformatie, zoals de Oracle BPA Suite die uitvoert. Voor de praktijk betekent dit dat duidelijk is welke beperkingen te verwachten zijn bij transformatie. Ten derde verbeteren de elf voorschriften voor modelleurs de haalbaarheid van EPC naar BPEL transformaties.

Preface

This thesis is part of the final assignment for a Master of Science degree in Business and Information Technology at the University of Twente. Several pleasant years, I have spent at the university, and I hope to spend some more there. The research for this assignment was conducted at Sogeti Nederland B.V. Most of my time during the past few months was spent at their office in Amersfoort or on the way to and from there. Sogeti provided me with the opportunity and resources required to do this research.

I would like to thank my supervisors at the university, Maria Iacob and Silja Eckartz, and at Sogeti, Sander Bosma and Margot van Es, for providing me with ideas, motivation, guidance, and constructive criticism. Furthermore, I would like to thank my parents for supporting me throughout my study. Finally, I would like to thank my girlfriend, Woutske Hartholt, for putting up with me having so little time for her.

Enschede, January 2009

Lucas Meertens

Table of Contents

Abstract.....	i
Samenvatting	iii
Preface	v
Table of Contents.....	vi
1 Introduction.....	1
1.1 Background.....	1
1.1.1 The worlds of business and IT.....	1
1.1.2 Sogeti (and the two worlds).....	2
1.1.3 Model transformation	3
1.2 Scope	3
1.3 Outline.....	4
2 Research Design.....	5
2.1 Research Objective.....	5
2.2 Research Questions.....	6
2.3 Research Model.....	7
2.4 Material for Investigation	8
2.5 Research Strategy.....	9
3 Related research.....	11
3.1 Ontology.....	11
3.1.1 Ontology for information systems.....	12
3.1.2 The choice for the BWW model.....	13
3.2 Workflow patterns	14
3.2.1 The source for patterns	14
3.2.2 Evaluation of patterns.....	15
3.2.3 Application	15

3.3	Event-driven Process Chain (EPC)	16
3.3.1	History	16
3.3.2	(Non-) local semantics	17
3.3.3	Extensions	17
3.4	Business Process Execution Language (BPEL)	17
3.4.1	History	18
3.4.2	Issues	18
3.4.3	Web services	19
3.4.4	Versioning	19
3.5	Model-Driven Engineering (MDE)	19
3.5.1	History	20
3.5.2	Levels of abstraction	20
3.5.3	Round-Trip Engineering	21
3.6	Model transformation	22
3.6.1	Transforming to BPEL	23
3.6.2	Transforming from EPC	23
3.6.3	Transforming from EPC to BPEL	24
3.7	Oracle BPA Suite	24
3.7.1	Components	25
3.7.2	Integration	25
3.8	Discussion of treated literature	26
4	A framework to evaluate model transformation	27
4.1	Ontological analysis of business process modeling languages	27
4.1.1	Completeness and clarity	27
4.1.2	BPEL and the BWW model	28
4.1.3	EPC and the BWW model	29
4.1.4	Representational power of EPC versus BPEL	29
4.2	(Business) Process patterns	32
4.2.1	Patterns in BPEL	32

4.2.2	Patterns in EPC.....	33
4.2.3	Limitations of transforming patterns from EPC to BPEL.....	33
4.3	Discussion of encountered issues	34
4.3.1	Possible solutions.....	34
4.3.2	Implementation	36
4.3.3	Other considerations	36
4.4	Conceptual mapping from EPC to BPEL	37
4.4.1	EPC construct based mapping	38
4.4.2	Pattern-based mapping	39
4.4.3	Ontology-based mapping.....	46
4.5	Concluding the conceptual model	48
5	Pattern Transformation Results	50
5.1	Input: EPC diagrams	50
5.2	Process: creating and transforming diagrams	51
5.3	General BPEL code generation.....	53
5.4	Transformation of Patterns.....	54
5.4.1	Diagram 1: WFCP 1 – Sequence.....	54
5.4.2	Diagram 2: WFCP 2 – Parallel Split & WFCP 3 - Synchronization	56
5.4.3	Diagram 3: WFCP 4 – Exclusive Choice & WFCP 5 – Simple Merge.....	57
5.4.4	Diagram 4: WFCP 6 – Multi Choice & WFCP 7 – Synchronizing Merge	59
5.4.5	Diagram 5: WFCP 11 – Implicit Termination.....	61
5.4.6	Diagrams 6 and 7: WFCP 10 – Arbitrary Cycles	62
5.5	Conclusion of pattern transformation	63
6	Validation: a composite case from practice	65
6.1	Case description: “Accounting close”	65
6.2	Transforming individual sub-processes.....	67
6.3	Diagram 8: Prepare accounting close	67
6.4	Diagram 9: Determine cost levels	68

6.5	Diagram 10: Check final hour download.....	68
6.6	Diagram 11: Preliminary rebilling.....	69
6.7	Diagram 12: Update report data.....	69
6.8	Diagram 13: Report	70
6.9	Diagram 14: Full composite diagram – Accounting close	70
6.10	Conclusion of case transformation	71
7	Guidelines for modeling	73
7.1	Criteria.....	73
7.2	Limitation 1: Construct excess (OR-connector)	75
7.3	Limitation 2: Construct overload and redundancy	76
7.4	Limitation 3: Pattern incompatibility (WFCP 10 - Arbitrary Cycle)	77
7.5	Limitation 4: Multiple start events.....	78
7.6	Limitation 5: Degree of connectors.....	78
7.7	Limitation 6: Multiple end events	79
7.8	Limitation 7: Combination of constructs and structures	80
7.9	Limitation 8: Block-structured versus graph-structured.....	80
7.10	General guidelines.....	82
7.11	Applying the guidelines	82
7.12	Validation of the guidelines in the composite case	84
7.12.1	Applying the guidelines while modeling.....	84
7.12.2	Applying the guidelines to an existing model.....	85
8	Discussion	89
8.1	Validity.....	89
8.2	Limitations.....	90
8.3	Alternatives	90

8.4	Further Research	91
8.5	Recommendations	92
9	Conclusions.....	94
9.1	Answers to research sub-questions	94
9.2	Answers to main research questions	95
9.3	Contributions.....	97
	List of References.....	98
	Appendix A - Output BPEL diagrams	I
	Appendix B - Output BPEL code.....	XI
	Appendix C - Original case EPC diagrams	XXXIX
	Appendix D - Composite case EPC diagrams	XLV
	Appendix E - EPC diagrams of full case.....	LI
	Appendix F - Transformable, modified EPC diagrams	LVI
	Appendix G - EPC diagrams decomposed from full model.....	LX

1 Introduction

Due to globalization and economic crisis, companies feel an increased market pressure. In response, they look for new ways of improving their business processes. To achieve processes that are more efficient, several established approaches are available. Examples are Business Process Management (BPM), Business Process Engineering (BPE), and Business Process Reengineering (BPR). Each of these approaches has their own set of supporting tools. A recent initiative introduced an extra means to support the existing approaches, by making the improved business process executable. This new approach goes under the name of Model-Driven Engineering (MDE) (Kent, 2002). The central idea of MDE is that models can transform into other models. While this idea sounds trivial, it allows transformation from business process models to executable models. This research handles a specific case of this model-to-model transformation.

1.1 Background

1.1.1 The worlds of business and IT

This research classifies two separate worlds: The business world of the modelers, and the information technology (IT) world of the developers. This research defines the modelers as the people who create and manage the business process models. These models are their main artifacts. They are generally business architects, business modelers, requirements engineers, and process analysts. On the other hand, developers are the people who create and edit executable code and applications. The created code is their main artifact. Developers are generally programmers, software engineers, and software architects. While this is a coarse division, it serves the purpose of this research.

From the business world point of view, the current volatile and competitive environment forces companies to be more flexible and agile. Long development and life cycles of IT systems are a hindering factor for such demands. Business needs to close the gap between strategy and IT (Peppard & Ward, 1999). MDE helps to close this gap by reducing human involvement and providing clearer artifacts to the IT world. This reduction requires less communication and leaves less room for errors. Tooling automatically validates and transforms a model into code. Therefore, agility and flexibility improve, as this needs less

coding. If the tooling also supports round-trip engineering, then it allows continuous process improvement to take place, as is required from a BPM perspective (Smith & Fingar, 2003). MDE provides the promise of achieving the business demands.

From the IT world point of view, the demands of the business world often appear to be unreasonable. IT has to work within strict specifications, limited budget, and harsh deadlines. For developers, the constraints often mean that they compromise one of the three constraints to achieve the others. The concession results in either hurried, over-budget, or sub-standard projects. MDE automates a large part of the process, and provides clearer artifacts. This frees up budget and time for the developers. Furthermore, maintenance is often a significant cost driver, due to poor development in the past and frequent changes to systems. Applying MDE allows modelers to change the systems correctly according to the business process. This saves work on the implementation of changes by maintenance. In conclusion, MDE enables IT to achieve the demands of business.

1.1.2 Sogeti (and the two worlds)

This research was conducted on behalf of Sogeti Nederland B.V. Sogeti provides IT services to businesses and public-sector organizations. Sogeti employs about 18,000 employees, of which about 3,400 work in The Netherlands. It is a wholly owned subsidiary of the international Capgemini organization. Sogeti Nederland B.V. has a divisional structure. Two of these divisions are of particular interest to this research, as they correspond to the two worlds of IT and business.

The division Architecture & Business Solutions (A&BS) first looks at the business objectives of customers. Based on this, they examine which business processes, systems, and information the customers require. They use methodologies such as Architecture of Integrated Information Systems (ARIS) (Scheer & Schneider, 1992) and Dynamic Architecture (DYA) (Wagter, Van den Berg, Luijpers, & Van Steenberghe, 2001). As this division focuses on the business, its professionals are in the modeler category of the business world.

The division Distributed Software Engineering (DSE) designs, develops, and maintains complex IT systems. They do this based on leading technology, such as the Oracle

development platform. Their aim is to deliver maintainable, future proof systems, which match the customer's demands. The professionals working for DSE represent the developers of the IT world.

1.1.3 Model transformation

MDE is an approach with many aspects. Of these aspects, this research focuses on model transformation. This resides at the heart of MDE. The specific type of model transformation under investigation is the transformation from Event-driven Process Chain (EPC) (Scheer & Schneider, 1992) models to Business Process Execution Language (BPEL) specifications (OASIS, 2003). EPC models are the notation of ARIS for the control flow of business processes. The business world uses these to model part of their business processes, using tools such as IDS Scheer's Architect. BPEL is the de facto standard for orchestrating web services. The IT world uses BPEL to model and execute the control flow from web service to web service, for example in a Service Oriented Architecture (SOA). For this, they use tools such as Oracle's JDeveloper, part of the Oracle SOA Suite.

For the direct transformation from EPC to BPEL, only two tools are available. These are the Oracle BPA Suite and IDS Scheer SOA Architect. The Oracle BPA Suite has the SOA Architect as basis, through an OEM license. Both tools have the same basis, but the Oracle BPA Suite is freely available in the form of an evaluation version. Therefore, this research uses only the Oracle BPA Suite for transformation.

The Oracle BPA Suite includes the Business Process Architect component. This component allows specifying business processes, using de facto standards, such as BPMN and BPEL, as well as the EPC models from ARIS. The product descriptions promise that, with a click on a button, transformation is possible from modeled EPC business processes to BPEL specifications (Oracle, 2008). From there, it is possible to generate, orchestrate, and execute web services, with the Oracle SOA Suite (Oracle, n.d.). Oracle claims that their SOA Suite and BPA Suite together are an integrated environment, enabling round-trip engineering.

1.2 Scope

This research tackles the empirical problem of determining the quality of model transformation. For that reason, it creates a conceptual model to evaluate model

transformation. The focus is on transforming EPC models to BPEL specifications, especially discovering the feasibility of automated transformation from EPC to BPEL. The model is put to the test in the Oracle BPA Suite, by transforming a series of diagrams. If any limitations and difficulties arise during transformation, then guidelines help the modeler to solve and avoid them. These guidelines improve the feasibility of model transformation. Together, these parts lead to a conclusion on the feasibility of the transformation of EPC models to BPEL specifications.

1.3 Outline

The results consist of three main parts. Before handling these parts, chapter 2 provides the research design. The design copes with the research objective and questions, as well as the way in which the research answers them. Chapter 3 presents a review of existing literature on the topics of this research. This includes the evaluation criteria, the modeling languages, model transformation, and the Oracle BPA Suite. Chapter 4 provides the first part of the results, a conceptual model for the evaluation of model transformation. The conceptual model comprises two sets of evaluation criteria, ontology and patterns. Besides these criteria, it offers a conceptual mapping from EPC to BPEL. The second part of the results starts with chapter 5, which presents the results of transforming small EPC diagrams based on the two criteria. Then, chapter 6 presents the results of transforming a larger EPC model based on a case from practice. The third part of the results resides in chapter 7, which presents guidelines for modelers. These guidelines help modelers to create EPC models that successfully and correctly transform to BPEL. The limitations found in the previous chapters serve as a basis for the guidelines. The last two chapters finalize the research. Chapter 8 discusses the research. It gives a view on the validity, points to issues for further research, and offers recommendations. Chapter 9 concludes the research, with answers to the research questions, and an evaluation of the contributions it made to theory and practice.

2 Research Design

As the research explores a single type of transformation within a single product, several choices are clear from the start. The research aims at providing a profound examination of the EPC to BPEL transformation, in a qualifying manner. Studying existing literature on related topics forms a basis. The basis is used to build a conceptual model as theoretical research. Empirical investigation tests this theory. Finally, guidelines are devised based on the results.

This chapter starts with a description of the research objective. Then, it specifies the research questions that require answers, in order to reach the research objective, as well as how to answer these questions. Finally, the chapter defines the materials used and explains the research strategy.

2.1 Research Objective

Within the scope of model transformation, this research contributes to three goals. First of all, it expands the knowledge about model-to-code transformation, especially the feasibility of EPC to BPEL transformations. The knowledge is achieved by comparing EPC to BPEL based on their ontological properties and their ability to represent standard workflow patterns. The acquired knowledge forms a general theoretical model on transformation. The second goal is to validate and explain this conceptual model in practice. Assessing results of transforming EPC models to BPEL specifications, as done by the Oracle BPA Suite, contributes to reaching this goal. Such a review reveals difficulties and limitations one can expect to face, during model transformation in general and with the Oracle BPA Suite in particular. The final goal is to improve the potential of success in such an endeavor. Based on the findings of the first two parts, a set of best practices and guidelines for EPC business modeling and models is devised to make progress towards unproblematic transformation. Together, these three parts help to derive the business impact of model transformation and the Oracle BPA Suite.

2.2 Research Questions

In order to reach the objectives of this research, several questions require answers. Each of the main research questions corresponds to a limited fraction of the research model (see section 2.3). When all main research questions are answered, the research objectives are reached.

1. To what extent is automated transformation from EPC models to BPEL specifications possible?
 - a. To what ontology must business process modeling languages adhere?
 - b. Which business process patterns are commonly used in business processes?
 - c. How do the constructs of each language relate to the patterns and ontology?
 - d. How do the constructs map from EPC to BPEL in theory?
2. What are the effectiveness and the limitations of (partially) automated transformations from EPC models to BPEL specifications, as supported by the Oracle BPA Suite?
 - a. What are the known limitations of such transformations?
 - b. Which patterns, concepts, and constructs do not transform successfully?
 - c. Are acceptable workarounds available for those cases?
3. What modeling guidelines must modelers follow, to enable (partially) automated transformations from EPC models to BPEL specifications, as supported by the Oracle BPA Suite?
 - a. What is the level of detail required for EPC diagrams, compared to that required by BPEL specifications?
 - b. What type of information must EPC models include before transformation can take place?
 - c. Are acceptable workarounds available for limitations that need them?
 - d. How must EPC models be structured (e.g. composed of patterns and avoid patterns) to allow transformation?
 - e. Is such a structuring acceptable?

The main research questions match to the chapters in this document in the following way: The conceptual model in chapter 4 answers question 1 by comparing the two languages, based on ontology and patterns as criteria. The direct results of the empirical research in

chapters 5 and 6 reveal limitations. Confronting the empirical results to the conceptual criteria provides the evaluation of question 2. Finally, chapter 7 prescribes the guidelines that the last question asks for, based on the earlier findings. The remaining chapters discuss and conclude the overall findings, and evaluate the research.

2.3 Research Model

In order to conduct the research successfully, a research model is formulated. Figure 1 provides a visual representation of this model. The following paragraphs specify a textual interpretation (Verschuren & Doorewaard, 1998).

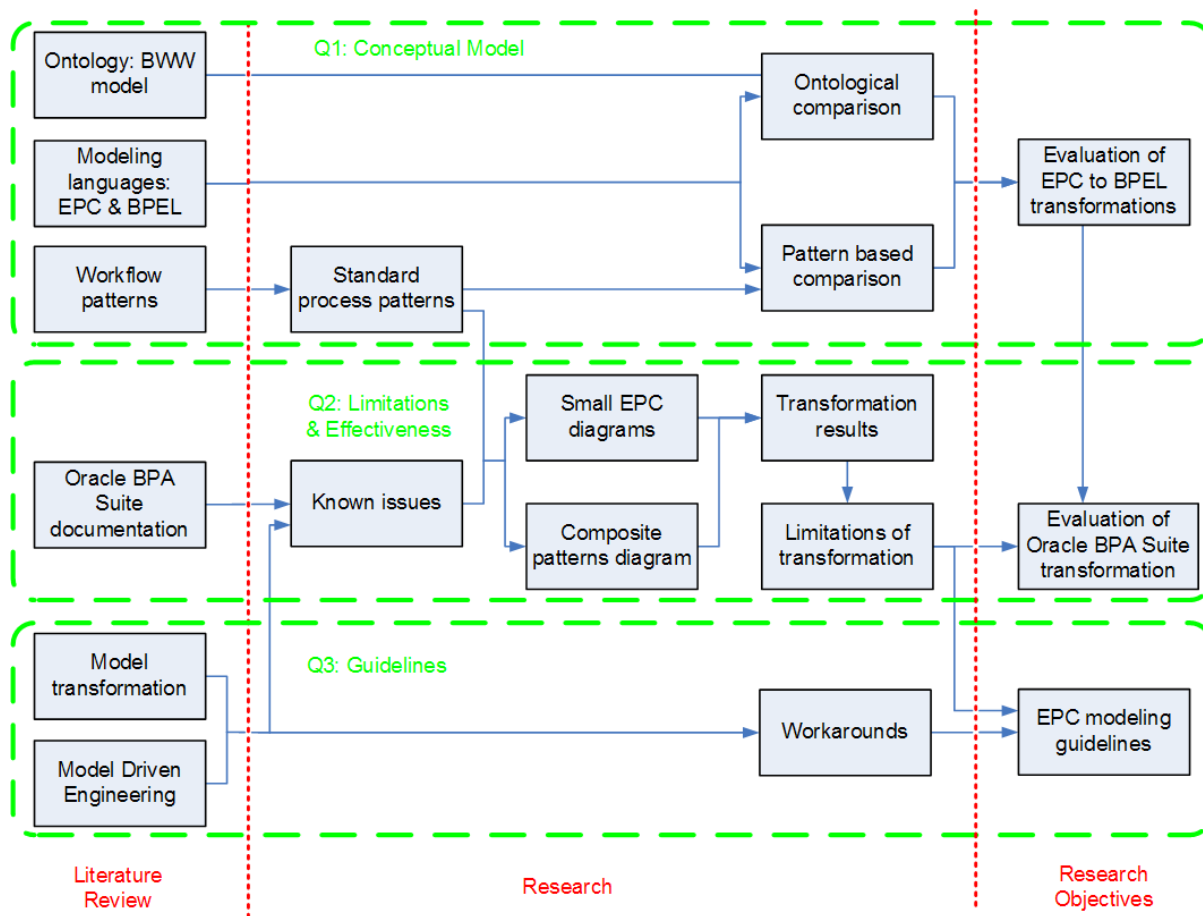


Figure 1: Research Model

The research starts with a review of selected literature on modeling. The focal points are workflow patterns and ontology: the basis for the criteria. Further literature includes model transformation, MDE, and the two modeling languages. Furthermore, documentation of the chosen tool, Oracle BPA Suite, is inspected. The literature review leads to a selection of commonly used business process patterns and an ontological foundation, on which to compare EPC to BPEL. The two sources together compose the criteria, on which to judge the

effectiveness of transformation from EPC models to BPEL specifications. Known issues are collected from literature. The tool's own specifications supplement the known issues. Possible workarounds provided in the literature are used to devise guidelines for modeling.

The known issues and standard business process patterns both act as a starting point to create small EPC diagrams. In addition to the small diagrams, a larger, real-life case is used, composed of as many of the standard patterns as reasonable. The whole set of diagrams is transformed from EPC to BPEL specifications using the Oracle BPA Suite. Limitations and difficulties of the transformation are detected by comparing the resulting specifications to the original models. Confronting the limitations to the desired criteria set previously supplies a conclusion on the effectiveness of the transformation. Combining the limitations with possible workarounds leads to a set of guidelines, which modelers can apply to improve automated transformation.

2.4 Material for Investigation

Depending on the research sub-questions, material to research is selected. Different questions require different sources. The first main research question, which leads to criteria to judge transformations, calls for an investigation of prior research on workflow patterns, as well as ontology. The literature used for workflow patterns is mainly recent conference papers and articles from (IT) journals. One of the main authors in the field, Van Aalst (Aalst, Hofstede, Kiepuszewski, & Barros, 2003) (Aalst, Barros, Hofstede, & Kiepuszewski, 2000), supplements this by an extensive collection of such patterns at <http://www.workflowpatterns.com>. The ontology is based on seminal work by Wand & Weber (1989), who applied a general ontology of Bunge (1977, 1979) to the field of IT. Specifications of BPEL and EPC applied in this literature are obtained from both documentation and related literature. In order to answer the second question and detect the limitations of the Oracle BPA Suite, three sources are relevant. At first, the literature on transformations reveals limitations in general, and the documentation of the Oracle BPA Suite notes those limitations that apply to the specific transformation under investigation. Secondly, to identify more limitations and confirm the ones found before, an experiment is conducted. The results of this experiment serve as basis for the second main research question, to determine the effectiveness of the transformation. Finally, the second question requires literature on MDE and model transformations, and documentation of the Oracle

BPA Suite. For the final main research question, the analysis of the experiment is most important in devising guidelines for modeling. Existing workarounds and guidelines are taken from the documentation of the Oracle BPA Suite and literature on model transformations. This literature functions to review the acceptability of the guidelines. Table 1 lists the material, in combination with the sub-questions that require it.

Table 1: Research material

Sources		Questions
Documentation	Language specifications	1c, 1d, 3a
	BPA Suite	2a, 3a
Experiment	Input/Output/Result	2b
	Analysis	3a, 3b, 3c
Literature	Workflow patterns	1b, 1c, 2b, 2c, 3d
	Model transformation	1d, 2a, 2c, 3c, 3a, 3b, 3c, 3d, 3e
	Ontology	1a, 1c, 1d

2.5 Research Strategy

The research as a whole consist of three parts, as divided by the research objectives and questions. The first two parts are explanatory and predictive in nature (Gregor, 2006). At first, analyzing the content of prior literature and documentation establishes a general theoretical model on transformation. The model consists of criteria based on both ontology and workflow patterns. Basing the research on a stable foundation supports the theoretical soundness and avoids duplicating previous research. The theoretical model explains and predicts the features of model transformation. Secondly, in order to prove the established theory in practice, conducting an experiment produces empirical results. The experiment consists of the construction and transformation of process patterns. The patterns are constructed as EPC diagrams, according to the criteria learned from literature. The Oracle BPA Suite transforms the selected diagrams into BPEL specifications. The results of the experiment include the output specifications and any errors encountered. The results are analyzed to evaluate the effectiveness of the transformation by considering the, severity of, limitations and possible workarounds. Selecting and analyzing the patterns happens in a hierarchical fashion, where the different patterns are first constructed, transformed, and

analyzed individually, before they are analyzed as a whole. Besides these small patterns, a single, larger, composite case is set up to examine the applicability to practice. The diagram for this case follows the same course of investigation as the small pattern diagrams. Finally, the last part provides design and action theory (Gregor, 2006). Prescriptive guidelines are devised, based on the limitations and workarounds found in the previous sections. The larger case serves to validate the guidelines. The guidelines are applied to the case, and then the Oracle BPA Suite transforms the resulting model again. Devising the guidelines serves the purpose of improving the possibilities of arriving at executable BPEL specifications by transforming EPC diagrams.

3 Related research

This chapter provides an overview of existing literature related to this research. It begins with elaborating the criteria, based on which to evaluate the model transformation. The conceptual model in chapter 4 uses these criteria, which originate from ontology (section 3.1) and workflow patterns (section 3.2). For the same chapter, sections 3.3 and 3.4 introduce EPC and BPEL as the two modeling languages. Before dealing with the main subject of this research, section 3.5 introduces Model-Driven Engineering, to provide a background for the whole research. Model transformation is one of the key issues within MDE, and as the focal point of this research, section 3.6 investigates it next. Then, section 3.7 treats the tool that actually transforms from EPC to BPEL, the Oracle BPA Suite. Chapters 5 and 6 use it to obtain empirical results. Finally, section 3.8 draws some conclusions based on the literature. Together, these subjects cover all aspects of the research.

3.1 Ontology

In general, ontology is a branch of metaphysics concerned with the nature of being. Metaphysics, in turn, is the philosophy concerned with abstract concepts, such as the nature of existence or of truth and knowledge (Soanes & Hawker, 2005). It is the discipline concerned with theories of how the "world" may be viewed, conceived, or modeled (Falkenberg, et al., 1998). As part of this research, ontology provides a theoretical foundation, as it studies the way the world, in this case business processes, is viewed, and especially modeled.

The Bunge-Wand-Weber (BWW) model (Wand & Weber, 1990) is selected as one of the two criteria in this research, the other being workflow patterns (described in the next section). This choice is based on extent of available literature on the BWW model, theoretical foundation, and general acceptance in the field of information systems. Due to this acceptance the BWW model was used to evaluate both EPC and BPEL already (Rosemann, Recker, Indulska, & Green, 2006). The prior research leaves the comparison of the two languages according to the BWW model for this research.

3.1.1 Ontology for information systems

The specific ontology applied is the BWW model. It is an adoption of Bunge's ontology (Bunge, 1977, 1979), specifically adapted for information systems by Wand and Weber (Wand & Weber, 1990). In the view of Bunge (1977), ontology is an attempt to categorize, in order to provide a mutual base for understanding. While two parties may agree to disagree, at least they agree on what they disagree. For modeling, this means anything can be expressed by a certain set of concepts, an ontology, and that agreement can be reached on these concepts. For example, in the case of EPC, two business analytics may disagree on the exact form of the business process (do we need an AND-split or an OR-split in this place in the process), but at least the notation (events, functions, arcs, and connectors) is agreed upon.

Wand and Weber (1990) view an information system as a model (abstract artifact) of a real-world system, as viewed by an individual. Information system development, in turn, is the transformation of this individual's view to the artifact (the information system) itself. They aim to specify the quality of the transformation from the individual's view to the artifact. In simpler words: How good the representation is.

Together, the views of Bunge and of Wand and Weber lead to a categorization of agreed upon concepts, based on which to evaluate a representation. The first column of provides a list of these concepts. A representation is evaluated by checking which of these concepts its constructs are able to represent. Any deficiency found during evaluation renders the representation less complete. If the representation is also evaluated based on how it represents the concepts, a verdict can also be given on the representation's clarity. The clarity is reduced by redundancy (more than one construct for a concept), overload (more than one concept for a construct), and excess (a construct that has no related concept).

Besides this representation model, Wand and Weber (1989) originally proposed two other models: a state-tracking model and a good-decomposition model. As this research does not use the two other models, the term BWW model identifies only the representational model in this document.

3.1.2 The choice for the BWW model

The choice for the BWW model is not without argument. The Scandinavian Journal of Information Systems (2006) published a debate on the BWW model. The main criticisms given in this debate (Wyssusek, 2006), include that the BWW model lacks ontological commitment, Bunge's ontology is inappropriately used as a language, conceptual modeling is by definition not captured in ontology, and Wand and Weber missed or ignored several relevant parts. Both Wand and Weber, and other respected researchers provided their view on the critique (Wand & Weber, 2006). They managed to put aside these criticisms, although the debate remains open.

Other ontologies, not BWW or even based on Bunge (for example the ideas of Guizzardi (2005)) or completely different evaluation approaches may be more suitable for analyzing models. Illustrative of this issue is that Wand and Weber (2006) themselves welcome suggestions about other ontologies that may be better in providing a foundation for conceptual modeling. Criteria for an applicable ontology are already described (Gehlert & Esswein, 2007). Still, this research uses the BWW model as one of the two criteria. This choice is based on general acceptance in the field of information systems, theoretical foundation, and extent of available literature on the BWW model. The following paragraphs further elaborate these points.

The general acceptance of the BWW model is demonstrated by the amount of research that empirical validated the model (Bodart, Patel, Sim, & Weber, 2001) (Gemino & Wand, 2005) (Parsons & Cole, 2005) (Burton-Jones & Meso, 2006) and as it is the one generally used in similar cases. For example, it was used to assign semantics to UML (Everman & Wand, 2001), evaluate UML (Opdahl & Henderson-Sellers, 2002), design an approach to evaluate reference models (Fettke & Loos, 2003), analyze the five views of ARIS (Green & Rosemann, 2000), and compare process modeling techniques (including BPEL and EPC) (Rosemann, Recker, Indulska, & Green, 2006).

The reasons for Wand and Weber (1990) to apply Bunge's ontology to information systems point out the theoretical foundation. Firstly, they choose Bunge's ontology because the ontology deals with familiar terms, such as system and event. Secondly, it has a great extent

of formalization and has a standard notation. Lastly, Bunge firmly rooted his ideas on prior ontological research.

More literature focused on deriving new artifacts from the BWW model. In relation to this research, several articles handled the evaluation of conceptual models by using the BWW model, including giving a framework of conceptual modeling (Wand & Weber, 2002), a method to validate conceptual models (Shanks, Tansley, & Weber, 2003), and possibilities and pitfalls for conceptual modeling (Weber, 2003).

3.2 Workflow patterns

A second approach to evaluate and compare modeling languages, as well as their mappings to other languages, is to identify their support for patterns. Patterns in general were first identified in architecture (Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King, & Angel, 1977). The style applied there was copied for use in other areas, including IT (Gamma, Helm, Johnson, & Vlissides, 1995). For this research, the applicable patterns appear in workflow literature, specifically the patterns by Van Aalst et al. (Aalst, Hofstede, Kiepuszewski, & Barros, 2003) (Aalst, Barros, Hofstede, & Kiepuszewski, 2000). They defined workflow patterns for control flow, resources (Russell, Aalst, Hofstede, & Edmond, 2004), data (Russell N. , Hofstede, Edmond, & Aalst, 2005), and exception handling (Russel, Aalst, & Hofstede, 2006). Both dynamic and static patterns were identified. As both EPC and BPEL only deal with static flow of control, only static workflow control patterns (WFCP, further use of patterns refers to these specific patterns, unless explicitly stated otherwise) are the patterns considered as criteria for this research. Figure 2 provides an example of one such patterns.

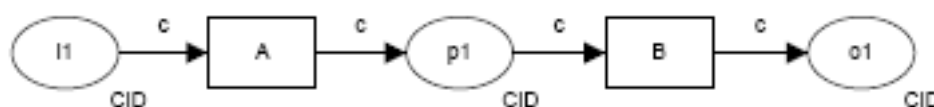


Figure 2: Example pattern using CP nets (from www.workflowpatterns.com)

3.2.1 The source for patterns

The original source for patterns (Aalst, Hofstede, Kiepuszewski, & Barros, 2003) considered 20 patterns. The patterns fall into categories, ranging from simple to complex: basic control flow (patterns 1-5), advanced branching and synchronization (6-10), structural (11-12),

multiple instance (13-16), temporal (17), state-based (18), and cancelation patterns (19-20). In parallel to architectural patterns (Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King, & Angel, 1977), each pattern has a description, synonyms, and examples. The complex cases include the problem (why it is hard to realize) and a possible implementation strategy. As criteria for transformation, only those patterns that EPC can model are applicable as criteria.

The first set of patterns was later complemented. A new article added four advanced patterns (Aalst, Barros, Hofstede, & Kiepuszewski, 2000). Besides workflow control patterns, the previously mentioned data, resource, and operational patterns were defined. Additionally, several new patterns were conceived and recorded. A more recent article introduces many extra patterns, including relations between the patterns (Russell N. , Hofstede, Aalst, & Mulyar, 2006). The complete set of (known) patterns resides at <http://www.workflowpatterns.com> (which Van Aalst and Ter Hofstede maintain).

3.2.2 Evaluation of patterns

The patterns were researched in a number of ways. Formal definitions of the patterns were given in relation to three evaluation strategies languages use: standard, safe (subset of standard based on or-join behavior), and synchronized (avoids arbitrary cycles and thus deadlocks) (Kiepuszewski, Hofstede, & Aalst, 2003). In order to verify and validate the patterns, they were transformed to Colored Petri Nets (Mulyar & Aalst, 2005). Additionally, the patterns were drawn on, to specify a Workflow Pattern Specification Language (Mulyar, Aalst, Hofstede, & Russell, 2006). With the aim to rank the patterns, two studies were conducted on the use of the patterns in practice (Vries & Ommert, 2001) (Vries & Ommert, 2002). The above research improved the knowledge on patterns widely.

3.2.3 Application

The patterns were used for several purposes. Originally, they served to evaluate several workflow management systems (WfMS), based on suitability and expressive power (Aalst, Hofstede, Kiepuszewski, & Barros, 2003). Unfortunately, the system under investigation (the Oracle BPA Suite) was, rightfully (it is not a WfMS), not in the list of evaluated systems. However, another article investigated the Oracle BPEL Process Manager (Mulyar, 2005). shows the result of the evaluation. Furthermore, the patterns were used to evaluate

modeling languages, including UML (Wohed, Aalst, Dumas, Hofstede, & Russell, 2005), BPMN (Wohed, Aalst, Dumas, Hofstede, & Russell, 2006), EPC (Mendling, Neumann, & Nüttgens, 2005), and BPEL (Wohed, Aalst, Dumas, & Hofstede, 2003). The prior research leaves the comparison of the two languages, EPC and BPEL, according to the patterns, for this research.

3.3 Event-driven Process Chain (EPC)

Event-driven Process Chains (EPC) is a business process modeling language, developed to model the control flow of business processes (Keller, Nüttgens, & Scheer, 1992). The basic elements of EPC are functions, events, connectors, and arcs. With the exception of arcs (arrows), Figure 3 shows the constructs.

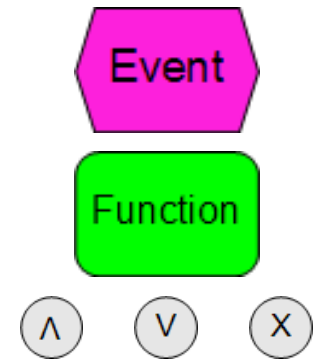


Figure 3: EPC constructs

Functions are the elemental activities in a business process. An internal or external event triggers each function, and each function leads to a new event itself. Events and functions always alternate, similar to places and transitions in Petri nets. Directed arcs, which represent the control flow through the model, attach the functions and events to each other. Connectors split and join the control flow. The EPC language has splits and joins of the AND-, OR-, and XOR-connector types. EPC is a graph-based language, as opposed to block-based, such as BPEL. Mapping the basic elements to Petri nets formalized their syntax and semantics (Aalst, 1999).

3.3.1 History

Originally developed for business processes in ARIS, EPC represents the center of the *ARIS-House of Business Engineering* (Scheer & Schneider, 1992), shown in Figure 4. The four other aspects of the ARIS architecture use different modeling languages to present their information. Coupling the different models of the ARIS-house

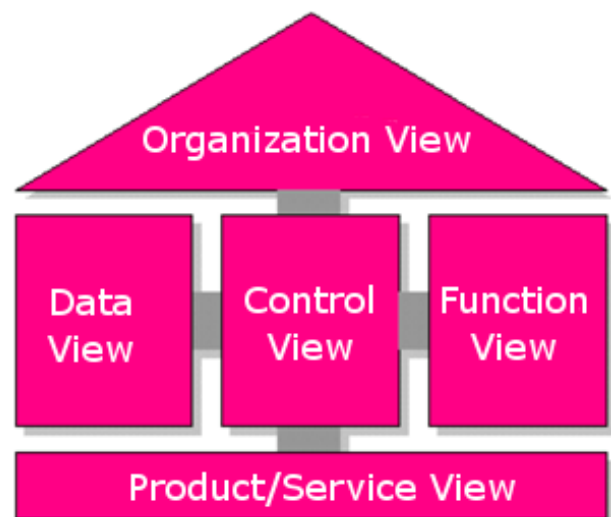


Figure 4: Five views of ARIS

produces a holistic, yet complex, view. EPC is a popular business process modeling language. It established this reputation thanks to its application in both the ARIS toolset and SAP, who

adopted it to model their workflows. EPC is chosen for this research because of the popularity it has among business analysts as a business process modeling language, while it nevertheless has little attention for use in execution of business processes by IT.

3.3.2 (Non-) local semantics

The semantics of the XOR-join are a particular issue for EPC. Local semantics for the XOR-join were proposed first (Rittgen, 1999) (Aalst, 1999). More recently, non-local semantics were proposed (Nüttgens & Rump, 2002) (Kindler, 2006) (Cuntz & Kindler, 2005). Non-local semantics imply that a join locks when one branch reaches the join. If another branch also ends (for example after an AND- or OR-split), then the join does not continue again. Similarly, the OR-join has non-local semantics in EPC too. After an OR-join, the next step only executes when all activated branches have reached the join.

3.3.3 Extensions

Several extensions to EPC were proposed. These proposals include modified EPC (modEPC) (Rittgen, 1999), extended EPC (eEPC) (Scheer, 1994), and YAWL EPC (yEPC) (Mendling, Neumann, & Nüttgens, 2005). Each of these extensions has its own use. The first extension was an attempt to provide formal semantics for EPC. It led to modEPC, which tries to retain the understandability of EPC, while having rigorous formal semantics. The second extension, eEPC, is applied in practice most often. As its name implies, it extends the functionality of EPC for business processes. For example, SAP uses it currently to model the control flow. The last extension came from the area of workflow patterns (see section 3.1). EPC was adapted to YAWL (Yet Another Workflow Language) (Aalst & Hofstede, 2005), so it could represent all workflow patterns. This research considers only classical EPC, as this is what the tool supports.

3.4 Business Process Execution Language (BPEL)

The Business Process Execution Language (BPEL) is a block-structured language. The basic building blocks include activities such as invoke, reply, receive, and assign. Besides activities, the language has a number of constructs to structure the control flow. The simplest ones are pick and flow, respectively an OR-split (choice) and an AND-split (parallelism). Joins are implicit. Scope is the main construct to represent hierarchy and reduce complexity. External

communication occurs through partnerLinks. Table 2 provides examples of some constructs. The official specification (OASIS, 2007) references all constructs.

3.4.1 History









BPEL is an eXtensible Markup Language (XML) based language. As the name implies, it is aimed at making business processes executable. Its first specification (1.0) was built in 2002 as a combination of Microsoft’s execution language XLANG and IBM’s execution language WSFL. Shortly after, others joined in and the second version (1.1) (OASIS, 2003), which is commonly in use now, was submitted to the standards body OASIS and approved as an official, open standard. Currently, the newest version is 2.0 (OASIS, 2007), which was approved in 2007. This version includes many improvements, but tool support still lags behind (OASIS, 2007).

3.4.2 Issues

BPEL was created to accommodate both the internal, executable view of processes, in addition to the external, abstract view (OASIS, 2007). In practice, the abstract view is hardly used (Aalst, Dumas, Hofstede, Russell, Verbeek, & Wohed, 2005). The cause for lack of use lies in the power BPEL has to be executable, which makes it a hard language to learn. The targeted users for the abstract portion, business modelers, prefer to stick to languages in which they are able to easier and better express themselves. These languages include, for example, Business Process Modeling Notation (BPMN) (Object Management Group, 2008) and EPC. The BPEL specification does not impose any graphical representation or design methodology for the processes modeled in it. Consequently, different tool vendors have different representations, possibly leading to confusion. Some tried to use BPMN as the notation for BPEL. However, the attempts drew attention to various incompatibilities between executable specifications and abstract models (Recker & Mendling, 2006).

Formal semantics are not yet complete (Reichert & Rinderle, 2006). BPEL was analyzed (Green & Rosemann, 1999) (Wohed, Aalst, Dumas, & Hofstede, 2003) and compared to

Table 2: BPEL constructs

Code	Diagram
<code><invoke /></code>	
<code><receive /></code>	
<code><reply /></code>	
<code><sequence> </sequence></code>	
<code><switch> </switch></code>	
<code><flow> </flow></code>	
<code><empty /></code>	
<code><scope> </scope></code>	

other languages (Shapiro, 2002) (Söderström, Andersson, Johannesson, Perjons, & Wangler, 2002). Several attempts cover the semantics of a limited number of elements (Reichert, Rinderle, & Dadam, 2004). One of the things that received adequate attention is the control flow (Ouyang, Verbeek, Aalst, Breutel, Dumas, & Hofstede, 2007). As this research focuses on the transformation of the control flow, it suffices that the control flow semantics are formalized.

3.4.3 Web services

For web services, BPEL is the de facto standard as orchestration language. It describes when to call which service. In addition to using standard XML technology, such as XPath, BPEL closely relates to WSDL and SOAP. BPEL uses WSDL to specify message and port types, both for web services it communicates to, and to specify itself as a web service. BPEL makes use of SOAP as a messaging protocol. Together with UDDI, for discovery and publishing, these standards provide a basis for web services and service oriented architecture (SOA).

3.4.4 Versioning

The primer, which accompanies the BPEL version 2.0 specifications (OASIS, 2007), documents the differences between the BPEL versions. In short, the main improvements are improved data access with XPath, extension possibilities, extra scope options, new structure activities, clearer names for some old activities, adaptation and formalization of some proprietary extensions, and clarified usage of abstract processes.

3.5 Model-Driven Engineering (MDE)

Model-Driven Engineering (MDE) is a recent advance in software and system development. The central idea of MDE is that models can be used to make business processes executable. The model itself is executable, instead of being delivered, only as a source for inspiration or requirements, in order for programmers to create the real software (Bézivin, 2004). The goal for MDE is to increase both short- and long-term effectiveness of software and system development considerably. Given the definition of the software development process, as a problem-solving activity that transfers a set of problems into a set of executable solutions (Aksit, 2004); the process is easier when the models of the problem and solution (two different abstraction levels) are already executable. MDE offers a framework to clearly

define methodologies, develop systems at any level of abstraction, and organize testing and validation (Fondement & Silaghi, 2004).

3.5.1 History

Kent (2002) coined the term Model-Driven Engineering (MDE). However, its history starts earlier. Kent's starting point was the Model-Driven Architecture (MDA), which he defined as an instance of MDE. With MDE, he tried to fill the gap left by MDA. Extra dimensions were introduced, mainly borrowed from Aspect Oriented Software Development. MDA is a registered trademark from the Object Management Group (OMG), who established it in 2001 as a base architecture. It is the most widely known form of MDE. MDA aims to achieve architectural separation of concerns in a four-step approach. The first step is to design the business process as a Computation Independent Model (CIM), which describes how the system fits in its domain. The second step devises a Platform Independent Model (PIM), according to the CIM. The PIM specifies the system itself, without advancing to platform specific implementation details. The third step creates a Platform Specific Model (PSM) through, preferably automated, transformation from the PIM. Finally, the PSM is converted to an executable implementation, which happens through, once again preferably automatic, transformation. Together, these four steps complete the route from (business) model to code (Object Management Group, 2003). MDE can be viewed as a continuation from Computer Aided Software Engineering (CASE) tools, which have been used from the eighties onwards.

Table 3: A couple of acronyms related to MDE

MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
MDSD	Model Driven Software Development
MDSE	Model Driven Software Engineering
4GL	Fourth Generation (Programming) Language
SF	Software Factories

3.5.2 Levels of abstraction

MDE provides modelers with the ability to work at different levels of abstraction. It provides this ability by taking the view that, "everything is a model" (adjusted for MDE from the object-oriented "everything is an object") (Bézivin, 2004). This view permits source code, as

Table 4: MDA "3+1"-level Architecture

Level	Name	Description	Example
M3	Meta-meta-model	A language to define a modeling language. Includes defining itself.	MOF
M2	Meta-model	A modeling language.	EPC; BPEL
M1	Model	The model itself.	A process model "BP"
M0	Instance	An instance of the model.	Process instance "BP1234"

well as conceptual models, to be seen as models, in turn allowing model transformation along the entire process chain of software and system development.

Models exist at different levels of abstractions. Two dimensions for abstraction of models exist, one focusing on use of the model and the other on definition of the model. The first dimension is the obvious difference between the source code and any other model of the business process. Here it is clear that any subsequent business process model is more abstract, in order to be easier to understand or be applicable to more situations. The second dimension is the way models are defined. This dimension is presented in a four level architecture (Object Management Group, 2003), as depicted in Table 4. A model (M1) represents the real world (M0). A model is written in a certain modeling language (M2, meta-model). A meta-meta-model (M3) describes the modeling language. Finally, the meta-meta-model also describes itself. This is of utmost importance for model transformations, as the ability to transform between different models depends on the clear definition of the modeling constructs used.

3.5.3 Round-Trip Engineering

As was already noticed in the eighties in the case of CASE tools, models have the tendency to get out-of-sync with the system. The cause for this problem is that the system is developed, without updates to the model, leading to maintenance problems. (Imagine the real life case, where a water pipeline is demolished, because a new road is constructed based on outdated infrastructural charts, which do not include the pipeline yet.) To avoid similar problems, not only transformation of models to (a representation closer to) source code must take place (also known as forward engineering), but also reverse engineering to update (abstract) models based on changes in concrete models, such as source code. This combination of reverse and forward engineering is known as round-trip engineering. It is

more than a sequence of forward and reverse engineering, as the representations must stay synchronized. Information “lost” during the initial action, must be recoverable by the reverse action. Round-trip engineering enables modelers and developers to work on the same artifact, at different levels of abstraction. Round-trip engineering, consequently, facilitates the desirable separation of concerns (Sendall & Küster, 2004).

3.6 Model transformation

In this research, model transformation corresponds to the transfer of a model from one modeling language (meta-model) to another. From the MDE point of view that everything is a model, this also includes the transformation to and from source code (Bézivin, Farcet, Jézéquel, Langlois, & Pollet, 2003). Thus, in contrast to (Alanen, Lilius, Porres, & Truscan, 2003) and sticking to MDA (Object Management Group, 2003), no distinction is made between model-to-model and model-to-code transformation. Transforming toward a more concrete model (for example code) is called forward engineering, while going toward a more abstract model is called reverse engineering. The specific focus is on mapping transformations, where each element of a source model maps to zero, one, or more elements in a target model (Alanen, Lilius, Porres, & Truscan, 2003). As opposed to update transformations, mapping transformations do not alter the source model. A classification of model transformation approaches was proposed (Czarnecki & Helsen, 2003). While transformation of models within a language is possible, this research only considers transformation of models between two languages, namely EPC to BPEL as provided by the Oracle BPA Suite.

Model transformation resides at the heart of MDE (Sendall & Kozaczynski, 2003). Development tools must be able to transform models according to predefined, as well as user-made transformation rules, perhaps even suggesting the preferred kind of model to transform to in a given context. Based on existing tools and languages, criteria for a transformation language that the tools may use were studied (Sendall & Kozaczynski, 2003). For MDA, the language QVT (Query/Views/Transformations) was chosen (Object Management Group, 2008), which consists of three parts: core, relational, and operational. For each of the parts (partially complaint) implementations are available in tools, such as respectively MTF, Medini, and SmartQVT. Good hope for, eventual, success is given by the progress compilers made. While currently compilers are considered to transform code-to-

code, the first code already is an abstraction from machine code. Machine code, in turn, is an abstraction (model) of bits and bytes. The history shows that what was previously deemed hard, impossible, or a typical human activity is now standard an automated process. The model transformation under consideration is regarded as a next step in “model becomes code”.

Much research on model transformation has been conducted already. For sake of brevity, the section discusses only studies on transformation to BPEL and on transformation from EPC. Some transformations that do not fit in these two categories but are still of interest, include the transformation of graph-structured to block-structured diagrams (Mendling, Lassen, & Zdun, 2007) (Mendling, Lassen, & Zdun, 2006), and the reverse engineering of BPEL to EPC (Mendling & Ziemann, 2005).

3.6.1 Transforming to BPEL

Within the first category, the predominant transformation investigated is from BPMN to BPEL. The aim of BPMN stated in its specification, to provide a visualization for BPEL (Object Management Group, 2008), stirs the interest for this transformation. The BPMN specification, as well as other work by one of the authors of the original specification, specified a simple mapping (White, 2005). However, when trying to formalize the transformation several researchers ran into complications (Recker & Mendling, 2006) (Ouyang, Aalst, Dumas, & Hofstede, 2006) (Srikarsemsira & Roongruangsuwan, 2005). The problems found, include the difficulty of mapping a graph structure to a block structure, loss and lack of data, ontological (domain representation) mismatch, and pattern (process representation) mismatch. Workarounds and improvements for some of these problems were formulated (Hauser & Koehler, 2004) (Ouyang, Aalst, Dumas, & Hofstede, 2006) (Ouyang, Dumas, Aalst, & Hofstede, 2006). Other transformations to BPEL studied, include mappings from workflow nets (Aalst & Lassen, 2005), Petri nets (Koschmider & Mevius, 2005) (and reverse (Hinz, Schmidt, & Stahl, 2005)), UML (Gardner, 2003), and even from requirements to BPEL via colored workflow nets (Aalst, Jorgensen, & Lassen, 2005).

3.6.2 Transforming from EPC

In the second category, from EPC transformations, transforming EPC to Petri nets was an important step towards formalization and verification of EPC (Aalst, 1999), as formalization

and verification tools are readily available for Petri nets. A transformation of private, internal EPC diagrams to public, external BPMN diagrams is explained, to improve business collaboration (Hoyer, Bucherer, & Schnabel, 2008). A (partial) transformation to UML is also elaborated, in order to link business processes to object orientation (Nüttgens, Feld, & Zimmermann, 1998).

Table 5: EPC to BPEL transformation assumptions

1	No new, BPEL specific, EPC constructs
2	Technical information has to be added, but control flow is complete
3	A BPEL activity relates to an event + function in EPC
4	EPC events are only significant for (X)OR splits
5	No cyclic EPC models

3.6.3 Transforming from EPC to BPEL

Finally, the transformation that is most interesting to this research, involves both categories: from EPC to BPEL transformation. Two research articles were located on the subject. One is an article by authors of IDS Scheer (Stein & Ivanov, 2007). They provided a ten-step approach, from modeling to implementation and testing, for the transformation of EPC to BPEL for SOA. The approach includes the input of four different human roles. The other article is from the same authors who published the BPEL to EPC transformation mentioned above (Ziemann & Mendling, 2005). They provided a conceptual mapping from EPC to BPEL, based on the five limiting assumptions in Table 5. These assumptions are required, as EPC is more abstract than BPEL.

3.7 Oracle BPA Suite

The Oracle Business Process Analysis (BPA) Suite contains one of the solutions for transformation. For this research, the Oracle BPA Suite is chosen as transformation tool, as it is the only tool accessible providing EPC to BPEL transformation. Besides EPC to BPEL, it is also capable of BPMN to BPEL transformation. The Oracle BPA Suite is based on an OEM agreement for the ARIS Design Platform from IDS Scheer (Oracle, 2006). It combines the capacity of execution and monitoring of Oracle, with the strength of business modeling of IDS Scheer. Oracle offers the Suite as a component of Oracle Fusion Middleware. Besides modeling business processes, it also provides simulation and publishing of the process models.

Table 6: Oracle BPA Suite components

Component	Description
Business Process Architect	Standards-based tool for process modeling. It uses various standards-based notations such as BPMN and EPC.
Business Process Repository Server	Server component for storing and sharing the process repository across multiple users in a collaborative environment
Business Process Simulator	Tool that simulates the process models based on a set of discrete events, to do "what if" analysis.
Business Process Publisher	Publishes process models to a large audience outside the core team designing the process models
Oracle Extensions for SOA	Allows bi-directional integration with the Oracle SOA Suite.

3.7.1 Components

The Oracle BPA Suite contains several components. For this research, the vital component is the Business Process Architect, wherein it is possible to specify business processes. Using on one hand, de facto standards, such as BPMN and BPEL, and, on the other hand, the EPC models from ARIS (Oracle, 2008) (Oracle, n.d.). This component is also able to simulate the business processes. Besides process modeling and simulation, the Business Process Architect provides transformation. This research does not use the other components. Table 6 lists all components (Oracle, 2006).

3.7.2 Integration

Oracle believes that BPEL provides an important building block for Service Oriented Architectures

Table 7: Three step integration

Step 1	Business process modeling
Step 2	Convert to Blueprint
Step 3	Retrieve Blueprint in JDeveloper

(SOA) (Oracle, 2006). Together with the Oracle SOA Suite, the Oracle BPA Suite is claimed to be an integrated environment, enabling round-trip engineering (“closed-loop BPM” in Oracle’s terms) (Oracle, n.d.). To enable this, Oracle attempts to adhere to standards. In the case of this research, the most important standard is BPEL. The integration of the BPA Suite and the SOA Suite happens in the three steps depicted in Table 7. The process is iterative, with the business model as master: the business (analyst) owns the model. In order to support the full extent of Oracle’s BPM Lifecycle (see Figure 5, on the next page), use is made of the existing coupling within the SOA Suite (for example BAM for monitoring, and the BPEL engine for execution) (Oracle, 2006).

Several articles cover an overview and analysis of the combination of the BPA Suite and the SOA Suite (Silver, 2008) (Butler Group, 2007). Another paper treats a comparison between a competing BPMS (Intalio) and the, pre-integration, combination of Oracle SOA Suite and IDS Scheer's SOA Architect (Scheithauer & Wirtz, 2008).

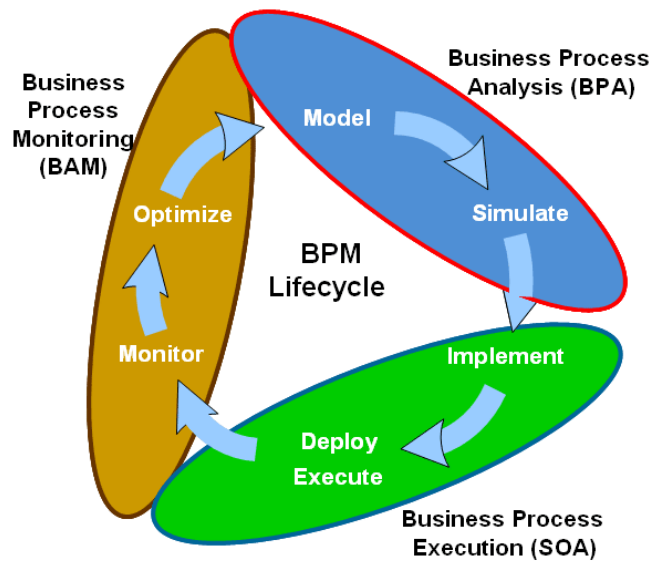


Figure 5: Oracle BPM Lifecycle

3.8 Discussion of treated literature

As can be concluded from the above, model transformations are at the core of MDE. It is possible to evaluate modeling languages based on their ability to represent real world concepts (ontology) and their ability to form certain patterns. A model transformation is theoretically feasible depending on those criteria. The two modeling languages, EPC and BPEL, are compared based on the BWW representational model and 20 workflow control patterns. Issues for model transformation are anticipated, due to mismatches found in the ontology and pattern comparison, as well as due to the graph-to-block structure transformation and different levels of abstraction. The specific transformation for this research is a model-to-model mapping transformation of (abstract) classical EPC to (concrete) BPEL 1.1, without any extensions, such as the other ARIS views. The resulting BPEL specification is considered as (executable) code for integration with the Oracle SOA Suite. The Oracle BPA Suite performs the model transformation, specifically the Business Process Architect component. It creates BPEL output from EPC diagram input. This research does not consider round-trip engineering, so it takes only one-way mapping into account.

4 A framework to evaluate model transformation

Before transforming models from one language (EPC) to another (BPEL), the two languages are compared. Firstly, this comparison happens based on ontology (section 4.1), to see which concepts both languages represent and how good the completeness and clarity of the languages are. Differences between the lists of concepts each language supports indicate limitations for transformation. Secondly, section 4.2 defines a list of (business) process patterns, from which EPC models can be constructed. Prior research on the two languages in combination with such patterns, leads to expectations to which patterns can transform and which cannot. The resulting conceptual model answers main research question 1, as it describes to what extent automated transformation from EPC models to BPEL specifications is possible. Section 4.3 discusses the limitations and issues encountered, together with several possible solutions. As one of these solutions, section 4.4 provides a possible mapping from EPC to BPEL. Finally, section 4.5 summarizes and concludes the conceptual model. The conceptual model serves as a foundation for the remainder of the research: The transformation results in chapter 5 validate it, and it provides the theoretical limitations which the guidelines in chapter 7 attempt to solve.

4.1 Ontological analysis of business process modeling languages

An ontology attempts to capture the reality by precisely defining a set of concepts that specify the domain of the reality (Gruber, 1993). As such, the ontology provides a common vocabulary for users of the domain. For the domain of information systems (IS), a commonly used ontology is the BWW representation model. This ontology was derived from Bunge's work on ontology in general (Bunge, 1977, 1979) and subsequently adapted for information systems (Wand & Weber, 1990). Constructs in business process modeling languages represent the concepts, which the BWW model defines. The leftmost column of in section 4.1.4 provides the (synthesized) concepts, which this research uses. This answers sub-question 1a: It is the ontology, to which business processes must adhere.

4.1.1 Completeness and clarity

An evaluation of modeling languages compares the constructs specified by the language to the concepts defined by the BWW model. This reveals which concepts are missing, and

which constructs are confusing or overkill. Two main, resulting measures are completeness and clarity. The amount of concepts in the ontology, which have no representation in the language, defines the lack of completeness (deficiencies, Figure 6). Clarity is a combination of redundancy, overload, and excess of constructs. Redundancy of constructs means more than one construct in the

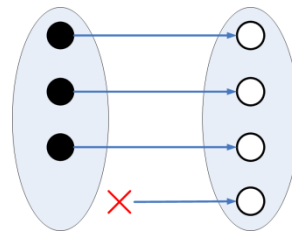


Figure 6: Deficiency

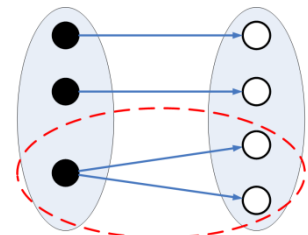


Figure 7: Redundancy

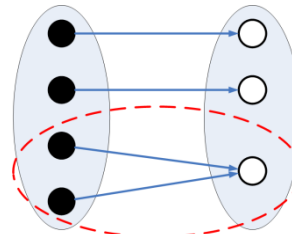


Figure 8: Overload

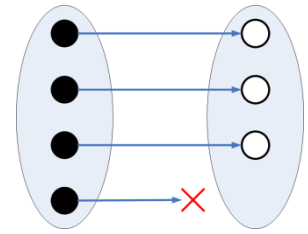


Figure 9: Excess

language represent one concept in the ontology (Figure 7). Overload of constructs indicates that one construct in the language represents more than one concept in the ontology (Figure 8). For example, the EPC function construct represents a property in general, a process, or a transformation. Constructs that do not represent any concept of the ontology cause excess of constructs (Figure 9). For this research, it is not only important to compare the languages to the ontology, but also to see to what extent the results of the comparison match.

4.1.2 BPEL and the BWV model

Prior research was conducted to evaluate BPEL according to the BWV model already (Green, Rosemann, Indulska, & Manning, 2007). BPEL has the ability to represent approximately half (51.7%) of the concepts in the BWV model. While only half does not seem good, only ebXML and BPMN scored better in a comparative study (Rosemann, Recker, Indulska, & Green, 2006). The most eye-catching deficiencies are the lack of the thing concept and the inability to represent most facets of states (which accounts for the largest portion of the deficiencies, 6 out of 14). The BPEL specification contains quite a lot of constructs. While the positive side of this is that the completeness is quite good, the negative side is the lower clarity. Many constructs in BPEL map to a single concept in the BWV model. This redundancy is most obvious for the BWV concepts transformation, 11 BPEL constructs, and event, 4 BPEL constructs. The amount of excess BPEL constructs is quite high too. These constructs are mainly control flow entities. On the other hand, the amount of overloaded constructs is very low. Only the Partners construct is overloaded. This

construct is actually removed from BPEL in version 2.0 (OASIS, 2007). Table 8 and in section 4.1.4 show the results of the comparison from BPEL to the BWW model.

4.1.3 EPC and the BWW model

Prior research also evaluated EPC, according to the BWW model (Green & Rosemann, 2000). EPC can only represent 37.9% of all the BWW concepts. The poor representation of things is especially striking. This deficiency indicates that extension is necessary to model things, for example by using eEPC. The redundancy of EPC is remarkable: no concept in the BWW model is represented by more than one EPC construct. Then again, the overload of constructs is high, as EPC uses the constructs function and event almost solely to represent BWW concepts. The small amount of constructs in EPC directly receives the credit for these two observations. The three excess constructs are the connector entities used for modeling the control flow. Table 8 and list the results of the prior research.

4.1.4 Representational power of EPC versus BPEL

The above elaborations and the overview provided in Table 8 and expose several differences between EPC and BPEL. The completeness of the two languages differ, as it is clear that some constructs that are part of EPC are not part of BPEL, and vice versa. Moreover, the clarity of the languages is at odds, as a number of constructs is overloaded, redundant, and excess. A positive aspect is that not all deficiencies and lack of clarity are insuperable for the case of model transformation. In many cases, the difficulties depend on which language contains the problem.

Table 8: Ontological excess and overload (adapted from (Recker, Rosemann, Indulska, & Green, 2006))

Excess		Overload	
EPC	BPEL 1.1	EPC	BPEL 1.1
AND-connector	Empty	Function	Partners
OR-connector	Message property	Event	
XOR-connector	Message definition		
	Sequence		
	Flow		
	Scope		
42.9%	12.8%	28.6%	2.1%

Table 9: Ontological completeness and redundancy (adapted from (Recker, Rosemann, Indulska, & Green, 2006))

BWW Concept	Completeness		Redundancy	
	EPC	BPEL 1.1	EPC	BPEL 1.1
Thing				
Class		x		1
Kind				
Property	x	x	*	*
State	x	x	1	1
Conceivable State Space				
State Law	x		1	
Lawful State Space				
Stable State	x		1	
Unstable State				
History				
Event	x	x	1	4
Conceivable Event Space				
Lawful Event Space				
External Event	x	x	1	1
Internal Event	x	x	1	3
Well-Defined Event	x	x	1	1
Poorly Defined Event		x		2
Transformation	x	x	1	11
Lawful Transformation	x	x	1	3
Acts On		x		1
Coupling		x		1
System		x		1
System Composition		x		1
System Environment				
System Structure		x		1
Subsystem				
System Decomposition				
Level Structure	x		1	
Degree of Completeness	37.9 %	51.7%		
Degree of Redundancy			0.0%	31.9%

In case of a deficiency in the source language, EPC in this case, no difficulty exists in transformation to the target language, BPEL in this case, as there just is no need to map the missing concept. This situation, for example, arises with the transformation of the BWW thing concept: no such concept is available as construct in EPC, so it cannot be used in a model, and thus does not need to be transformed ever. This condition stands, even if the

target language has the deficiency too. On the other hand, problems do appear if the target language has a deficiency that the source language does not have. For transforming EPC to BPEL, this fact indicates problems for BWW concepts state law and stable state, as they have a representation in EPC (respectively, any split and an end event), and not in BPEL.

Excess of constructs is possibly problematic for model transformation, if the excess occurs in the source language. Excess in the target language leads to no problems, as an excess construct in the target language is never used in the source language. This situation is similar to deficiency in the source language. Even if the source language has an excess construct, this may not be a problem, if it maps to an excess construct of the target language. Of the excess construct in EPC, the AND-, OR-, and XOR-connectors, only the OR-connector is problematic (Kindler, 2006) for transformation from EPC to BPEL. The other two construct map to the excess flow construct of BPEL.

If a construct is overloaded, a dilemma may emerge. If the overload of constructs occurs in the target language, multiple concepts from the BWW model map to the overloaded construct. For transformation, this abstraction is not problematic, but for round-trip engineering, extra data may need to be stored. On the contrary, mapping from one language to the other becomes ambiguous, if in the source language constructs are overloaded. A choice then has to be made, to map the overloaded language construct to one of the corresponding BWW concepts. In some circumstances, the appropriate concept might be clear from the context. However, in any case, this dilemma poses a challenge and the choice for a certain mapping is open to debate. As two of the most important constructs in EPC, function and event, are overloaded, this is a serious obstruction to unambiguous transformation from EPC to BPEL.

Redundancy of constructs may lead to a similar dilemma as construct overload. The problem arises for redundancy only in the opposite situation as for overload, namely if the target language contains redundant constructs and the source language does not have similar overloaded constructs, but has the power to represent them. A choice once again has to be made, to which of the redundant constructs in the target language, the BWW concept has to map. If the right choice is not apparent from the context, then the mapping may be contested. BPEL has overloaded constructs for the BWW concepts event and

transformation, while EPC is capable of representing them in a single fashion. Defining transformation mappings from EPC to BPEL for events and transformations is, thus, prone to ambiguity.

4.2 (Business) Process patterns

This second part of the conceptual model considers process patterns. As EPC is a language to model the control flow of processes (Scheer & Schneider, 1992), use is made of control flow patterns, as opposed to data and resource flow patterns. Process patterns are also known as workflow patterns. In prior research, a list of such patterns was prepared already (Aalst, Hofstede, Kiepuszewski, & Barros, 2003) (Aalst, Barros, Hofstede, & Kiepuszewski, 2000). A full list of patterns is available from <http://www.workflowpatterns.com>. The patterns used in this research are the basic set of control flow patterns, which at the end of the section provides. This answers sub-question 1b: They are the standard patterns commonly used in business processes.

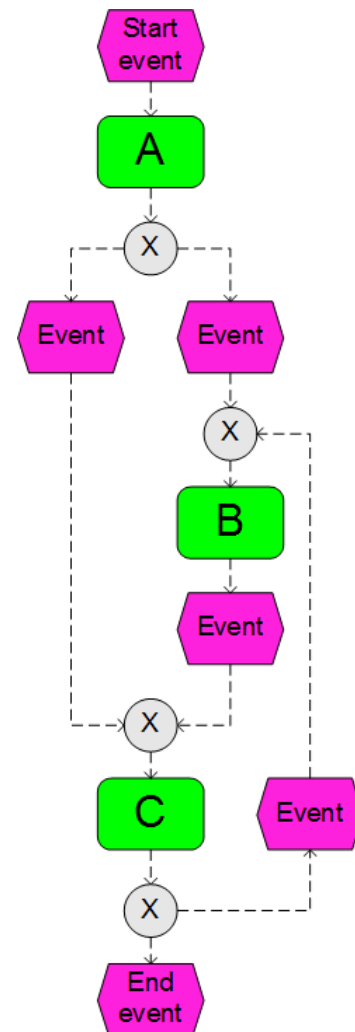


Figure 10: Example of an Arbitrary Cycle in EPC

4.2.1 Patterns in BPEL

An evaluation of BPEL, by investigating which patterns it can represent, was done before, for BPEL in general (Wohed, Aalst, Dumas, & Hofstede, 2003), as well as for the Oracle BPEL PM (Mulyar, 2005). This illustrated that BPEL cannot model some patterns, presumably partly due to the deficiencies discovered when evaluating BPEL in respect to the BWW model. The most notable pattern that BPEL cannot model is Workflow Control Pattern (WFCP) 10: Arbitrary Cycles. Figure 10 shows an example of such a cycle. As opposed to structured cycles (for example while-loops), BPEL has no direct possibility to model unstructured cycles. The block structure of BPEL is the cause for the impossibility. provides an overview of the supported patterns, according to the BPEL specifications, and as found in the Oracle BPEL PM.

Table 10: Capability of constructing workflow patterns

	Patterns	EPC	BPEL 1.1	Oracle BPEL
WFCP 1	Sequence	+	+	+
WFCP 2	Parallel Split	+	+	+
WFCP 3	Synchronization	+	+	+
WFCP 4	Exclusive Choice	+	+	+
WFCP 5	Simple Merge	+	+	+
WFCP 6	Multi Choice	+	+	+
WFCP 7	Synchronizing Merge	+	+	+
WFCP 8	Multi-Merge	-	-	-
WFCP 9	Discriminator	-	-	-
WFCP 10	Arbitrary Cycles	+	-	-
WFCP 11	Implicit Termination	+	+	+
WFCP 12	MI without Synchronization	-	+	+
WFCP 13	MI with a Priori Design Time Knowledge	-	+	+
WFCP 14	MI with a Priori Runtime Knowledge	-	-	+
WFCP 15	MI without a Priori Runtime Knowledge	-	-	+/-
WFCP 16	Deferred Choice	-	+	+
WFCP 17	Interleaved Parallel Routing	-	+/-	-
WFCP 18	Milestone	-	-	+/-
WFCP 19	Cancel Activity	-	+	+/-
WFCP 20	Cancel Case	-	+	+

4.2.2 Patterns in EPC

An evaluation of EPC, according to which patterns are supported, was conducted before too (Mendling, Neumann, & Nüttgens, 2005). EPC directly supports most of the simplest patterns only. Most notably, EPC supports no multiple instantiation (MI) patterns at all. Besides MI, support for cancelation is missing. Due to its free graph structure, arbitrary cycles are possible. shows which patterns EPC supports.

4.2.3 Limitations of transforming patterns from EPC to BPEL

Depending on the supported patterns of both languages, some transformations are troublesome. As with construct deficiencies and excess in respect to (the BWW) ontology,

patterns are hard to transform when the source language has support for them, but the target language lacks such support. Those patterns, which EPC can model directly and BPEL cannot, are also limitations to the effectiveness of the transformation. The only pattern in that category is WFCP 10: Arbitrary Cycles.

4.3 Discussion of encountered issues

In the previous sections, several issues are encountered, which point to challenges for transforming EPC models to BPEL specifications. For each of these issues, a solution must be proposed when trying to implement such a (partially) automated transformation. Table 11 lists all of the encountered issues.

Table 11: Conceptual issues for transformation

Area		Challenge
Ontology	Deficiency	State law
		Stable state
	Excess	OR-connector
	Overload	EPC Event
		EPC Function
	Redundancy	BWW Event
BWW Transformation		
Patterns	WFCP 10	Arbitrary Cycles

4.3.1 Possible solutions

For the encountered issues, many different solutions can be suggested to improve the transformation. Two of the most straightforward solutions in nearly any situation are to forbid the use of the problematic construct or pattern, or to leave that part of the transformation to a human developer. A third solution is to disregard the part. As is clearly illustrated for the case of overload, these solutions are often trivial and unacceptable. Applying the first solution to the overloaded EPC constructs event and function results in a model containing only connectors and arcs, effectively removing all significant meaning. The second solution, alternatively, leaves all actual transformation to the developer, consequently making the automated part trivial. The third solution results in an empty target diagram.

The first solution, forbid the construct or pattern, may be one of the best solutions for the arbitrary cycles issue, though. It is possible to rewrite many of these unstructured cycles to

structured cycles (Zhao, Hauser, Bhattacharya, Bryant, & Cao, 2006). The same applies to the OR-connector, which is possible to rewrite as a combination of XOR- and AND-connectors. Figure 11 shows this for the simplest case of two branches. Otherwise, a solution, such as using event-handlers, can be applied (Ouyang, Aalst, Dumas, & Hofstede, 2006), possibly resulting in unreadable code. A comparable tradeoff, between correctness and readability, often needs to be settled upon. For fully automated transformation, correctness has the highest priority. If a developer still has to work with the resulting code, then readability of that code is also important.

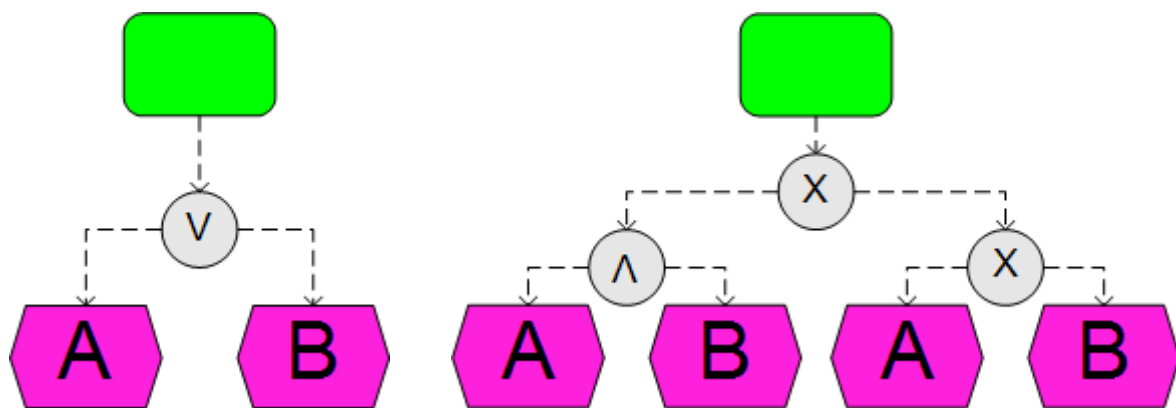


Figure 11: Rewriting an OR-connector to an combination of XOR and AND

The second solution, leave part of the transformation to a human developer, is more appropriate for construct redundancy in the target language. While it is difficult for an automated algorithm to decide which of the redundant constructs to pick, a human developer can judge this by the context of the construct. This could be by the label given to a construct or annotations made by the business analyst.

A third solution is to disregard constructs. For EPC to BPEL transformation, for example, this can be done for many of the EPC event constructs (Ziemann & Mendling, 2005), without loss of information. As with the other proposed solutions, it has to be used very selectively.

As choices have to be made to select a solution, differences may arise between the specifications of EPC and BPEL, and their implementations in tools. The difference is visible for BPEL in case of the Oracle BPEL PM, as the differences between the last two columns of show, BPEL 1.1 versus Oracle BPEL. For EPC, the implementation for the OR-join connector

in the ARIS simulator illustrates it, which uses a time-out to avoid certain issues with the “vicious circle” (Kindler, 2006).

In conclusion, the best solution depends on the situation. The results of the transformation, in Chapter 5, indicate that the Oracle BPA Suite applies all three solutions for different parts of the transformation. The conceptual mapping in section 4.4 provides a slightly different view. It attempts to minimize the input of human developers, so it either provides a mapping, including disregarding a construct, or forbids the pattern. The guidelines in Chapter 7 aim to avoid the need for any of the above solutions. In addition, they guide the modeler and developer to optimize the second solution.

4.3.2 Implementation

Five possible transformation strategies are documented to transform from graph-structured languages, such as EPC, to block-structured languages, such as BPEL (Mendling, Lassen, & Zdun, 2007). They are element-preservation, element-minimization, structure-identification, structure-maximization, and Event-Condition-Action (ECA) rules. Each of these strategies has its own application, due to inherent advantages and disadvantages. The mapping, which section 4.4 provides, uses the structure-maximization strategy. This allows for the most correct and economic-efficient mapping from EPC to BPEL possible, without adding elements to BPEL, as the ECA rules would require. Increased complexity is the drawback, as it requires the use of both the structure-identification and the element-preservation strategies.

4.3.3 Other considerations

Construct excess of control flow entities may be considered a problem of the BWW model, instead of a problem of the modeling language. Similar to this idea, the redundancy of the BWW concept transformation may be a reason to split transformation in disjoint sub-entities. On the other hand, it is argued that neither the control flow entities, nor the subclasses of transformation are required to model the real world (Recker, Rosemann, Indulska, & Green, 2006).

This research only uses the basic control flow patterns for pattern-based evaluation. The full list of 43 control flow patterns may be applied in the same manner. Few extra results are

expected though, as only three patterns are new for the EPC supported patterns. Of those, only the “[generalized AND-join](#)” is a troublesome pattern, in the sense that BPEL does not support it. No problems are expected if data and resource patterns are considered, as classical EPC has no support for data and resources. Thus, these patterns are never modeled. Modeling resources and data requires eEPC.

As the goal of MDE is to arrive at executable code in the end, the requirements for the code must be considered. From the perspective of this research, the resulting BPEL specifications are regarded as the executable code. In order for BPEL specifications to be executable, they have to include several aspects. A data model may be necessary, for example. These conditions are beyond the scope of this research.

4.4 Conceptual mapping from EPC to BPEL

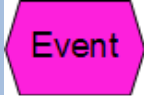


Based on the representational capabilities of the two modeling languages, EPC and BPEL, this section defines a conceptual mapping between them. However, several choices are made to provide a mapping that is as complete as reasonable. The need for these choices arises from the lack of clarity in each of the languages. As another solution can be chosen, the mapping is not normative. The subsequent sections explain the mapping in detail, including the choices and the reasons for picking a specific alternative. The basis of the conceptual mapping is buildup out of the EPC constructs (section 4.4.1), the patterns that can be constructed with EPC (section 4.4.2), and the BWW concepts that EPC is able to represent (section 4.4.3). The resulting conceptual mapping is the answer to sub-question 1d.

A successful conceptual mapping needs to meet several criteria. For this particular mapping, the obvious main criterion is that the mapping needs to be correct. Furthermore, the mapping aims to produce output that is general applicable, accurate, readable, and directly executable. These criteria motivate several of the reasons for picking particular alternatives. Tables 12, 18, and 19 provide the mapping. The text explains each of the individual mappings, starting with the relative straightforward mappings and ending with the complex ones, for which dilemmas arise.

4.4.1 EPC construct based mapping

The three types of EPC constructs, event, connectors, and functions, can all be mapped to BPEL. The patterns and ontology-based mappings enclose this transformation in its totality. Therefore, this section focuses on the issues of the EPC constructs applicable to those mappings.

Table 12: Conceptual mapping of EPC constructs

	EPC	BWW issue	Covered by
EPC events		Overloaded construct	BWW stable state, BWW state, and BWW external event
EPC connectors		Excess constructs	Patterns 2 until 7
EPC functions		Overloaded construct	BWW transformation

Firstly, three BWW concepts overload the EPC construct event. The first one of those is the concept stable state. It is specified as the end event(s) in an EPC diagram. It is disjoint from the other two. The second concept, which an EPC event covers, is state. The EPC events, which denote the state concept, are only those that precede a function. As any event that succeeds a function, either precedes another function or is an end event, the state and stable state concepts cover all occurrences of an EPC event already. However, another concept is also interpreted as an EPC event, namely the concept external event. This is specifically the start event of an EPC diagram. As such, it overlaps with the representation of state. As state is not explicitly mapped (see section 4.4.3), this is not an issue.

Secondly, the excess EPC constructs connectors are useful to represent splitting and joining of the control flow. Their application in patterns 2 until 7 clearly illustrates their usefulness. These patterns also define the mapping of the EPC connectors to BPEL constructs.

Lastly, several concepts of the ontology overload the EPC construct function. All patterns also use this construct. As section 4.4.3 explains, the choice is made to map the EPC construct function to the BPEL construct empty.

4.4.2 Pattern-based mapping

This section describes the conceptual mapping from EPC to BPEL based on the workflow control patterns. These mappings include the expected results of the transformation from EPC to BPEL. Chapter 5 describes the actual transformation, as done by the Oracle BPA Suite.

WFCP 1 - Sequence

Of the pattern transformations, the first one is the simplest one. WFCP 1 – Sequence, which is an alternating sequence of events and functions in EPC, maps directly to the BPEL construct sequence. shows a diagram of the alternating events and functions in EPC.

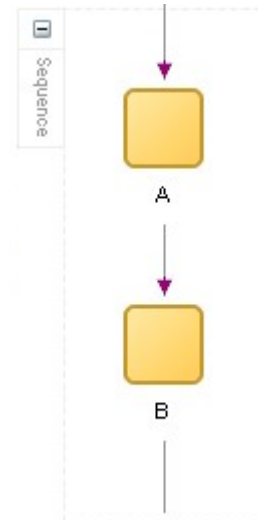
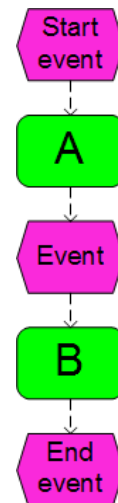


Figure 12: WFCP 1 - Sequence in EPC **Figure 13: WFCP 1 - Sequence in BPEL**

However, the EPC diagram contains the overloaded EPC constructs event and function.

The overloaded constructs require a choice to be made between which concept is represented. A similar selection is required, in order to decide which of the redundant BPEL constructs is chosen for the concepts event and transformation, as the EPC diagram represents them both. Two other concepts that it represents, stable state and state law, are a deficiency for BPEL. This implies that they cannot be transformed to BPEL. No excess EPC constructs exist in the diagram. Section 4.4.3 provides solutions for these ontology-based mappings. This section continues with only the pattern-based mappings.

Table 13 provides the simplest fragment of BPEL code, which represents this pattern. The fragment shows two placeholder activities, `<empty />`, within a sequence construct. shows the corresponding BPEL diagram fragment. The figure does not show the transformation of the start and end events. Those constructs map to a receive construct and an invoke construct, respectively. Section 4.4.3 describes this in more detail, together with the other ontology-based mappings.

Table 13: BPEL code fragment: Sequence

```
<sequence name="Sequence">
  <empty name="A" />
  <empty name="B" />
</sequence>
```

WFCP 2 – Parallel Split & WFCP 3 – Synchronization

Compared to WFCP 1, the other patterns have a more complex representation in EPC and BPEL. They represent splitting and (re-)joining of the control flow. WFCP 2 – Parallel Split has a simple mapping to the BPEL construct flow. Figure 15 shows the expected BPEL diagram. In EPC, however, it is a composition of either an event followed by an AND-connector followed by two or more functions, or a function followed by an AND-connector followed by two or more events. Figure 14 shows the first approach, while Figure 25 (further on) shows the second approach.

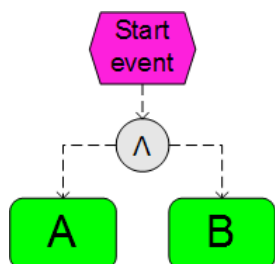


Figure 14: WFCP 2 - Parallel Split in EPC

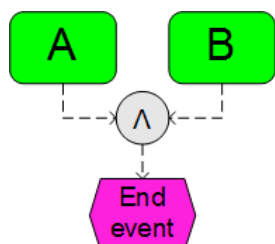


Figure 16: WFCP 3 – Synchronization in EPC

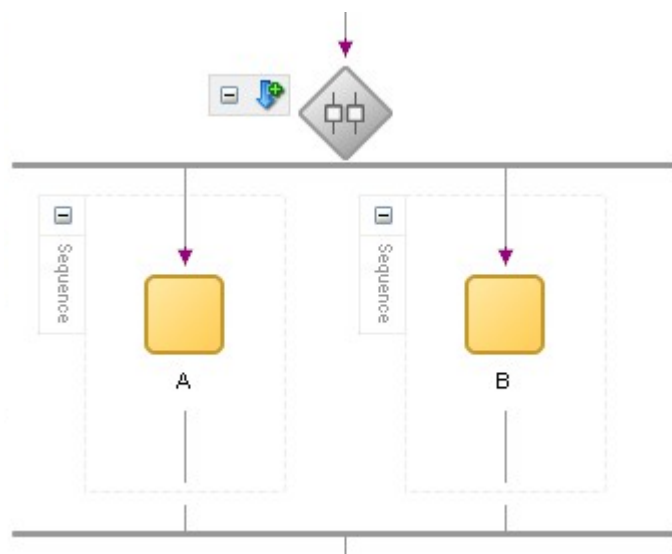


Figure 15: WFCP 2 – Parallel Split & WFCP 3 – Synchronization in BPEL

WFCP 3 – Synchronization always follows WFCP 2 in BPEL, as closing the flow construct represents it implicitly. Figure 16 shows the pattern in EPC. EPC represents the join analogous to the split, except with multiple events or functions before the AND-connector, and a single function or event after the connector.

Table 14 provides the simplest fragment of BPEL code, which represents these patterns. The fragment shows two placeholder activities, each within their own sequence construct. A single flow construct encapsulates each of the sequence constructs. Synchronization, WFCP 3,

Table 14: BPEL code fragment: Flow

```
<flow name="flow">
  <sequence name="Sequence">
    <empty name="A" />
  </sequence>
  <sequence name="Sequence">
    <empty name="B" />
  </sequence>
</flow>
```

implicitly happens when closing the flow construct. Figure 15 shows the corresponding BPEL diagram fragment.

WFCP 4 - Exclusive Choice & WFCP 5 - Simple Merge

WFCP 4 – Exclusive Choice follows a similar path to WFCP 2, but has some notable differences. Figure 17 shows the first two clearly, while the last one is a mapping issue. Firstly, the pattern obviously uses another connector: the XOR-connector. Secondly, the EPC representation allows only a function construct to precede the connector, because events cannot make decisions. Finally, a choice has to be made for the BPEL construct to use, as BPEL offers two of them for this pattern: switch and pick. Pick makes a choice for the thread to take based on an incoming event. Switch, on the other hand, decides based on variables or expressions. The distinction between event- and variable-based choices is not apparent from an EPC diagram. The choice is made to map to the switch construct. While both are correct, it is possible to rewrite any pick construct to a switch construct by letting the incoming event set a variable, and not vice versa. Thus, choosing for switch allows for a general applicable solution. Figure 18 shows the BPEL diagram of this solution. Care has to be taken that the expression conditions, which enable the threads following the switch, are indeed mutually exclusive.

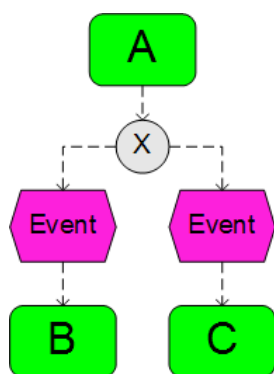


Figure 17: WFCP 4 - Exclusive Choice in EPC

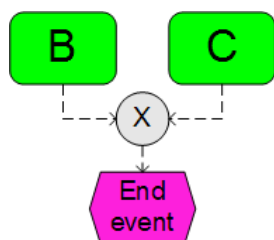


Figure 19: WFCP 5 - Simple Merge in EPC

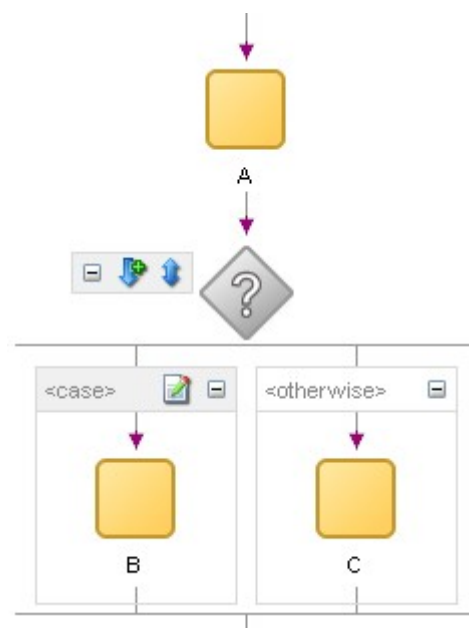


Figure 18: WFCP 4 - Exclusive Choice & WFCP 5 - Simple Merge in BPEL

Figure 19 shows WFCP 5 – Simple Merge. It maps analogous to WFCP 3: it is the closing of the choice construct; switch in this case. As with WFCP 3, it has another representation with two events and a single function.

Table 15 provides the simplest fragment of BPEL code, which represents these patterns. The fragment shows two placeholder activities, each within a branch of the switch construct. The first branch (and in the case of extra branches, those too) is a case construct. It contains a condition on

Table 15: BPEL code fragment: Switch

```

<empty name="A" />
<switch name="Switch">
  <case condition="true()">
    <empty name="B" />
  </case>
  <otherwise>
    <empty name="C" />
  </otherwise>
</switch>

```

which it fires, which is set to true in this example. The second (and in general, the last) branch is the otherwise construct. It fires if none of the other cases evaluates to true. WFCP 5 – Simple Merge implicitly happens by closing the switch. To ensure that the threads never run in parallel, as the pattern requires, the case conditions must be mutually exclusive. Figure 18 shows the corresponding BPEL diagram fragment.

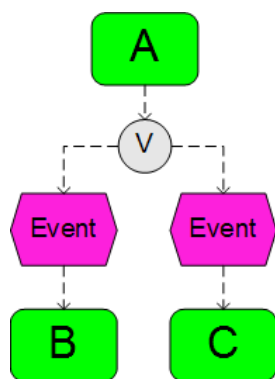


Figure 20: WFCP 6 - Multi Choice in EPC

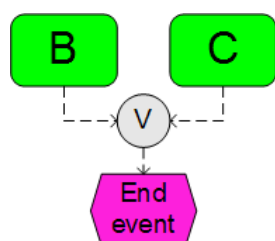


Figure 22: WFCP 7 - Synchronizing Merge in EPC

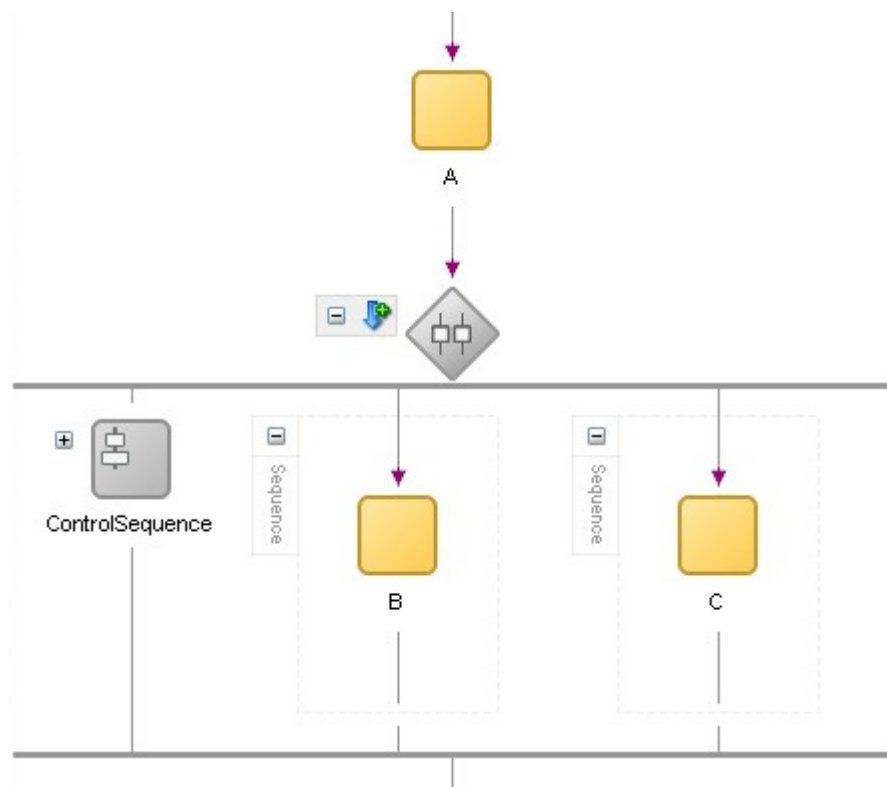


Figure 21: WFCP 6 - Multi Choice & WFCP 7 - Synchronizing Merge in BPEL

WFCP 6 - Multi Choice & WFCP 7 - Synchronizing Merge

WFCP 6 – Multi Choice is less straightforward to map, as it has no direct representation in BPEL. In EPC, it is the same as WFCP 4, except that it uses an OR-connector. Figure 20 shows this. Applying a flow construct, in combination with link constructs, answers the lack of direct representation in BPEL. Table 16 shows the BPEL code for this solution. The flow construct in principle triggers all of its threads, but the link constructs contain expressions, which may limit the activation of the threads. Closing the flow construct from WFCP 6 ensures the synchronization required for WFCP 7 – Synchronizing Merge. The flow construct waits until all threads are either completed or skipped due to the link expressions. The OR-join, which Figure 22 shows, has non-local semantics, as section 3.3.2 explains. This is a correct, yet hard to read, mapping.

The code fragment in Table 16 opens with a flow construct after the initial placeholder activity. This indicates that the three sequences it encapsulates run in parallel. However, due to the use of links, the first sequence (ControlSequence) serves as a control sequence, which enables or disables the other two sequences. A link works by pointing from a source to a target. If the transition condition in the source evaluates to true, then it enables the target sequence. Else, if the transition condition evaluates to false, it disables the target sequence.

Table 16: BPEL code fragment: Flow with links / OR-connector

```
<empty name="A" />
<flow name="Flow">
  <links>
    <link name="linkB" />
    <link name="linkC" />
  </links>
  <sequence name="ControlSequence">
    <source linkName="linkB" transitionCondition="true()" />
    <source linkName="linkC" transitionCondition="false()" />
  </sequence>
  <sequence name="Sequence">
    <target linkName="linkB" />
    <empty name="B" />
  </sequence>
  <sequence name="Sequence">
    <target linkName="linkC" />
    <empty name="C" />
  </sequence>
</flow>
```

For this example, the “false()” expression in linkC causes activity C to be skipped, while the “true()” expression in the other link cause activity B to activate. The example also illustrates the use of the link construct. It requires declaration of the links first. The links then reference each other by name. Figure 21 shows the corresponding BPEL diagram fragment. The links are noticeably not visible in this diagram, created by Oracle JDeveloper. Explicitly adding the links makes the functionality of the links clearer.

WFCP 10 - Arbitrary Cycle

WFCP 10 –Arbitrary Cycles is the most complex pattern to map. In many cases, it has no (straightforward) representation in BPEL at all. For simple (structured) cycles, the pattern maps to the BPEL construct while. In all other cases, the pattern needs modification in EPC. So, generally the pattern is forbidden, as section 4.3.1 proposes.

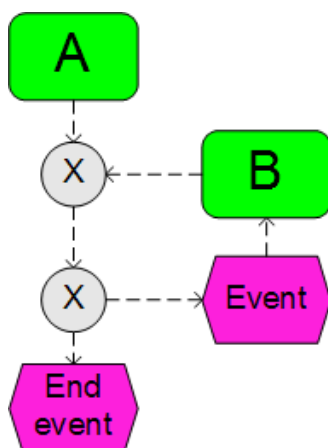


Figure 23: WFCP 10 - Arbitrary Cycle (While-loop) in EPC

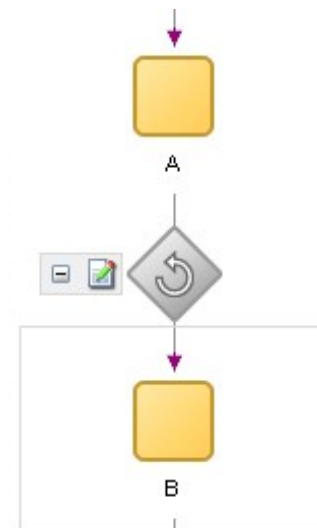


Figure 24: WFCP 10 - Arbitrary Cycle (While-loop) in BPEL

While EPC fully supports WFCP 10 inherently, by its directed graph structure, BPEL is only capable of a limited subset of cycles. The subset is build up out of those cycles, which are possible to represent using while-loops. Due to this, the diagram in is transformable to BPEL. On the other hand, the diagram in Figure 10 is impossible to transform.

Table 17 provides the simplest fragment of BPEL code, which represents WFCP 10. shows the corresponding BPEL diagram fragment. The while construct contains

Table 17: BPEL code fragment: While-loop

```
<empty name="A" />
<while condition="false()">
  <empty name="B" />
</while>
```


activity B. This activity repeats, as long as the condition in the while construct holds true. In this example, the content of the while construct never executes, because the condition is set to “false()”. This is chosen, to avoid the live-lock, which would be caused by setting the expression to “true()”.

WFCP 11 - Implicit Termination

WFCP 11 – Implicit Termination is present in both EPC and BPEL. In EPC as multiple end events, which BPEL represents by closing the main process construct. However, BPEL requires the use of a flow construct. For all other construct an explicit terminate construct is required. Due to this, the pattern is in theory only applicable if the main construct of the BPEL process is a flow construct and not a sequence construct. In practice, nevertheless, any split of control in BPEL is also converged back to one thread, rendering this pattern trivial.

Table 14 (earlier on) provides the simplest fragment of BPEL code, which represents this pattern. Closing the flow construct causes the required implicit termination. When no activities within the flow construct can trigger or are active anymore, the flow terminates. The activities are either skipped or executed. Figure 26 shows the corresponding BPEL diagram fragment.

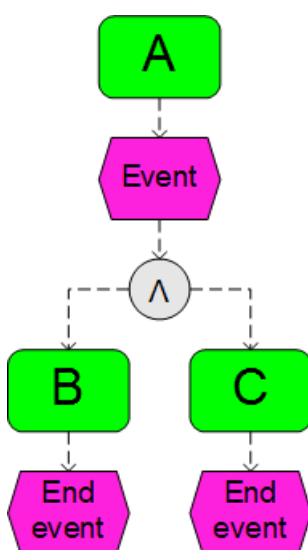


Figure 25: WFCP 11 - Implicit Termination in EPC

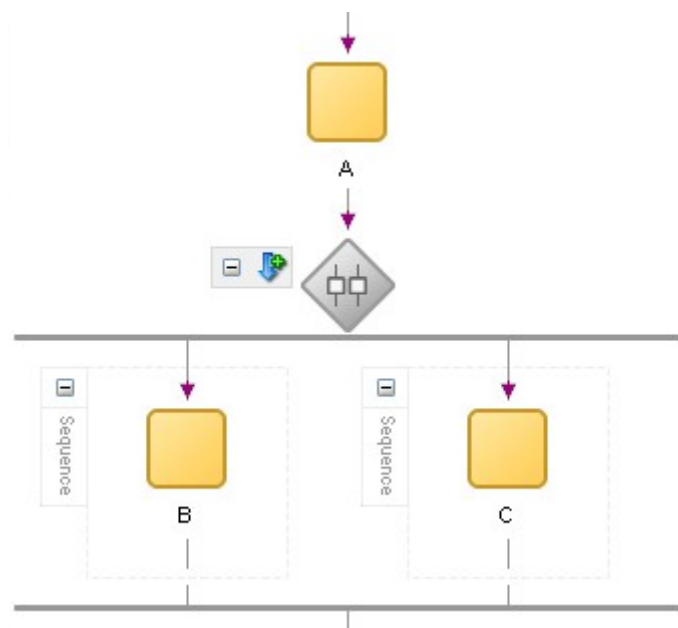













Figure 26: WFCP 11 - Implicit Termination in BPEL

Table 18: Conceptual mapping of patterns

Pattern	EPC	BPEL
WFCP 1 - Sequence	See	
WFCP 2 - Parallel Split		
WFCP 3 - Synchronization		</flow>
WFCP 4 - Exclusive Choice		
WFCP 5 - Simple Merge		</switch>
WFCP 6 - Multi Choice		
WFCP 7 - Synchronizing Merge		</flow>
WFCP 10 - Arbitrary Cycle (While-loop)	See	
WFCP 11 - Implicit Termination	See Figure 25	</process>

4.4.3 Ontology-based mapping

The ontology-based mapping both complements and overlaps the pattern-based mapping. The patterns 2 until 7 cover the BWW concepts state law and lawful transformation. This is illustrated by the fact that EPC represents these two concepts with the same compositions as the patterns: a composition of events or functions, followed by a connector, followed by functions or event. In the same manner, WFCP 1 covers the concept level structure, which consists of a series of events and functions.

The most straightforward BWW concept to map from EPC to BPEL is the external event. In EPC, the start event represents the external event, which corresponds to the BPEL construct receive that triggers the BPEL process. The process receives this trigger from the client. Subsequent receive structures are not possible, as EPC is not capable of explicit extra external interactions.

The stable state, which the end event in EPC represents, has no direct representation in BPEL. However, it is implicitly present when the main construct is closed. This happens only after the client receives the callback. The EPC end event maps to this callback. Depending on

the type of BPEL process, synchronous or asynchronous, the call back takes the form of the BPEL construct reply or invoke, respectively. As EPC diagrams do not represent this difference, a choice is made for the asynchronous variant (invoke). This improves executability for long running processes, as the calling process does not need to wait for the process to finish.

The ability of EPC, to represent the state concept in the form of events, matches to BPEL variables. In order to be generally applicable and executable, this mapping is not done explicitly. The BPEL construct assign could have been used, but EPC diagrams do not specify which, if any, variables are accessed and modified. Thus, as the mapping is not known, it cannot be made explicit.

Finally, the mapping of the BWW concept transformation covers the transformation of all leftover structures. In EPC, event→function→event represents all the other concepts. As EPC events are always part of another mapping (the BWW concepts state, stable state, and external event), only the construct function remains, which is interpreted as the BWW concept transformation. BPEL overloads this concept with, no less than, eleven constructs. Each of these constructs is applicable in its own context. However, it is not possible to obtain the context directly from an EPC diagram. Therefore, any mapping is contestable,

Table 19: Conceptual mapping of BWW concepts

BWW concept	EPC	BPEL
Property	Function	Covered by transformation
State	Event (only triggers of functions)	Variables (not explicitly mapped)
State law	Function(s) → connector → event(s)	Covered by patterns 2 until 7
Stable state	Event (only end)	Final callback to client (reply or invoke construct)
Event	Event → function → event	Covered by transformation
Internal event	Event → function → event	Covered by transformation
External event	Event (only start)	First receive
Well-defined event	Event → function → event	Covered by transformation
Transformation	Function	Empty construct
Lawful transformation	Event → connector → function	Covered by patterns 2 until 7
Level structure	Series of events and functions	Covered by pattern 1

possibly incorrect, and inaccurate. To ensure that the BPEL process is readable and directly executable, a placeholder construct is chosen: the excess construct empty. This is inaccurate, but easy for a developer to change to the real implementation.

4.5 Concluding the conceptual model

The conceptual model, as presented in this chapter, provides the answers to main research question 1 and its sub-questions, as well as the first part of sub-question 2a. The conceptual model starts with discussing the ontology, to which business process modeling languages must adhere (sub-question 1a). The selected ontology is the BWW representation model. Wand and Weber (1990) adopted Bunge's (1977, 1979) ontology, and adapted it for information systems. Table 8 presents the concepts of the BWW model. In combination with , it also shows how EPC and BPEL relate to the ontology (sub-question 1c). Neither EPC nor BPEL perfectly fulfill the properties completeness and clarity (see section 4.1.1), according to the BWW model. BPEL is unable to represent some aspects of state, of which EPC is capable.

The second section of this chapter describes the commonly used business patterns, which business process modeling languages use (sub-question 1b). The selected patterns are the workflow control patterns (Aalst, Hofstede, Kiepuszewski, & Barros, 2003), which the first column of lists. The other columns of that table reveal the capabilities of both languages to represent the patterns (sub-question 1c). BPEL is able to represent more patterns than EPC. However, EPC is able to represent one pattern, WFCP 10 – Arbitrary Cycle, which BPEL is not able to represent.

The conceptual issues for model transformation, supplied in Table 11, partly answer sub-question 2a, by indicating possible problems. The issues arise from lack of completeness and clarity, as well as the inability to represent patterns. Section 4.3 discusses possible solutions and implementations to the issues, as well as handling some extra issues.

In order to answer sub-question 1d (How do the constructs map from EPC to BPEL in theory?), a conceptual mapping from EPC to BPEL is defined in section 4.4. The mapping advises how to accomplish the transformation from EPC diagrams to BPEL specifications. Several choices are made for this mapping, which may be made differently for tools implementing such a mapping.

As a whole, the conceptual model provides a foundation for the rest of the research. The next chapter uses the conceptual mapping and issues. There they form the basis of the expectations for the EPC to BPEL transformations by the Oracle BPA Suite. That chapter also empirically validates the model. Chapter 7 provides guidelines, in order to bypass the limitations, which the conceptual issues indicate.

5 Pattern Transformation Results

This chapter describes the experiment, which is conducted for this research. It starts with the input to the experiment: relatively small EPC diagrams based on the process patterns of the conceptual model. These diagrams are created and transformed to BPEL, by using the Oracle BPA Suite. Subsequently, section 5.2 explains the process of creating and transforming the diagrams. Section 5.3 describes some general aspects of the transformations. Then, several sections follow that explain each EPC diagram, including the source patterns, the transformation results, and an analysis of the transformation results versus the expectations, on a per diagram basis. The conceptual mapping in the previous chapter provides the expectations of the transformations. This chapter gives an account of what is actually implemented. Finally, section 5.5 provides an overall analysis, in order to conclude the findings.

5.1 Input: EPC diagrams

The input for this empirical part of the research consists of several EPC diagrams. These diagrams are constructed based on the conceptual model in the previous chapter. In addition to the single pattern that is theoretically an issue for transformation, WFCP 10 – Arbitrary Cycles, all workflow control patterns, which EPC can model, are input for the transformation. These patterns are WFCP 1 to 7, WFCP 10, and WFCP 11. As is explained below, these patterns also include all aspects, which cause lack of clarity and completeness according to the BWW model. The subsequent sections elaborate and analyze each of the EPC diagrams in isolation.

While representing the process patterns, the EPC diagrams also include the four ontological issues, which became apparent in chapter 4: excess, deficiency, overload, and redundancy. The diagrams of WFCP 6 and 7 include the excess EPC construct OR-connector, as the connector is used for multi choice and synchronizing merge (see section 4.4.2). As the BWW concept stable state maps to the obligatory EPC construct end event, each EPC diagram intrinsically represents the stable state. This concept is a deficiency of BPEL, together with the concept state law. The triple function→connector→event represents a state law in EPC (Green & Rosemann, 2000). It commonly occurs in case of any kind of control flow split, such as the ones in WFCP 2, 4, and 6. The overloaded EPC constructs, event and function,

naturally occur in all EPC diagrams too. The triple event→function→event represents the BWW concept event in EPC (Green & Rosemann, 2000). The triple is the simplest control flow conceivable in EPC. As BPEL has redundant constructs for the concept, it is an issue. This holds even stronger for the other BWW concept that BPEL provides redundant constructs for, transformation. This concept maps to the EPC construct function, which occurs in any EPC diagram. Therefore, the EPC diagrams in this chapter together cover all of the conceptual issues (see Table 11).

5.2 Process: creating and transforming diagrams

The EPC diagrams are created in the Business Process Architect component of the Oracle BPA Suite. The diagrams resemble the pattern definitions as strictly as possible. After creation, the “Share Blueprint with IT” option transforms the diagrams. This option is available in the tool, as part of the SOA functionality (see Figure 27). After clicking this option, the tool asks if it should validate the diagram before transformation. The

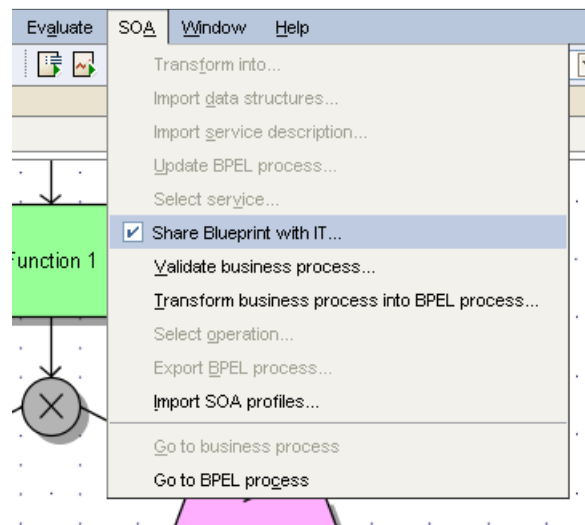


Figure 27: Share Blueprint with IT...

validation of an EPC diagram consists of four structure rules (see Table 20), as well as a set of rules for service-oriented EPC. Independent of whether this validation is done, the tool

Table 20: Structure rules for validation of EPC

Rule	Description
All functions and events have only one incoming and outgoing connection	This rule checks whether all functions and events have a maximum of one incoming or outgoing connection.
Each path must begin and end with an event	This rule checks whether all paths begin and end with an event.
No objects without connections may exist	This rule checks whether a model contains object occurrences without connections to other occurrences. Each object in a model must have one or more predecessors and/or successors.
Number of outgoing or incoming connections at the rule	This rule checks whether at each simple rule there are either exactly one incoming and a minimum of two outgoing connections, or a minimum of two incoming and exactly one outgoing connection.

resumes with the possibility to enter a description or to enter the extended wizard. Clicking OK advances to the transformation. Exceptions may arise at this point. The sections below handle these for those cases where they arise. In case of successfully performed transformation, a confirmation notification appears.

At this point, the transformation finished creating the models for IT. Together, these models comprise the shared Blueprint. The models include at least a BPEL process diagram, several BPEL allocation diagrams, and several UML class diagrams. Figure 28 shows an example of a BPEL process diagram of the "Prepare accounting close" process (see section 6.3), as the Oracle BPA Suite presents it. From top to bottom the diagram includes opening the process, a receive activity, two invoke activities, and closing of the process. Figure 30 on the next page shows the allocation diagram corresponding to the process, including namespace declaration to the left, variables at the bottom, and a partnerLink to the right. Figure 29 shows one of the UML class diagrams, in this case the callback of the process.

It is possible to view all of the (automatically) created models in the Business Process Architect. However, it is more interesting to see how the development environment of IT, Oracle JDeveloper, receives them. In JDeveloper, the shared Blueprint can be loaded as a new BPEL project. JDeveloper then automatically creates the files needed to deploy the process to the Oracle SOA Suite as a web service. These files include at least a WSDL file and XSD schema corresponding to the BPEL process. The developer can choose to observe the BPEL process from a high level of abstraction with little editing possibilities in the Blue Print view, or he can choose to observe the process from the BPEL view with full editing possibilities. From both views, the Source Code view is also accessible. The source

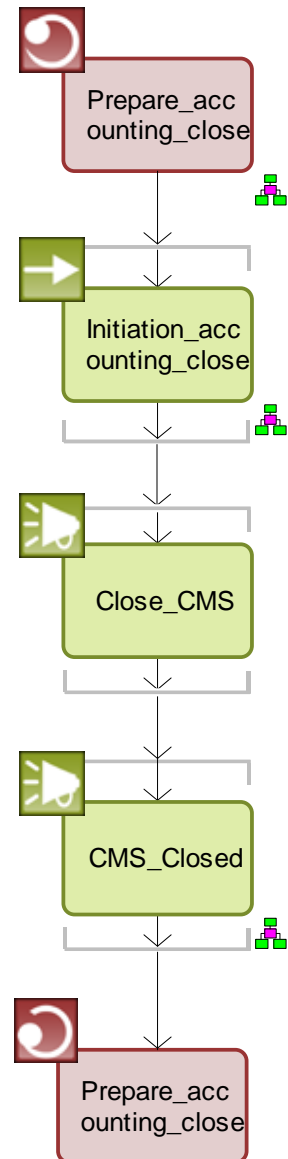


Figure 28: BPEL diagram in the Oracle BPA Suite

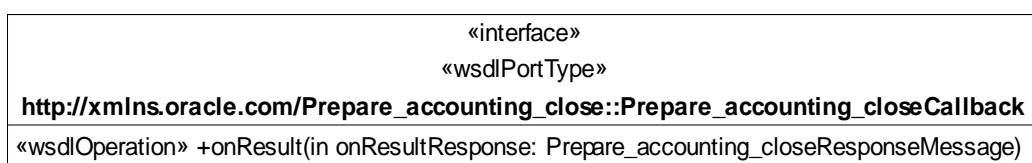


Figure 29: UML diagram of the callback of "Prepare accounting close"

code is compared to results from earlier research on BPEL (Wohed, Aalst, Dumas, & Hofstede, 2003) and the Oracle BPEL PM (Mulyar, 2005), in order to analyze the model transformation as done by the Oracle BPA Suite.

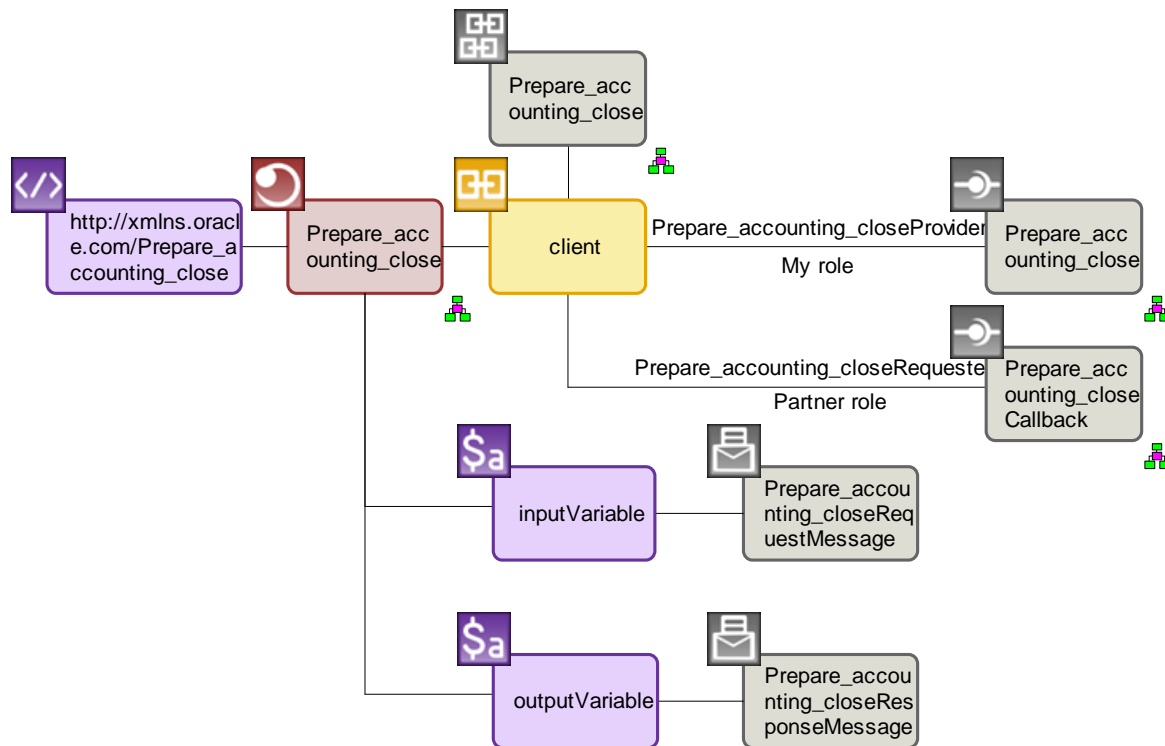


Figure 30: Allocation diagram of the process "Prepare accounting close"

5.3 General BPEL code generation

While the most interesting part of the resulting BPEL code is the fragment covering the pattern itself, the Oracle BPA Suite outputs nearly complete code. In addition to the pattern fragment, BPEL code requires some extra content, which this section explains. The extra content mainly consists of declarations required either by standards or for communication among activities (internal to the process) and among processes (external to the process).

The document starts with the XML declaration, specifying that the document language is XML, and the version of XML it uses. Next, the root of the document (the main and first element, with the name "process") includes the necessary namespace declarations. The next set of declarations is for communication. PartnerLink declarations, as the name implies, define the links to partners (other processes, services, or other entities the process communicates with), and the roles they play. These declarations specify at least the client partnerLink, which is the entity initiating the process and finally receiving the results by callback. The declared variables specify the messages that the process sends to and receives

from the partnerLinks, as well as any variables needed within the process. Finally, the document defines the orchestration logic, which is the part that actually includes the control flow. By default, a sequence construct encloses the whole control flow. Within this sequence, the first step is the trigger from the client partnerLink. A receive construct represents this trigger, which instantiates the process. The part containing the specific pattern fragment follows next. Before closing the enclosing sequence and ending the process, the last construct returns the result to the client partnerLink. Depending on whether the process is synchronous or asynchronous, respectively, a reply construct or an invoke construct fulfills this callback.

The Oracle BPA Suite creates two auxiliary files too. The first one is a WSDL (Web Service Definition Language) file, which is a definition of the process as a web service. The file defines the communication protocol, which other processes or services need to apply to use the process. The second file defines the schema of the variables used. It is an XSD (XML Schema Definition) file, which specifies the type of each variable, for example a string, an integer, or a composite type. These files are required for successful deployment and execution.

5.4 Transformation of Patterns

This section describes the actual transformation of the diagrams. Each section only discusses the fragments relevant to the patterns. Only if unexpected results occur in the extra content, the sections treat that content. This keeps the focus on the conceptual issues. The full output BPEL diagrams are available in Appendix A - Output BPEL diagrams, and the BPEL specifications in Appendix B - Output BPEL code.

5.4.1 Diagram 1: WFCP 1 – Sequence

This EPC diagram (see) represents the pattern WFCP 1 – Sequence: *An activity in a workflow process is enabled after the completion of another activity in the same process.* It is almost the simplest conceivable diagram possible in EPC. The diagram consists of a start event, followed by a sequence of two functions with an intermediary event between them, to finish with the end event. The input for the Oracle BPA Suite is the diagram in .

Results

Besides the obligatory declarations explained in section 5.3, the output BPEL specification consists of a series of scopes encompassed by the main sequence. Figure 31 shows these scopes within the main sequence. Each of the scopes contains a sequence construct, which in turn holds the other activities. Figure 32 shows this activity, within sequence, within scope. The first scope, named Start, includes a receive activity in its sequence, which is connected to the client partnerLink to trigger the process. The last scope, named End, includes an invoke activity, to return the result back to the client, in an asynchronous fashion. The intermediary scopes each include an invoke activity as well, as Figure 32 shows. These, however, do not connect to any partnerLink yet. Oracle JDeveloper shows this with the yellow, triangular error sign. JDeveloper shows less severe issues, such as unassigned variables, with a yellow warning flag.

Inserted between the functional code are several annotations, in the form of BPEL extensions from Oracle. They do not appear in the diagram. The annotations provide extra data, such as creation time, descriptions, and identifiers. This data is assumedly used for identification, and possibly also for synchronization and round-trip engineering with the business diagram.

Analysis

While the results differ from the expectations on many points, the pattern still transforms successfully and manages to fulfill the EPC diagram. Most of the differences arise due to the choices made to resolve the clarity issues of overload and redundancy. Apparently, the choice was made to map the overloaded EPC

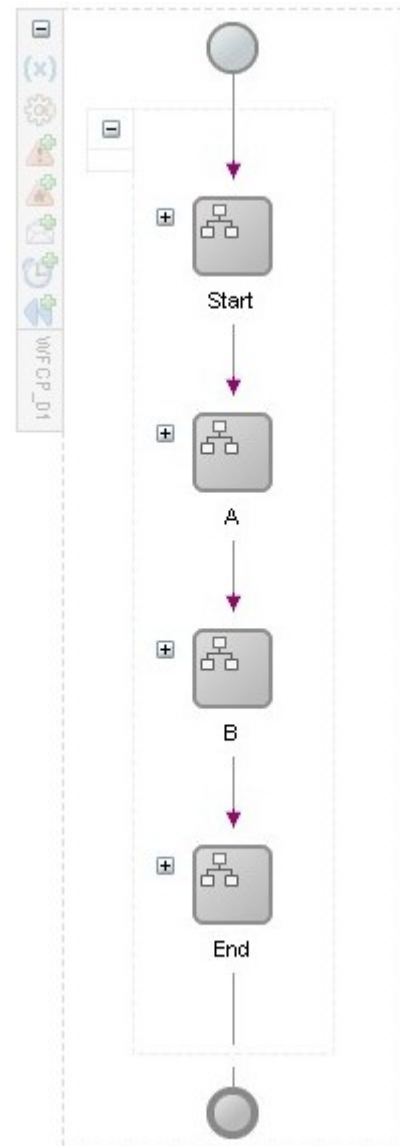


Figure 31: BPEL diagram 1

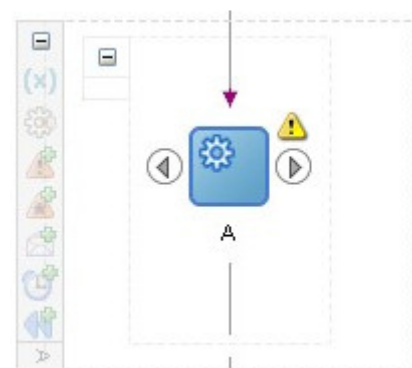


Figure 32: Invoke, within sequence, within scope

construct function to the BWV concept transformation. For this concept, BPEL has redundant constructs. The one chosen is the BPEL scope, including an invoke construct within a sequence, as Figure 32 shows. In a similar fashion, the start and end events are mapped to a scope, with respectively a receive construct and an invoke construct within their sequence. This last invoke indicates the choice for an asynchronous BPEL specification. The intermediary EPC event is dropped altogether. This was expected, as it only functions as a trigger from the first function to the second, which is implicitly present in the BPEL specification. The addition of annotations was not expected, but it does not influence the control flow in any respect. Another point, where the diagram differs from the expected pattern fragment (see Table 13), is the lack of a sequence directly surrounding the two function scopes. It was in place to ensure the sequential execution of the two functions. In the output BPEL specification, the main sequence construct takes care of the sequential processing already. Therefore, it does not change the semantics.

5.4.2 Diagram 2: WFCP 2 – Parallel Split & WFCP 3 - Synchronization

This EPC diagram, which Figure 33 shows, represents two patterns. The first pattern is WFCP 2 – Parallel Split (see Figure 14): *A point in the process, where a single thread of control splits into multiple threads of control, which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.* This pattern is known commonly as the AND-split. The second pattern, which this diagram models, is WFCP 3 – Synchronization (see Figure 16): *A point in the process where multiple parallel branches converge into one single thread of control, thus synchronizing multiple threads. It is an assumption of this pattern that after an incoming branch has been completed, it cannot be completed again while the merge is still waiting for other branches to be completed. Also, it is assumed that the threads to be synchronized belong to the same global process instance.* WFCP 3 is known commonly as the AND-join. Often found at the other end of WFCP 2, it brings multiple threads, created by the AND-split, back to one.

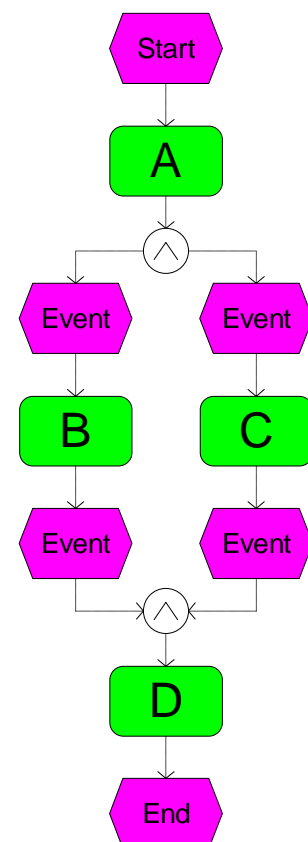


Figure 33: EPC diagram 2

The diagram consists of a start event, followed by function A. The function leads to an AND-connector, which is the main construct for WFCP 2. The connector splits the flow into two separate threads. Each thread includes two events, with a function between them. The final events of each thread lead to another AND-connector, which is the main construct for WFCP 3. From this connector, the process continues as a single flow with a last function D, and finally the end event.

Results

The output BPEL specification consists of two scopes (Start and A), followed by a flow structure named AND_rule, and finally another two scopes (D and End). The contents of the scopes are the same as with the previous diagram (see Figure 32). Therefore, this section only describes the part directly relevant to the patterns: the flow

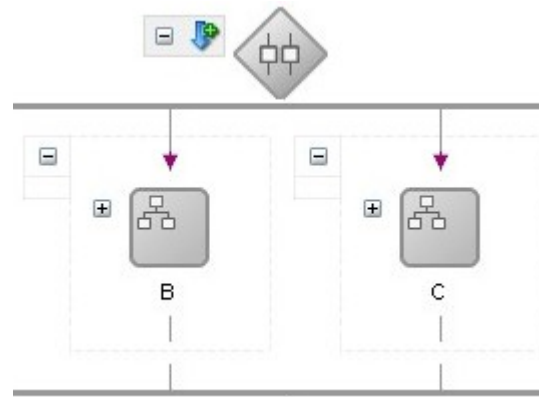


Figure 34: BPEL diagram 2

structure, which Figure 34 shows. The flow structure includes two separate sequences. The sequences represent the two different threads. Each of these sequences contains a scope (B and C). When the flow construct is closed, it passes the control on to the succeeding scope (D).

Analysis

The results are similar to what was expected. The only differences arise due to the aforementioned choice to map an EPC function to the BPEL scope structure. The previous diagram handles most issues that arise during the transformation already. The AND-connectors cause two new issues. The first is the fact that BPEL has no explicit AND-join. This results in BPEL only implicit modeling WFCP 3 – Synchronization. BPEL still supports the pattern though. The second issue is the appearance of the state law concept. BPEL has a deficiency here, and thus is unable to model state law. While BPEL is unable to model the concept, it causes no problems.

5.4.3 Diagram 3: WFCP 4 – Exclusive Choice & WFCP 5 – Simple Merge

Analogous to the previous diagram, this EPC diagram represents two patterns. Figure 35 shows the diagram. The first pattern is WFCP 4 – Exclusive Choice (see Figure 17): *A point in*

the workflow process where, based on a decision or workflow control data, one of several branches is chosen. This pattern is known commonly as the XOR-split. The second pattern, which this diagram models, is WFCP 5 – Simple Merge (see Figure 19): A point in the workflow process where two or more alternative branches come together without synchronization. It is an assumption of this pattern that none of the alternative branches is ever executed in parallel. WFCP 5 is known commonly as the XOR-join. Often found at the other end of WFCP 4, it brings multiple threads, created by the XOR-split, back to one.

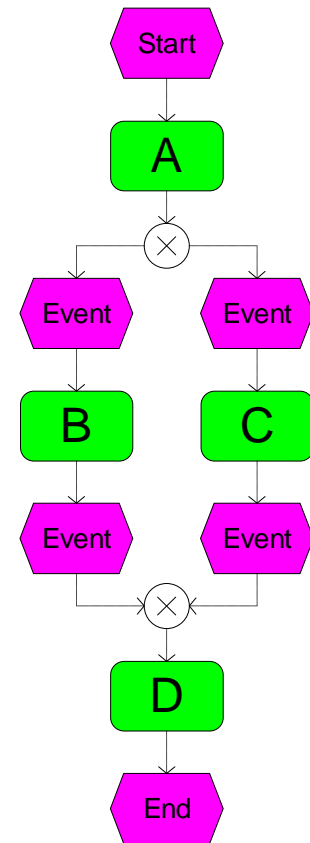


Figure 35: EPC diagram 3

The diagram consists of a start event, followed by function A. The function leads to a XOR-connector, which is the main construct for WFCP 4. As opposed to the previous diagram, an event cannot directly precede the connector, as events cannot make decisions. The connector splits the flow into two separate threads. Each thread includes two events, with a function between them. The final events of each thread lead to another XOR-connector, which is the main construct for WFCP 5. From this connector, the process continues as a single flow with a last function D, and finally the end event.

Results

The output BPEL specification is almost the same as the results from the previous diagram. In comparison with the previous diagram, a switch construct replaces the flow construct.

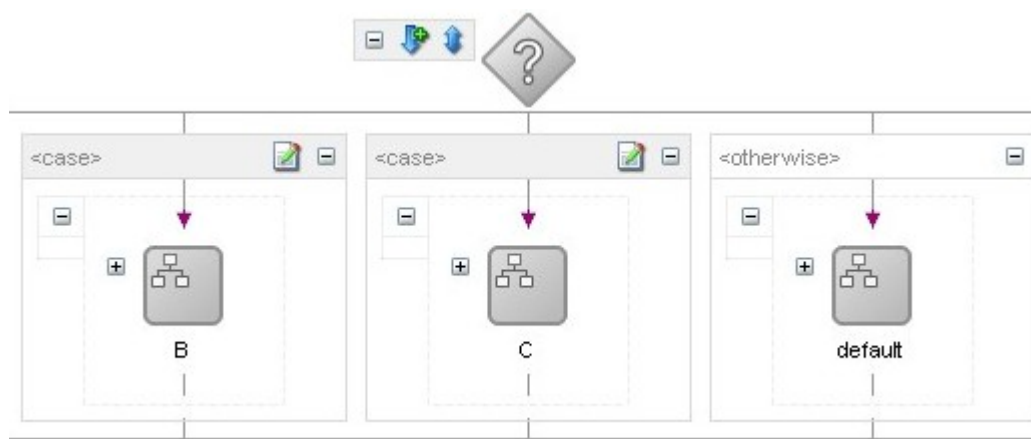


Figure 36: BPEL diagram 3

Figure 36 shows this. The switch construct contains two case constructs and an otherwise construct. Each of these constructs includes a sequence, which in turn contains a scope (B, C, and default). For the two case constructs, the scope contains an invoke construct similar to Figure 32. The scope of the otherwise construct contains an empty construct named default. The conditions for the case constructs are set to the two event names below the XOR-connector. When the switch construct closes, it passes the control on to the succeeding scope (D).

Analysis

The results are quite similar to what was expected. The most notable difference is the addition of a second case construct. It enables the condition for that branch to be set as well, instead of making it fire if the first case condition does not fire. Instead of providing the conditions with an executable expression, the conditions are set to the names of the events following the splitting connector in the EPC diagram. Presumably, this is done to help the developer create the right expressions, or alternatively let the diagram designer set them directly by naming the event. The third thread, triggered by the otherwise construct, does nothing by default. The process, thus, continues if none of the conditions evaluates to true. As was expected, the switch construct is chosen (see section 4.4.2). Setting the conditions to the event names makes this look remarkable, as the BPEL construct pick is the construct meant for choice based on events.

5.4.4 Diagram 4: WFCP 6 – Multi Choice & WFCP 7 – Synchronizing Merge

This EPC diagram, shown in Figure 37, represents two patterns. The first pattern is WFCP 6 – Multiple Choice (see Figure 20): *A point in the workflow process where, based on a decision or workflow control data, a number of branches are chosen.* This pattern is known commonly as the OR-split. The second pattern, which this diagram models, is WFCP 7 – Synchronizing Merge (see Figure 22): *A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken,*

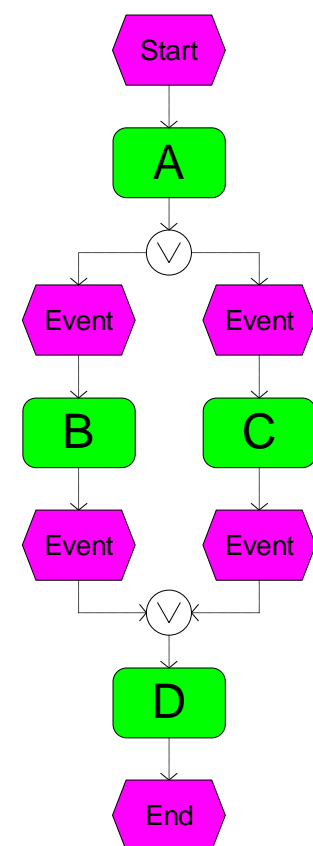


Figure 37: EPC diagram 4

synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should re-converge without synchronization. It is an assumption of this pattern that a branch that has already been activated, cannot be activated again while the merge is still waiting for other branches to complete. WFCP 7 is known commonly as the OR-join. Often found at the other end of WFCP 6, it brings multiple threads, created by the OR-split, back to one.

The diagram consists of a start event, followed by function A. The function leads to an OR-connector, which is the main construct for WFCP 6. The connector splits the flow into two separate threads. Each thread includes two events, with a function between them. The final events of each thread lead to a second OR-connector, which is the main construct for WFCP 7. From this connector, the process continues as a single flow with a last function D, and finally the end event.

Results

The Oracle BPA Suite successfully validates the diagram. However, when attempting the transformation, an error message appears (Figure 38). It explains the transformation is

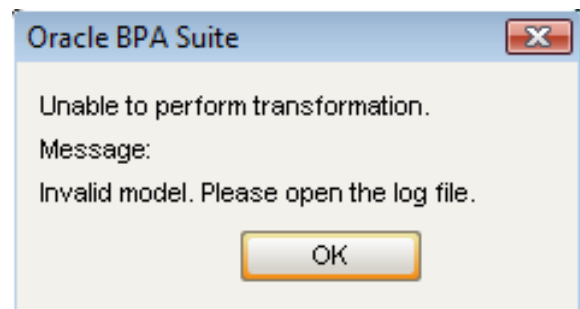


Figure 38: Diagram 4 - Error message

not possible to perform because the model is invalid. Following the reference to the log files, it becomes clear that this version (10.0.3.4) of the Oracle BPA Suite does not support the transformation of the OR-rule (connector).

Analysis

As both EPC and BPEL are able to represent the patterns in this diagram, it was expected that this diagram is transformable. Apparently, the lack of direct BPEL support for the OR-connector avoided implementation. On the other hand, another issue is considered too. The (non-) locality argument, described in section 3.3.2, forces the implementation to make a choice for one of the two solutions. As business modelers make this choice on an individual basis, the created diagrams render incorrectly, when their decision differs from the one implemented. Forbidding (the transformation of) the OR-connector provides a solution, as is described in section 4.3.1. The log message, which notes that this version (10.0.3.4) does not support it, provides the hope that a future version does support it.

5.4.5 Diagram 5: WFCP 11 – Implicit Termination

Figure 39 shows this EPC diagram, which represents WFCP 11 – Implicit Termination (see Figure 25): *A given sub-process should be terminated when there is nothing else to be done. In other words, there are no active activities in the workflow and no other activity can be made active (and at the same time the workflow is not in deadlock). Termination does not require an explicit termination activity.* This pattern occurs in processes with multiple threads that do not converge back to one. If a modeling language does not support the pattern, it is usually possible to rewrite the diagram, in such a manner that the threads come back together, without changing the meaning of the diagram.

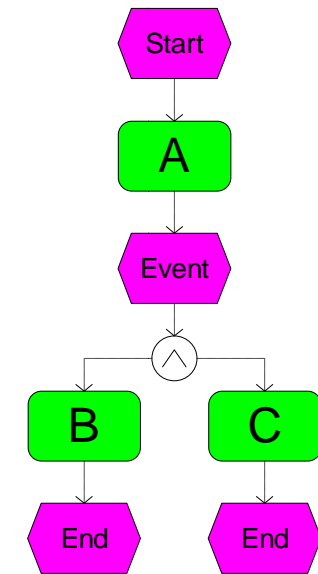


Figure 39: EPC diagram 5

The diagram consists of a start event, followed by function A. The function leads to an AND-connector, which is the cause for multiple threads to occur. The connector splits the flow into separate threads. Each thread includes an event, followed by a function, followed by the end event.

Results

The relevant part of this diagram is the flow construct, which Figure 40 shows. The flow encompasses two separate sequences. Each of these sequences encapsulates two scopes. The first scope (C and B) of each sequence contains an invoke structure. The second scope (End) contains the invoke structure that calls back the client. After that, the sequences, the flow, and the main construct are closed. The client, thus, receives two callbacks: one for each of the end events.

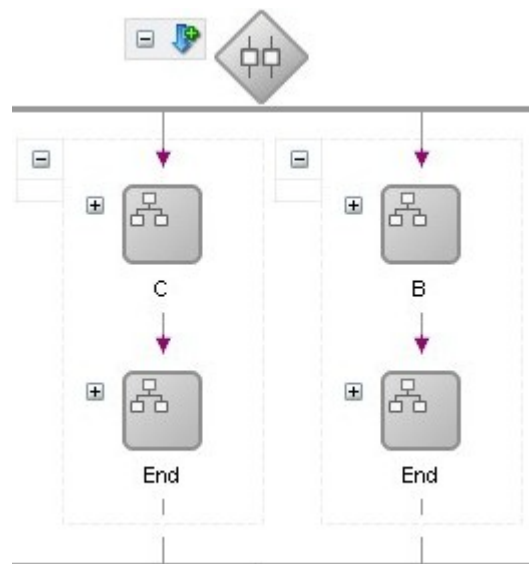


Figure 40: BPEL diagram 5

Analysis

The output BPEL specification differs from the expectations, in that it has two invoke constructs calling back to the client. This construction also represents the pattern

accurately, as the flow construct implicitly terminates after all callbacks complete. The question remains whether the client should receive a callback this often. If not, the business modeler needs to indicate this in the EPC diagram, by explicitly converging separated threads.

5.4.6 Diagrams 6 and 7: WFCP 10 – Arbitrary Cycles

The pattern, which is the input for diagrams 6 and 7, is WFCP 10 – Arbitrary Cycles (see): *A point where a portion of the process (including one or more activities and connectors) needs to be*

visited repeatedly without imposing restrictions on the number, location, and nesting of these points. The fact that this pattern requires (at least) two diagrams illustrates the difficulties, which arise due to it. Repeating a portion of the process is something, of which most modeling languages are capable. However, the requirement that no restrictions are imposed, especially on nesting, leaves any block-structured language unable to represent this pattern. Many different types of cycles exist. Two of these cycles are chosen to illustrate both a possible mapping, and an impossible mapping.

Figure 41 shows the first diagram, which represents a simple repetition of a part of the process. This part is function B and the preceding event. The sub-pattern represented is known commonly as a while-loop. While a condition holds, the part within the while-loop is repeated. The while-loop is build from two XOR-connectors. The first connector simply passes the flow of control to the next connector, either from function A if the process is just started, or from function B after the first iteration. The bottom connector leads to the end event or starts the repeatable part of the process.

The second diagram, which Figure 42 shows, represents a more complex cycle. The combination of XOR-connectors in this case leads to two cycles. The larger cycle, containing functions B and C, enters the smaller cycle. This

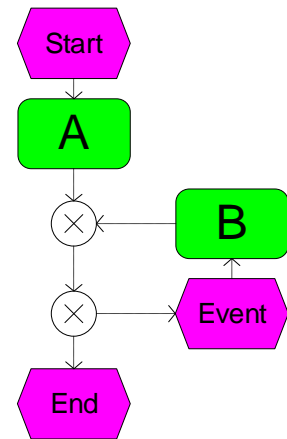


Figure 41: EPC diagram 6

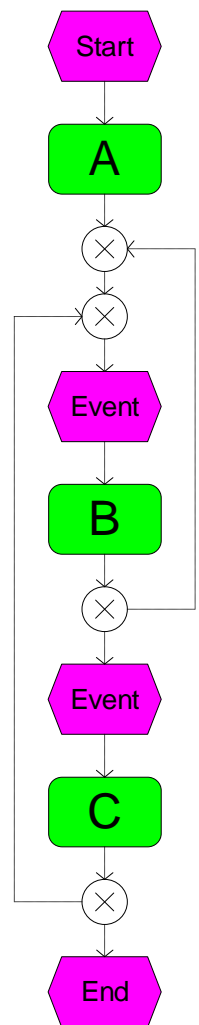


Figure 42: EPC diagram 7

specific diagram is impossible to rebuild using a combination of while-loops or other block-structured constructs. That is possible if the two top XOR-connectors are in the opposite order, as the loops are properly nested then (the loop with function B is totally within the other loop).

Results

Diagram 6 successfully transforms. Figure 43 shows the part relevant to the pattern. After scope A, the while construct includes a single sequence, which encloses scope B. The content of scope B is similar to Figure 32. The name of the while construct is XOR_rule. Its condition is not set. JDeveloper shows this with the triangular, yellow error sign. When the while construct closes, it passes control to the final scope, including the callback to the client.

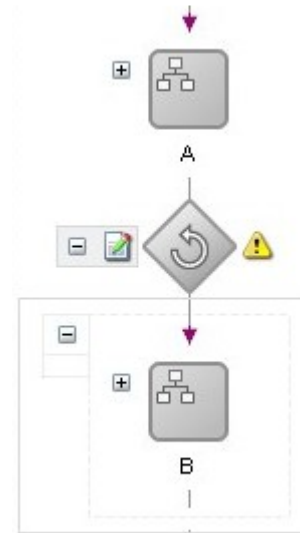


Figure 43: BPEL diagram 6

Diagram 7 already fails at the validation, according to the rules for service-oriented EPC. The validation report informs that the structure is incorrect: *The model contains one or more cycles whose structure is not correct.* When attempting to transform the diagram, the same error appears as with the transformation of diagram 4 (see Figure 38). However, the corresponding log literally confirms the error of the validation report.

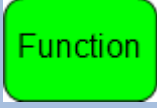
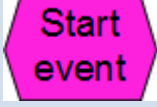
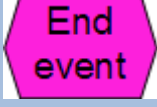

Analysis

The results are similar to what was expected. For diagram 6, apart from scopes representing the functions, the only difference is the missing condition for the while construct. While this lack makes direct execution impossible, it does show up as a warning in Oracle JDeveloper. This warning allows a developer to easily detect it and set an appropriate expression. As expected, diagram 7 fails to transform. Transformation of this diagram is theoretically impossible.

5.5 Conclusion of pattern transformation

In general, the transformation results of the patterns and BWW concepts are similar to what was expected. The most notable difference arises from the choice to model the BWW concept transformation to a BPEL construct scope, with an invoke structure in a sequence. Figure 32 shows this graphically, and Table 21 provides the transformations from each EPC

Table 21: Transformation of BWW concepts, as done by the Oracle BPA Suite

BWW concept	EPC construct	BPEL construct
Transformation		<code><scope> <sequence> <invoke /> </sequence> </scope></code>
External event		<code><scope> <sequence> <receive /> </sequence> </scope></code>
Stable state		<code><scope> <sequence> <invoke /> </sequence> </scope></code>
State		-
-	-	<code><bpelx:annotation /></code>

construct to BPEL code. The choice for this mapping indicates that the tool does not aim to generate executable code directly, but always requires human interference. A developer decides to what partnerLink an invoke construct connects, or if the invoke has to change to another BPEL construct, which also represents the BWW concept transformation. The business analyst can pass descriptions to the developer by using the annotations. When the developer imports the BPEL code in Oracle JDeveloper, the use of the invoke construct without connected partnerLinks results in warnings. These warnings help the developer identify the issues, which he needs to work on. Identification of these issues does not occur with the use of the placeholder construct empty. As the transformation creates a scope construct, the developer can add implementation details within the scope, without bothering the business analyst, which only sees the named scopes in the Blue Print View.

While less interesting from a control flow perspective, the added annotations were not expected. They greatly inflate the BPEL code, by providing extra data for each of the constructs, such as identifiers and creation/update times. As the annotations inflate the code, they reduced readability of the control flow.

Another implementation choice is the use of an invoke construct at the end of the process, which calls back the client. It makes the process asynchronous, which better facilitates long running processes.

6 Validation: a composite case from practice

In order to validate the feasibility of EPC to BPEL transformation in practice, a real life case serves as input for another series of transformations. The case under investigation is the closing of accounts in a large Dutch insurance organization. It involves several departments and many information systems but, as this research focuses on the control flow, these resource and organizational entities are not present in the diagrams.

This chapter starts with an introduction to the case. Then, section 6.2 provides the steps to execute the transformation and the criteria for successfulness and correctness. The subsequent sections elaborate on the transformation of the individual sub-processes. Section 6.9 explains the transformation of the full composite case. Finally, section 6.10 concludes the chapter with an overview of the resulting issues.

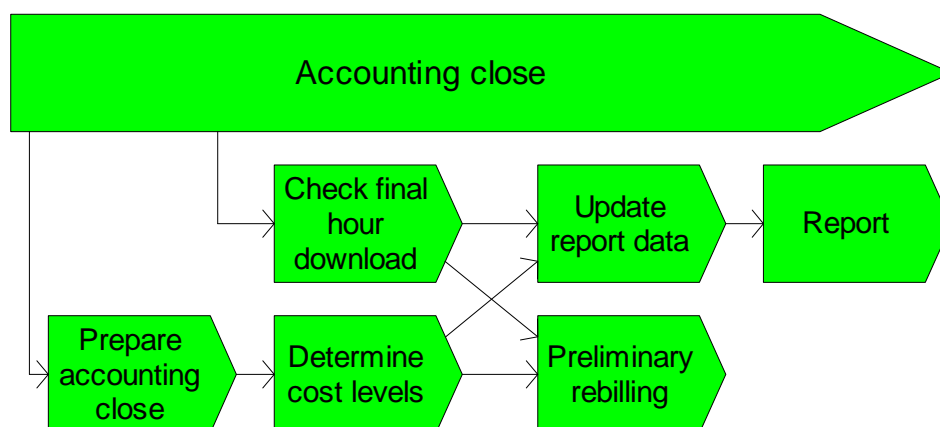


Figure 44: Value Added Chain of Accounting Close

6.1 Case description: “Accounting close”

Within the insurance organization, the process represents the closing of accounts after the end of the month in order to bill customers. Originally, the process was modeled as a composition of several smaller business processes. Figure 44 shows this composition in the form of a Value Added Chain diagram (another of the ARIS diagram types). The main process consists of six sub-processes, which have the order indicated by the diagram. A monthly trigger enables the first sub-process “Check final hour download” (diagram 10). This sub-process runs several checks to ensure the approved hours are posted, the download of the hours is finished, and the hours are complete. In parallel, “Prepare accounting close”

(diagram 8) starts. It closes the Cost Management System (CMS), which is necessary to do the rebilling. When the CMS is closed, “Determine cost levels” (diagram 9) executes. This sub-process calculates and checks the accruals as well as the journals. Then it checks the realization versus the budget. When both “Check final hour download” and “Determine cost levels” are ready, the next two sub-processes start. “Preliminary rebilling” (diagram 11) calculates which costs (hours, direct, product, and overhead) to bill, it checks these costs, and possibly adjusts them. After that, it sends out the preliminary rebilling model to the customer. “Update report data” (diagram 12) loads the hours and costs in the CMS, possibly with some corrections. Then, the final sub-process starts: “Report” (diagram 13). As the name implies, this sub-process creates and sends several reports. These include the invoices, FMR, and updated rebilling model.

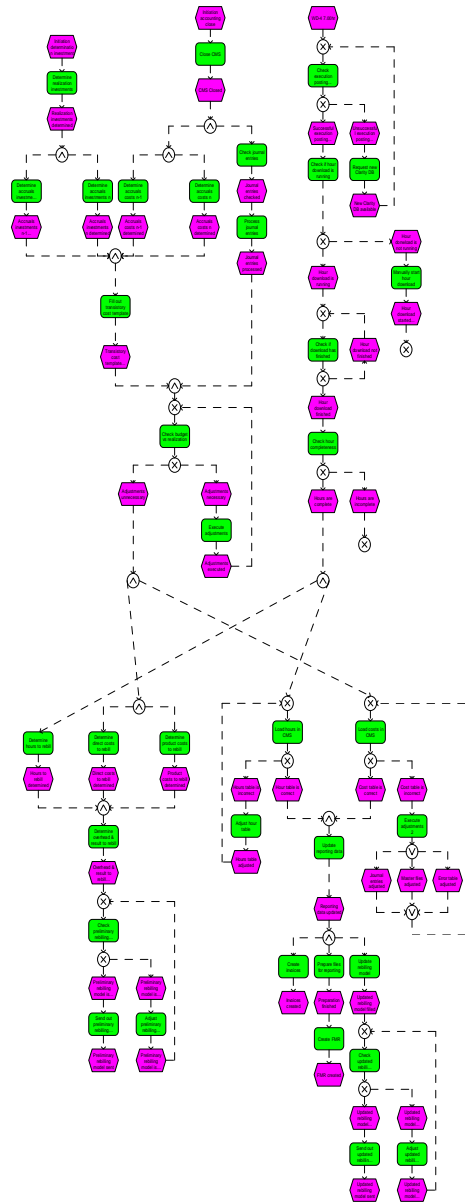


Figure 45: Full case diagram

The individual sub-processes differ in size and complexity. For example, “Prepare accounting close” only consists of a single function, with a start and end event, as well as a process interface to indicate the process that follows it. On the other hand, “Determine cost levels” is far more complex. Its diagram contains multiple start events, incoming and outgoing process interfaces, multiple splits and joins, and even a cycle.

Figure 45 shows the full composite case diagram. It shows all the sub-process combined. Together, the sub-processes cover all patterns and BWW concepts, which EPC is able to represent. Appendix D - Composite case EPC diagrams - provides EPC diagrams of the sub-processes, and Appendix E - EPC diagrams of full case - provides a detailed EPC diagram of the full process.

6.2 Transforming individual sub-processes

Before attempting to transform the full process, the sub-processes are transformed one by one. Some of the diagrams transform successfully without changes, while others need modification first. It is not always possible to preserve the semantics in case of modification. An example of this is the use of the OR-connector in “Update report data”. Replacing the OR-connector with a XOR-connector slightly changes the semantics.

Successful transformation, according to the Oracle BPA Suite (Figure 46), does not always indicate that the control flow in BPEL is the same as in EPC. In the case of “Preliminary billing”, for example, the multiple start events result in exclusion of one branch of the

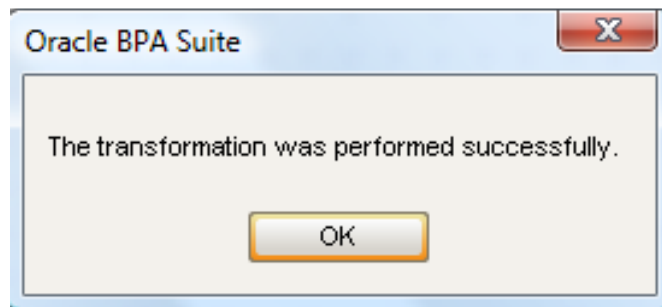


Figure 46: Successful according to the tool

EPC diagram. While the transformation happened successfully, this is clearly not correct. Therefore, *successful* indicates that the Oracle BPA Suite performs the transformation, and *correct* indicates that the resulting BPEL retains the meaning of the input EPC diagram. While successfulness is apparent from the message in Figure 46, correctness is checked by manually evaluating the resulting BPEL.

6.3 Diagram 8: Prepare accounting close

The simplest of the sub-processes, “Prepare accounting close” (Figure 47), transforms successfully and the resulting BPEL code is correct as well. However, the pre-transformation step of validation catches the process interface that refers to the following process, as offending the rule “Each path must begin and end with an event” (see section 5.2). As the Oracle BPA Suite considers the process interface as a variant of function, indeed, the process does not end with an event. Apparently, this is not an issue for transformation, as it is still successful and correct. In order to avoid the offence, however, the process interfaces are stripped from the rest of the diagrams before validation. Semantics do not change due to this modification.

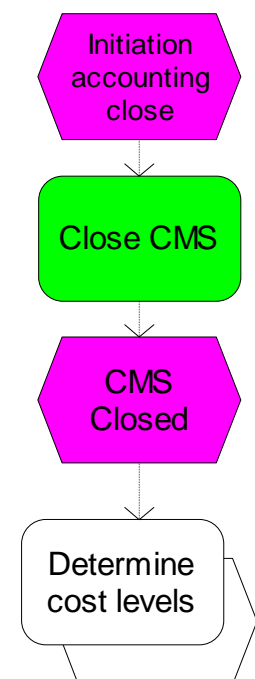


Figure 47: EPC diagram 8

6.4 Diagram 9: Determine cost levels

The sub-process that follows is “Determine cost levels”. With the process interfaces stripped, the diagram successfully transforms to BPEL. A review of the resulting BPEL, nevertheless, reveals that the code is not correct. One of the top branches following from the event “CMS Closed” is missing. Due to that, the BPEL code does not represent several events and functions from the EPC diagram. The rest of the process is correct.

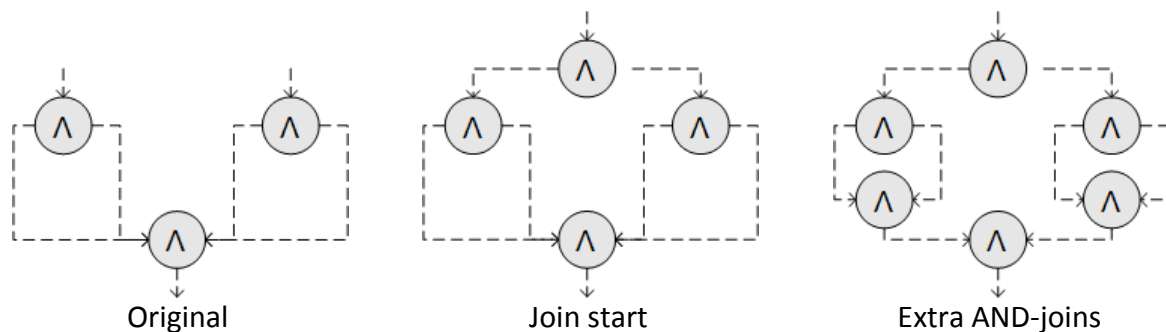


Figure 48: Modifying the structure of diagram 9

The input diagram is modified by adding a single start event, followed by a function and an AND-split, which leads to the original start events. Figure 48 shows the structural changes, without events and functions. Transformation of the modified diagram is successful too. However, the BPEL code now displays peculiar results. The two AND-splits where the accruals are determined do not merge before “Fill out transitory cost template”. This causes that function to be executed twice (at the end of each of the two flows), instead of once. The rest of the process continues correctly. A solution to the above problem is to add another two AND-connectors to join the control flows separately for the costs part and the investments part, before joining those two again. The corresponding diagram transforms successfully, as well as producing correct BPEL output. Furthermore, the semantics are still nearly intact. The only thing, which is no longer explicit, is the fact that part of the process can start when not all start events are triggered. In BPEL, correlation could make this explicit again (OASIS, 2003).

6.5 Diagram 10: Check final hour download

Before the next step in the process occurs, the other top-level branch must complete. This is the sub-process “Check final hour download”, which initiates based on a schedule. With the process interfaces stripped, the diagram transforms successfully and correctly to BPEL,

including a while-loop. One remark can be made on the way the XOR-connector transforms: Just as with the pattern transformation, it contains an extra branch “default”, in case none of the other branches fire.

6.6 Diagram 11: Preliminary rebilling

The next two sub-processes require the previous two sub-processes, as the Value Added Chain diagram in Figure 44 shows. The first is “Preliminary rebilling”. The transformation shows similarity with the transformation of “Determine cost levels”. While the transformation is successful, the resulting BPEL is not correct. Due to the multiple start events, one of the starting branches is not included in the BPEL code. The same solution as applied in the other sub-process, applies here too. It again results in successful and correct transformation, with loss of the fact that part of the process could already start when not all events are triggered yet.

6.7 Diagram 12: Update report data

The second sub-process is “Update report data”. This sub-process contains OR-connectors, due to which the diagram cannot transform (see section 5.4.4). A combination of XOR- and AND-connectors could replace the OR-connectors, while retaining their non-local semantics. As the replacement results in an exponential increase in elements, XOR-connectors simply replace the OR-connectors (see Figure 49). This results in slightly different semantics, but preserves the control flow structure better.

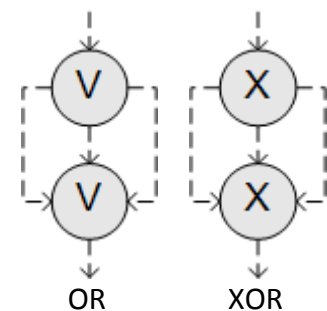


Figure 49: OR to XOR

With this solution and the process interfaces stripped, the diagram successfully transforms to BPEL. The resulting code captures all functions and both the loops, but the two branches do not execute simultaneously, but after each other. This sequential processing is incorrect. Combining the branches at the top with an AND-connector, as was done with previous sub-process, reveals a new obstacle. The Oracle BPA Suite seems to have difficulties when AND-connectors enclose a while-loop. This glitch in the tool results in BPEL code that starts fine, but after the while-loops close, the full sub-process is attached to the loop sequentially. Each one of those full sub-processes individually could have been the correct process. With the purpose to make the composite process transformable as a whole, the two while-loops

are reduced to the single functions (see Figure 50) for loading hours and costs in the CMS. For the business process, this entails that the process does not handle the exceptions, which the loops took care of. This rigorous modification has no further effect on the control flow.

6.8 Diagram 13: Report

The final sub-process is “Report”. The transformation of the corresponding EPC diagram is both successful and correct. This includes the while-loop and implicit termination. No need for modification exists for this sub-process.

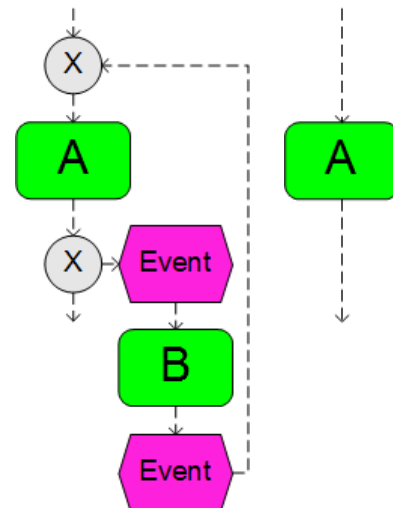


Figure 50: Reduce loop

6.9 Diagram 14: Full composite diagram – Accounting close

Combining the modified, individual sub-processes, which 0 – shows, creates the full composite diagram 15 in Appendix E - EPC diagrams of full case. Combining the sub-processes happens based on the process interfaces, and as the Value Added Chain diagram in Figure 44 shows. Successful transformation requires several modifications, similar to those in the sub-processes. First, adding a single start event and function followed by an AND-connector combines the multiple start events. This connector splits the control flow to the start of both “Prepare accounting close” and “Check final hour download”. Due to this combination of start events, it is only implicit that the two sub-processes can start independently. Connecting the first function of “Determine cost levels” to the final event of its predecessor “Prepare accounting close” achieves the coupling between them. However, combining all events named “CMS Closed”, instead of leaving multiple occurrences of them, improves it further. In the other branch emerging from the combined start event, similar issues to the ones in “Update report data” arise. After the AND-split, the while-loop structure causes errors and incorrect BPEL code. Therefore, this while-loop is removed from the diagram. The open end events do not cause trouble, and neither does the loop toward the end of the sub-process.

At this point in the process, two sub-processes are required for each of the two following sub-processes. Thanks to the combination of the multiple start events of the sub-processes

to a single event, it is possible to combine these with the end events of the predecessors, in a simple manner. That is, by using a combination of two AND-connectors. The first one combines the control flow, which comes from the end events, and the second one splits the control flow to each of the branches, which contain the sub-processes. As with the combination of multiple start events, some of the semantics are lost. Before the new sub-processes may start at all, both preceding sub-processes need to be totally finished. Previously, part of the sub-processes could already start on completion of one of the predecessors.

The sub-process “Preliminary rebilling” and “Update report data” are attached as described above. The first sub-process is also the end of the branch. As the implicit termination pattern works fine, no need exists to combine the end events. The second branch still requires attaching the sub-process “Report”. Combining its first event with the end event of its predecessor accomplishes the attachment in a trivial fashion.

The resulting diagram 14, which is available in Appendix A - Output BPEL diagrams, transforms from EPC to BPEL code both successful and correct. Of course, the EPC diagram is modified in many places.

6.10 Conclusion of case transformation

Both the individual sub-processes and the full composite case diagram require modifications to transform successfully and correctly. The modifications are required, either because part of the transformation is theoretically impossible, or because the Oracle BPA Suite does not (yet) support the structure or construct. The modifications change the semantics in a way that may not be acceptable in practice. On the other hand, the modifications are required to make the process executable. Thus, they can be seen as guidelines for modelers, who need to model an executable process.

Due to the modifications, the case diagram no longer covers the patterns associated with the OR-connectors (WFCP 6 - Multi Choice and WFCP 7 -Synchronizing Merge). As Table 22 shows, the number of entities decreased due to the modifications. The exclusion of

Table 22: Entity statistics

	Original diagram	Modified diagram
Functions	36	35
Events	50	43
Connectors	31	28
Arcs	130	117

several loops is the main cause for the reduction.

As opposed to the transformation of the patterns, the case from practice shows more limitations than theory predicted. Several things that should be possible, according to theory, are not possible for the transformation in the Oracle BPA Suite. This mainly includes diagrams with while-loops. The loops are possible, but often in practice cause unexpected, incorrect results. Multiple start events may lead to similar issues. They are possible, but under certain circumstances, branches are missing. Due to time constraints, it was not investigated what those circumstances are.

While two of the individual sub-processes without modifications transformed successfully and correctly, combining them with the others still results in difficulties. For a modeler this means that small diagrams with couplings to others are usually better than large composite diagrams. Clearly, “The whole is more than the sum of its parts” (Aristotle). This especially holds for the limitations revealed while transforming the EPC diagrams in this case.

7 Guidelines for modeling

The previous chapter presents the results of several EPC to BPEL transformations. Especially the results of the composite case, taken from practice, illustrate the limitations of those transformations. This chapter focuses on ways for modelers to avoid these situations, and how best to deal with them when they, necessarily, occur anyway. Together with an approach to apply this, it is methodological support for EPC to BPEL transformation feasibility. Each limitation receives attention, in the form of guidelines for (business) modelers, who wish to transform their models from EPC to BPEL. Whenever possible, workarounds provide the easiest solution to a problem. If no workaround is available, other forms of guidelines take their place. Depending on the situation, the modeler needs to find the balance between several criteria. The guidelines provide rules based on the criteria required for transformation from EPC to BPEL.

The first section of this chapter specifies criteria for modeling in general, based on a framework. It also defines the specific criteria, which are required for the transformation at hand. The following sections each focus on one of the discovered limitations. These sections provide the actual guidelines. They start with the limitations encountered in both the theory and the experiment, continue with limitations encountered in the composite case, and finish with guidelines that do not directly solve encountered limitations, but improve the feasibility of transformation anyway. The last sections of this chapter provide validation of the guidelines, by applying them to the case from practice, provided in the previous chapter.

7.1 Criteria

The quality of models is a point of discussion, which received (too) little attention throughout the past decades (Moody, 2005). Modelers create models based only on experience and sometimes some form of training. The resulting models often display poor characteristics. They may be hard to read, are ambiguous, are on the wrong level of abstraction, and occasionally even turn out to be incorrect. However, in practice these models may still be useful. This makes it hard to define the quality for models.

This research adopts the perspective that a good model is fit for its purpose. Thus, the quality of a model depends on its usability in the situation at hand. Therefore, it is necessary

to be able to evaluate models from different points of view. The Guidelines of Modeling (GoM) (Becker, Rosemann, & Uthmann, 2000) provide a framework for such evaluation, which is tailored for process models in general. The framework defines six principles: correctness, relevance, economic efficiency, clarity, comparability, and systematic design. The first three of the principles are preconditions for model quality, while the last three are optional properties. Table 23 defines each of the principles.

Table 23: Principles for guidelines (adapted from (Becker, Rosemann, & Uthmann, 2000))

Principle	Description	Priority
Correctness	Consistent and complete against its meta-model (syntactic), and structural and behavioral consistent with the reality it models (semantic).	1
Relevance	No elements that can be removed without loss of meaning.	2
Economic efficiency	Cost/benefit reasoning for the use of structures and constructs.	3
Clarity	Readable, understandable, and useful.	4
Comparability	Follow the same set of guidelines, to be consistent.	5
Systematic design	Well defined integration of different views.	6

For this research, the most applicable principles provide criteria for the (results of the) guidelines. Applicable, in this case, relates to improving the feasibility of the EPC to BPEL transformation, as done by the Oracle BPA Suite. Prioritizing the guidelines results in “perspective-specific guidelines” (Becker, Rosemann, & Uthmann, 2000). These guidelines are the answer to main research question 3.

To improve the feasibility of the transformation, correctness is the single most important principle. Relevance and economic efficiency serve to make a better model, but they are inferior to the correctness of the model. The need for a computer to interpret the model is the direct cause for the importance of the principle of correctness. As opposed to humans, a computer is hardly able to use any context, to correct possible errors in either syntax or semantics. Any of those errors result in the impossibility to transform correctly, or even transform at all. Lack of relevance and lack of economic efficiency, on the other hand, mainly cause the model to become overly complex and overly large, respectively. This might cause difficulties, but mainly due to the modeler or developer losing the overview and, thus, making mistakes. Their mistakes influence the correctness of the model.

The optional principles are merely secondary criteria. The optional principle clarity is helpful to the modeler and the developer, in order to improve communication between them and make validation of the model easier. Improved communication may improve the manual parts of the transformation, but clarity does not directly influence the transformation. Comparability is mainly useful in cases where (reference) models are adapted: the changes should then be easily visible. The same is applicable when attempting round-trip engineering. Comparability makes it easier for the modeler to see what the developer changed, and vice versa. The final optional principle, systematic design, is not applicable to the transformation under investigation, as it focuses on the control flow. Would the transformation incorporate other views, for example the data view of ARIS, then the principle becomes important.

7.2 Limitation 1: Construct excess (OR-connector)

The theory, in section 4.3, already indicates that the OR-connector is an issue. The transformation results of both the small diagrams and the composite case demonstrate that the Oracle BPA Suite does not transform this construct, indeed. Two possible solutions exist, as presented in section 4.3.1. The first solution changes the structure from OR-connectors to a combination of AND-connectors and XOR-connectors. The second solution is to forbid the construct. Each approach has its own advantages and disadvantages. In view of a model, which already correctly represents the real world, only the first solution truly adheres to the principle of correctness. On the other hand, it goes against the principle of economic efficiency, as it creates an exponential amount of constructs. In contrast, the second solution is economically more efficient. In addition to that, it often is also correct. Forbidding the OR-connector is not as problematic as forbidding several other constructs and structures, as a XOR-connector can replace the OR-connector usually. In neither case, the OR-connector is still present in the EPC diagram after applying the guideline, which improves the feasibility of the transformation. Avoiding the OR-connector is also advocated as a guideline for process modeling in general (Mendling, Reijers, & Aalst, 2008) (Gruhn & Laue, 2006), as the OR-connector has ambiguous semantics. The modeler still has the choice to replace the OR-connector with a XOR-connector, or a combination of XOR-connectors and AND-connectors. For most cases, the XOR-connector is the best option. However, for

cases with few branches where correctness is crucial (and the OR-connector was correct), the combination with the AND-connector is possible.

Guideline 1 Do not use the OR-connector.

7.3 Limitation 2: Construct overload and redundancy

The theory in section 4.3 also indicates that the overloaded EPC constructs, function and event, are an issue according to the BWW model. The small diagram transformations show that events transform only in special cases, and that functions transform to scopes, including a BPEL invoke construct enclosed by a sequence (see Figure 32). For both the event and function constructs, the resulting BPEL code is not directly executable. A human developer has to add several implementation details (for example links to the other services), which may include changing the BPEL construct type to which the redundant BWW concept transformation (EPC construct function) is mapped. In order to make the right decision, the developer needs extra information. This can be provided in the form of annotations, attached documentation (Becker, Rosemann, & Uthmann, 2000), but also by using clear, descriptive labels for the EPC constructs. The use of “verb-object” labels is advocated (Mendling, Reijers, & Aalst, 2008). While primary aimed at improving the clarity of a model, it improves the correctness of the final (executable) BPEL code.

Guideline 2 Use clear, descriptive labels.

Developers often also attempt to use the context of a construct to make their decisions. This includes the events surrounding a function or connector. For example, a developer could decide to call a web service synchronous, because the label of the subsequent event is “reply received”, which clearly indicates that the process has to wait for the reply. Besides applying guideline 2, the modeler should pay attention to alternating events and functions in the EPC diagram. Besides being a modeling rule of EPC anyway (thus, the only correct way), it also improves the clarity of the model for the developer.

Guideline 3 Alternate functions and events.

For events after a XOR-connector, the label is even more important. In this case, the Oracle BPA Suite maps the label to the expression conditions of the switch construct (see section 5.4.3). This allows the modeler to specify the conditions directly, if (and only if) he possesses

enough knowledge of the expression language (XPath). If his knowledge of the expression language is not enough, the labels of the events must clearly specify when to take which path. Because the transformation uses the events as expressions, events must always directly follow a XOR-split. In combination with the alternation of events and functions, this leads to the quote, “Events do not decide, functions do”. Ordering the branches can help, but this must be agreed upon with the developer. Otherwise, the ordering has no meaning. It should be kept in mind that the transformation adds a default/otherwise branch to the split (BiZZdesign, 2006).

Guideline 4

Always follow XOR-splits with events.

7.4 Limitation 3: Pattern incompatibility (WFCP 10 - Arbitrary Cycle)

The final limitation, which turned up in both theory and practice, is workflow control pattern 10 – Arbitrary Cycle. EPC supports this pattern, but BPEL does not, which makes transformation hard or even impossible. The only cycles, which BPEL is able to represent, are the while-loop and, its variant, the do-while-loop. None of the other cycles has a direct representation in BPEL, but does have one in EPC. As section 4.3.1 proposes, several solutions exist. Letting a human developer decide, is not a feasible solution in this case, as even they are unable to perform the transformation, if it is impossible. Besides the complexity of transforming Arbitrary Cycles, they are considered an “anti-pattern”, as they are often the cause for multiple instantiation (MI) and deadlocks (Koehler & Vanhatalo, 2007).

Guideline 5

Avoid loops.

Forbidding all cycles would be infeasible too, as many business processes rely on iteration. Often, they require feedback-loops of some kind. Not being able to model those loops would greatly influence the correctness and relevance of the model. Therefore, the simple while-loop is possible. All other loops are forbidden. If such a loop exists in the original EPC diagram anyway, it is often possible to rewrite it to a while-loop in combination with other connectors (Kiepuszewski, 2003). Use clear labels (guideline 2) to communicate the loop-condition to the developer.

Guideline 6

If loops are necessary, then use only while-loops with a single exit.

7.5 Limitation 4: Multiple start events

Diagrams 9 and 11 illustrate difficulties with multiple start events. This structure, which is possible in EPC, is possible in BPEL as well. However, several cases show that the transformation does not always happen correctly. The diagram transforms successfully, but not correctly, as only a single branch of the structure exists in the resulting BPEL code. A general solution for this is to merge the start events. In the case where all start events are required, adding an AND-connector before the branches achieves the merge. In case the process starts and is able to end when only one start event fires, a XOR-connector before the branches realizes it. The type of connector must match a connector further in the diagram. The connectors require at least a new, artificial start event to precede them, as an EPC diagram cannot start with a connector. Literature supports the use of only one start event, for the need of “structured” diagrams (Kiepuszewski, 2003), avoiding the anti-pattern “dangling inputs” (Koehler & Vanhatalo, 2007), reducing error probability (Mending, Reijers, & Aalst, 2008), and because a process with multiple triggers often indicates multiple processes (BiZZdesign, 2006).

Guideline 7 Avoid multiple start events.

7.6 Limitation 5: Degree of connectors

Diagrams 9 and 11 require the addition of several AND-connectors. Besides the multiple start events, the large amount of incoming arcs at one of the existing AND-connectors causes the need. It illustrates that the degree of incoming arcs at a join must match the degree of

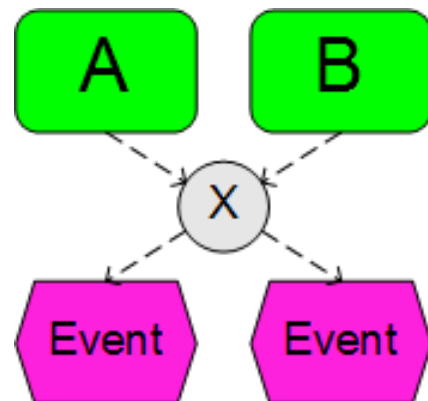


Figure 51: Erroneous connector

of outgoing arcs at the corresponding split. The validation of the structure fragment fails due to the rule “Process parallel flows, inclusive decision paths and exclusive decision paths are well-formed” (see Table 20). Still, it is possible to transform the diagram successfully. However, the resulting BPEL code is incorrect. The same happens when disobeying the rule “Number of outgoing or incoming connections at the rule”. This rule limits connectors to being either a split or a join, not both: A connector either has a single incoming and multiple outgoing arcs, or a single outgoing and multiple incoming arcs. For example, the diagram fragment in Figure 51 is erroneous. Validation catches the error, but it still transforms successfully, yet with unexpected, incorrect resulting code. In general, minimizing the

routing paths per element solves these problems (Mendling, Reijers, & Aalst, 2008). Applying this guideline usually violates the principles of economic efficiency and relevance, but that effect is inferior to correctness. As a side effect, the guideline also improves clarity.

Guideline 8

Minimize the amount of arcs attached to a construct.

7.7 Limitation 6: Multiple end events

As with multiple start events, multiple end events are possible in both EPC and BPEL. Moreover, analogous to multiple start events, multiple end events sometimes transform correctly and sometimes do not. The implicit termination of WFCP 11 and diagram 13 illustrate the case where the transformation goes well. The multiple callbacks to the client are questionable already, though. At first, the transformation of the multiple end events in the sub-process depicted in diagram 10 seems to be fine too. When coupled to the rest of the process in diagram 14, however, an ambiguity in the semantics presents itself. The original EPC diagram already contains the ambiguity, as it is not clear what happens when taking the “dead-end” branches. Questions arise, such as: Should the process stop? With or without throwing an error? What does the client receive as callback? Does the client receive a callback at all? Does the process require human interference?

For the process in diagram 10, the branch with event “Hours are incomplete” probably indicates that the process should stop, and requires human interference. The other dead-end branch, ending with the event “Hours download started manually”, rather seems to indicate a modeling error. The branch should probably reconnect to the main flow, at the event “Hours download is running”, which is the logical consequence of starting the download. This last example demonstrates how style checking can improve models (Gruhn & Laue, 2006).

Semantics of dead-end branches are not clear in general (Dongen, Jansen-Vullers, Verbeek, & Aalst, 2007). Several possibilities exist to avoid the problem. The first solution is to communicate clearly with the developers what is supposed to happen. Modelers can improve communication by providing clear labels (guideline 2) and linking to documentation. In cases of explicit termination or throwing errors, they can agree with the developers to use certain conventions to indicate this. For example, the modelers could append an event labeled “explicit termination”, or “throw error” to dead-end branches. For

automated transformation, improving communication is not an option. Another solution is required to arrive at correct BPEL code. The second solution is to rejoin all (dead-end) branches back to one end event. While converging branches is not always easy, it is the best way to arrive at correctly working BPEL code. It also improves communication with the developer, as the meaning is no longer ambiguous. In addition, it also solves the issue with multiple callbacks to the client.

Guideline 9

Avoid multiple end events. Join them.

7.8 Limitation 7: Combination of constructs and structures

The obstacle encountered in diagram 12 is a wicked problem. It does not have a clear cause, and therefore no simple solution. Splitting the process into multiple processes provides a workaround for this problem, where while-loops in combination with parallel processing cause unexpected, incorrect BPEL code. As combining sub-processes also causes this problem in diagram 14, splitting them again solves the problem. Literature also advocates the reduced granularity, as short processes are often more usable (Olsen, 2006). Besides usability, research indicates that larger diagrams include more errors (Mendling, Neumann, & Aalst, 2007). Based on those observations, it is advised to decompose diagrams, and keep them below 50 elements (Mendling, Reijers, & Aalst, 2008). When calling sub-processes from the main process, the BPEL code must make a synchronous call to the sub-process. This ensures the sub-process completes before the process continues. Communication (of data) between sub-processes may be difficult with this solution. EPC has no construct to pass the flow of control to a sub-process, which its extension, eEPC, has. The modeler has to indicate it by annotations and labeling a function and events appropriately. Several examples of possible sub-processes appear in the composite case diagram. They include the removed loops around “Check execution posting process”, “Load costs in CMS”, and “Load hours in CMS”. In most cases, the source model describes them as sub-processes or variants already.

Guideline 10

Decompose processes containing problematic structures, such as loops.

7.9 Limitation 8: Block-structured versus graph-structured

One limitation remains that the diagram transformations do not reveal directly. Yet, it is the underlying source for some of the limitations. As chapters 3 and 4 mention, the

transformation from EPC to BPEL is hard, because the source language is graph-structured and the target language is block-structured. The impossibility to transform arbitrary cycles (see section 5.4.6), and the ambiguities of dead-end branches (see limitation 6) confirm the difficulty.

Categories for the degree of freedom in modeling languages exist, ranging from standard, through safe and structured, to synchronizing (Kiepuszewski, 2003). Graph-structured languages usually fall within the standard category, which allows for a high degree of freedom. That is the case for EPC. Block-structured languages fall within the structured category per definition. They enjoy a lower degree of freedom, as is the case with BPEL. The same difference arose in programming languages, where the use of structured programming is now preferred to the use of “spaghetti”-programming. The difference in freedom causes the problems.

Structured languages require that all splitting connectors match a joining connector of the same type, and vice versa. The connectors must nest properly, according the last-in first-out (LIFO) rule. For example, if an AND-split precedes a XOR-split, the XOR-split must close first, and then the AND-split must close too. In this way, every structure is contained within another structure. Structures do not partly overlap. Viewing the structures as a balanced bracket formula can help. Most of the diagrams in 0 - are structured, thanks to the measures taken to transform them successfully and correctly. Several of the above guidelines also serve to create better-structured diagrams.

Using only structured models is a severe restriction. Fortunately, most unstructured diagrams also possess a structured form. Combining multiple start events is an example of it. Duplicating constructs and combining/dividing connectors are other ways to obtain structured diagrams. Literature provides a vast amount of information on structuring unstructured diagrams, including which cases are possible and impossible to adjust (Kiepuszewski, 2003), and how adjust arbitrary cycles to structured diagrams (Zhao, Hauser, Bhattacharya, Bryant, & Cao, 2006).

Guideline 11	Create structured models.
---------------------	---------------------------

7.10 General guidelines

Successful and correct transformation requires the above guidelines. Modelers should use other guidelines too. For example, it is good practice to check back on the resulting BPEL diagram (and code), after the transformation finished. The difference between successful and correct transformation signifies the importance of this check. Other issues, dedicated to BPEL, include the choice between synchronous and asynchronous processes, and the otherwise/default addition to XOR-splits. In order to arrive at correct, executable BPEL code, it is important to communicate between the modeler and the developer. Agreeing on modeling conventions provides a good start for improving communications (Becker, Rosemann, & Uthmann, 2000). The conventions can include basic things, such as using one main route in the diagram and the use of labels. The choice between horizontal and vertical modeling also has an impact on clarity. The use of process management methodologies, such as ARIS (Scheer & Schneider, 1992) or Pronto (Noorman, 2008), can also play an important role, both in communicating and in the modeling process. However, as the above is not strictly necessary for the transformation from EPC to BPEL, these guidelines are beyond the scope of this research.

Table 24: List of guidelines

Nr.	Priority	Order	Guideline
1	Must	1	Do not use the OR-connector.
2	Would	11	Use clear, descriptive labels.
3	Could	10	Alternate functions and events.
4	Should	9	Always follow XOR-splits with events.
5	Could	2	Avoid loops.
6	Must	3	If loops are necessary, then use only while-loops with a single exit.
7	Should	4	Avoid multiple start events.
8	Could	6	Minimize the amount of arcs attached to a construct.
9	Should	5	Converge multiple end events.
10	Should	8	Decompose processes containing problematic structures.
11	Must	7	Create structured models.

7.11 Applying the guidelines

Table 24 lists the guidelines conceived in this chapter. The guidelines are prioritized, according to their importance for successful and correct transformation. The imperative words Must, Should, Could, and Would describe the guidelines, descending in importance. Guidelines 1, 6, and 11 are necessary for successful and correct transformation. For

example, offending the first guideline results in an error, instead of transformation of the diagram. Guideline 2, on the other hand, only serves for communication, and is not necessary for transformation. For example, naming a function differently does not change the control flow.

It is possible to apply the guidelines in any order. The order in which the limitations arose is the only reason for the numbering. However, it is best to follow the order that the third column of Table 24 shows. In that order, the first six guidelines solve issues for transformation that are relatively simple to solve. These issues are the OR-connector, loops, multiple start/end events, and constructs with a too high degree. Solving these issues also makes the next steps easier.

These next two steps (guidelines 11 and 10) solve harder issues for transformation. For guideline 11, several algorithms exist to create structured models from unstructured ones. The Event-Condition-Action algorithm is an example (Ouyang, Dumas, Breutel, & Hofstede, 2006). The algorithm boils down to finding the smallest structured element, then collapsing that to a single activity, and finding the element that is smallest then. The algorithm repeats until no structured elements exist anymore. When the algorithm reaches that point, guideline 11 provides a way to solve any unstructured parts by splitting that part. As a rule of thumb, guideline 11 should split the process at the point where most issues arise. After splitting the process, guideline 10 may be useful again. Instead of using such an algorithm, however, it is easier to model in a structured way from the start.

The final three guidelines are mainly for communication. These steps require that the other steps are complete already, iteration may be necessary otherwise. For example, decomposing the process in guideline 10 can break the alteration of events and functions.

The above steps can be viewed as a normalization algorithm. It is a model transformation on its own, as it transforms one EPC model, which cannot transform correctly, to another EPC model, which can. Further details and (partial) automation of the algorithm remain an issue for future research.

7.12 Validation of the guidelines in the composite case

This section applies the guidelines to the case. If the resulting diagram transforms successfully and correctly, then that validates the guidelines. Two ways exist to apply the guidelines to the case. The next two sections describe those ways. The first way is to start from the simplest form possible, and build a structured, transformable process from scratch. The second way is to take the case as a whole, and apply the guidelines as the previous section describes.

7.12.1 Applying the guidelines while modeling

This way of applying the guidelines starts with the individual, original diagrams. These are even simpler than the ones in diagrams 8 until 13. They are decomposed already. All exceptions are modeled as alternatives to the main flow. These exceptions include all seven of the loops, both the decisions, and a sequence. This results in sixteen diagrams in total, which Appendix C - Original case EPC diagrams - shows.

Guideline 1 only applies to “Load costs in CMS”. It is the alternative of “Update report data”, which contains the OR-connector. As section 6.7 suggests, a XOR-connector replaces the OR-connector. The next guidelines, 5 and 6, are present in the diagrams already. The six diagrams containing the main control flow have no cycles. The simpler diagrams only contain while-loops. The two guidelines on multiple start events and end events, 7 and 9, are skipped, as these events do not lead to issues in these diagrams. Guideline 8 has two issues to solve. The first is in “Determine cost levels”. The solution is similar to the one in section 6.4, where extra AND-joins combine the control flow in a structured way. The difference is that the branches do not connect with an extra AND-split at the top. The second place is “Preliminary rebilling”. In parallel to the other place, and section 6.6 an extra AND-join solves the issue. It is not necessary to apply guideline 11, as all the diagrams are structured. Therefore, guideline 10 is surplus, as no problematic structures exist anymore. The final three guidelines are all present in the diagrams too. The syntax of EPC enforced this already. The resulting sixteen diagrams all transform successfully and correctly.

This section shows that applying the guidelines while modeling is relatively easy. The resulting model has many dependencies though. A developer has to fill the implementation details for each connection between the diagrams. As a result, the benefits are limited.

7.12.2 Applying the guidelines to an existing model

This second way of applying the guidelines starts from the full diagram. Coupling all the original diagrams 8 until 13, based on the Value Added Chain diagram in Figure 44, and the process interfaces, produces the full diagram 15. Appendix E - EPC diagrams of full case - contains this diagram. This full case diagram contains all the exceptions, which had separate diagrams in the previous sections.

For this diagram, guideline 1 works at the same place and in the same manner as the previous section. Two XOR-connectors replace the two OR-connectors. The diagram fulfills the next two guidelines, 5 and 6. All loops are while-loops. However, most of these loops cause issues later on in the process. As they are in place, it is not possible to avoid them anymore.

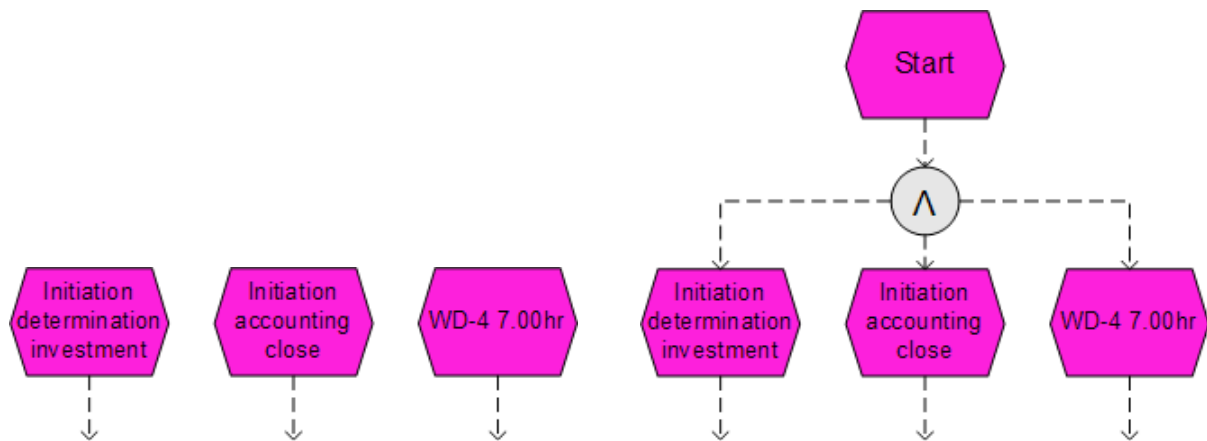


Figure 52: Join multiple start events

The next two guidelines, 7 and 9, deal with multiple start events and end events. In this full case diagram, both are an issue. Adding an extra AND-split and a single start event joins the branches at the start in a simple way. Figure 52 shows this. The multiple branches at the end cause more difficulties. Similar to the start branches, adding an AND-join merges the branches from “Preliminary rebilling” and “Report”. The branches from “Check hour download” are harder to merge. The first one indicates an error in the model. Instead of stopping when the event “Hour download started manually” occurs, the process should continue to “Check if download has finished”. Figure 53 shows how to fix this. Adding an extra XOR-join after the freshly created AND-join merges the second loose end. Finally, an extra end event follows this connector. These steps solve the multiple start events and end events.

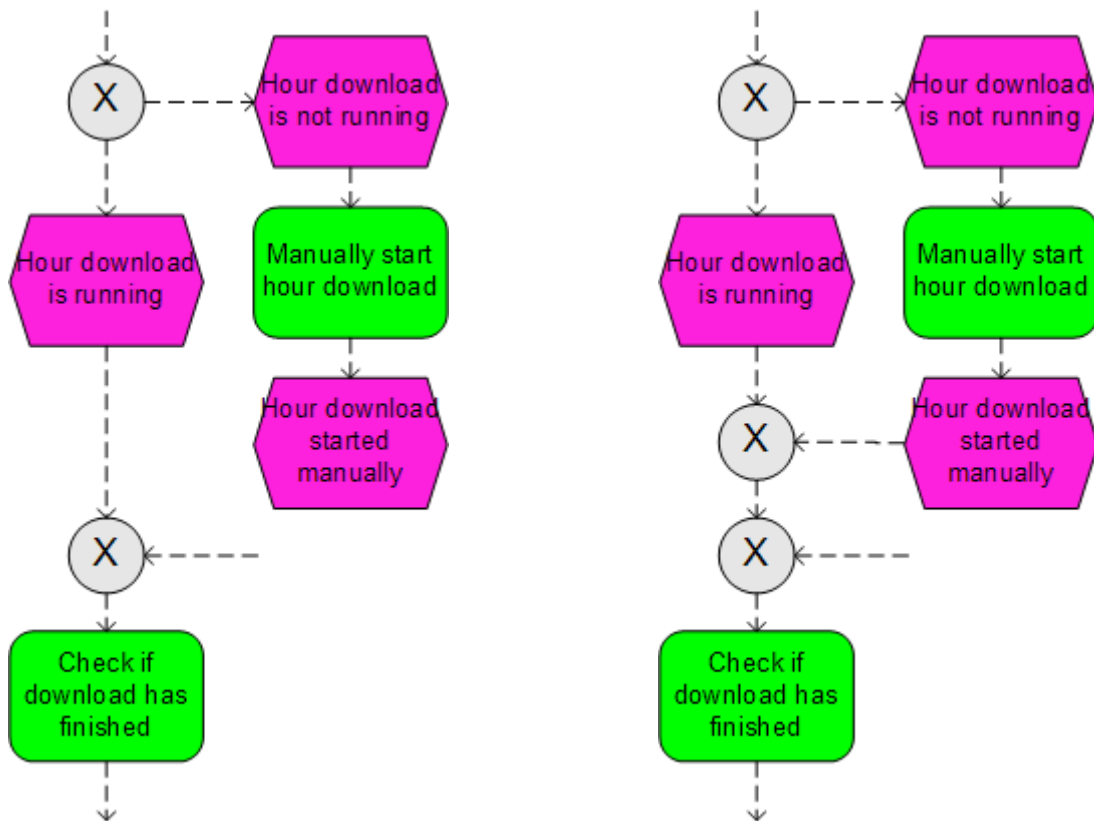


Figure 53: Fixing a loose end

Guideline 8 requires the addition of several AND-connectors, in order to reduce the degree of some existing connectors. Besides the two issues that the previous section solves, combining the start events and end events caused new issues there. Adding an AND-join to join the branches of “Report” solves the issue at the end. Similar, adding an extra AND-split to combine the branches of “Determine cost levels” and “Prepare accounting close” solves the issue at the start.

The next guideline, 11, is the hardest to apply. The previous guideline structures part of the diagram already, but it is impossible to structure the entire diagram. Therefore, guideline 10 is required too. The left side of Figure 54 shows the diagram without any functions and events. This shows the organization of the model. In order to apply guidelines 10 and 11, all structured parts are collapsed. This results in the organization that the right side of Figure 54 shows. This figure shows what parts are impossible to structure (Kiepuszewski, 2003).

Guideline 10 serves to decompose the diagram into parts, which are possible to structure. In this case, the best place to start decomposing the model is where most issues arise. Cutting the diagram across lines A and B in Figure 54 produces a top, middle, and bottom diagram. Figure 55 shows each of these. In these diagrams, process interfaces indicate the places

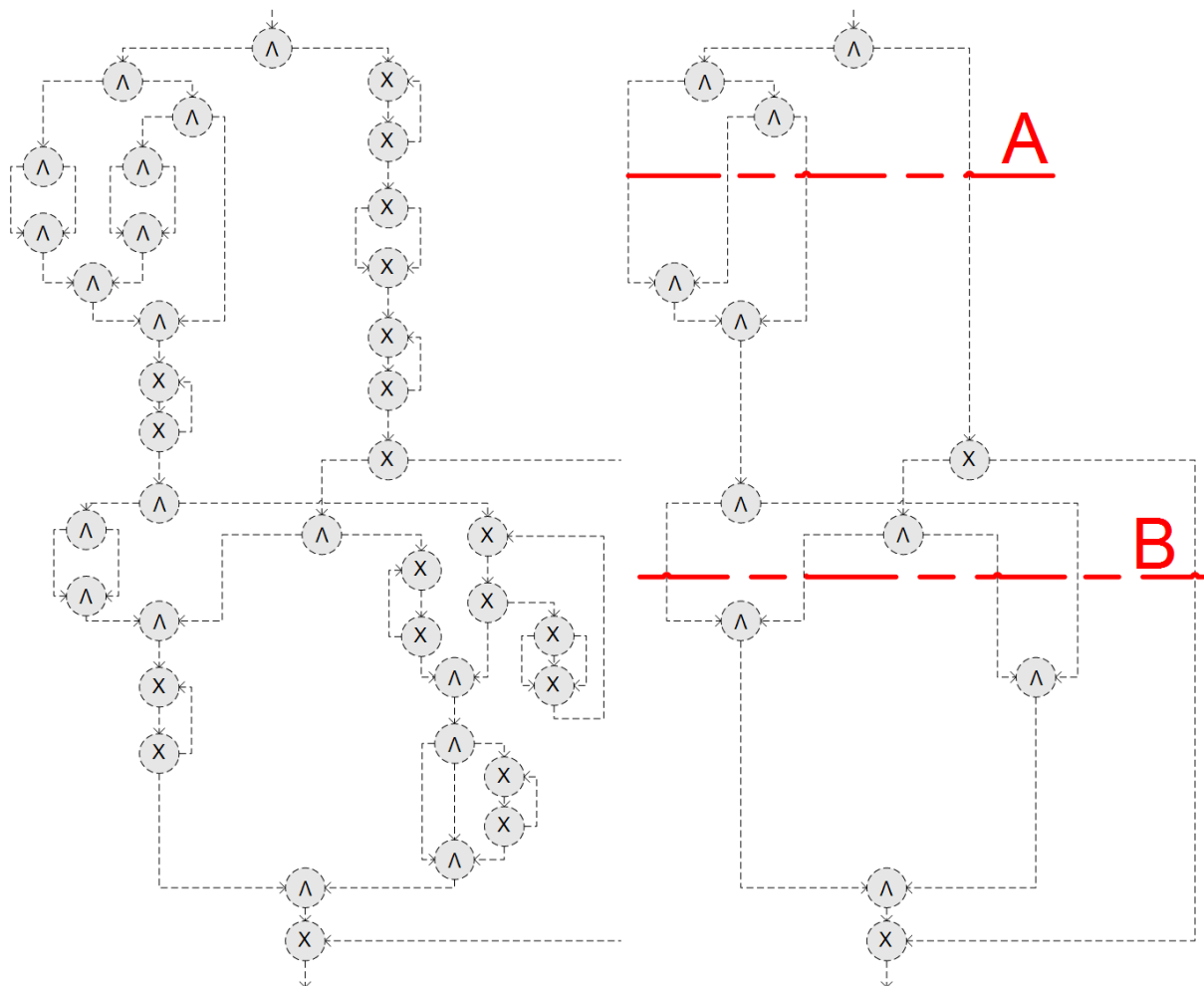


Figure 54: Structure of the full case, with only connectors and arcs

where the diagrams communicate with the other diagrams. Appendix D - Composite case EPC diagrams - provides larger versions of the diagrams.

The top diagram includes some modifications. It no longer includes the top two AND-splits, as the multiple start events needed these. Extra events at the bottom ensure the alteration of events and functions. That fulfills the final three guidelines.

The middle diagram is more complex. Removing the loops is required too, as they present the problem encountered in section 6.7. The loops do not transform properly when AND-connectors enclose them. Removing the loops produces the same sub-diagrams as in the previous section. A series of AND-joins combines the branches at the bottom. The branch, which originates from the XOR-split, connects to them. For this branch, an extra function is added to indicate that there is a serious error, which a human needs to solve. Adding an extra end event at the end and duplicate events at the start is enough to satisfy the three final guidelines.

The bottom diagram requires removal of the final XOR-connector, because the top diagram already resolves it. As with the middle diagram, removing the loops is required. For this diagram, applying the final three guidelines require the addition of a function before the final end event, and the addition of duplicate events before the process interfaces at the top. Both are required for the alternation of events and functions.

The resulting model has ten diagrams. Seven of these are the same as in the previous section; they are the while-loops. The other three are the top, middle, and bottom diagram in Figure 55. The resulting ten diagrams all transform successfully and correctly.

This section illustrates that applying the guidelines to an existing model is much harder than applying them while modeling from scratch. Especially, solving unstructured models is hard. It requires the choice to be made on where to cut the diagram, so that the decomposed parts are structured diagrams. The decomposition may be irrational from a business point of view. That is the situation for this case. The original model was decomposed into parts logically done by different departments. The diagrams from the decomposed model cover different departments, while they split at points within a department.

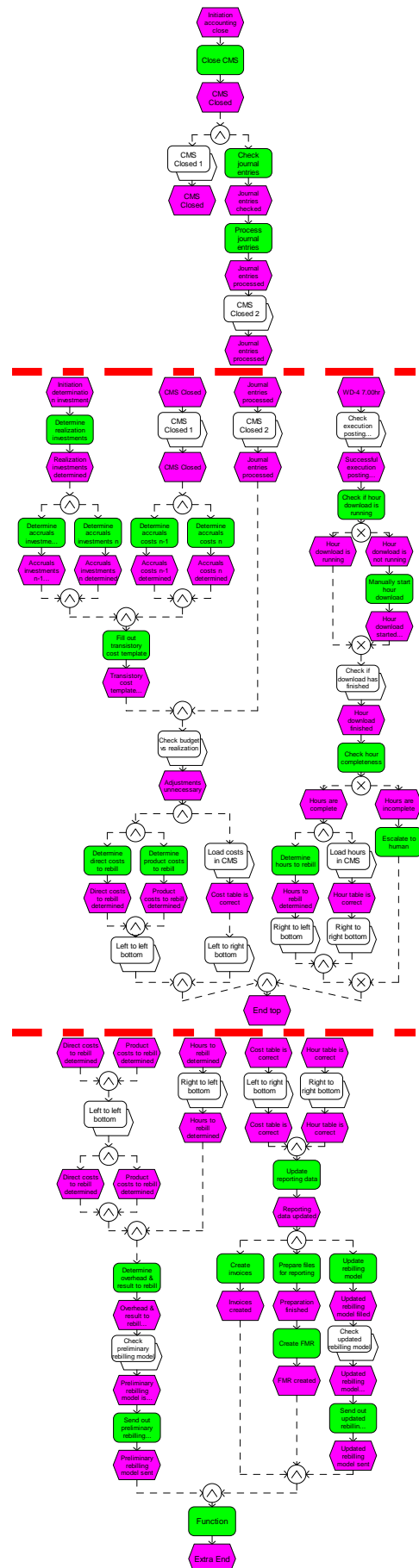


Figure 55: Decomposed full diagram

8 Discussion

This section provides a discussion of the research, in the form of an evaluation and future actions. The first part describes the validation and limitations of the research, as well as alternative paths that were beyond the scope of this research. The second part includes possibilities for further research and recommendations for practice.

8.1 Validity

Rigorous prior research provides the foundation for the conceptual model. Both workflow patterns and the BWW model proved their use individually. They did this in the areas of language evaluation, and transformation. Together, they are a more complete model. The prediction the model made, about the transformation of the small diagrams, shows its correctness for the case of EPC to BPEL transformation. As these predictions from theory match the results obtained during the empirical research, the conceptual model is internally valid. The research does not directly provide external validity of the conceptual model. However, it is highly probable that the model applies to transformations in different languages too, as both of its parts, ontology and patterns, were used individually for other languages (Recker, Rosemann, Indulska, & Green, 2006) (Wohed, Aalst, Dumas, Hofstede, & Russell, 2005).

The results of the model transformations, as chapters 4 and 5 give them, are valid only for the Oracle BPA Suite. It is even limited to the version (10.0.3.4) used. Other tools and newer versions may transform the diagrams in another way. For example, they could support the OR-connector, or forbid WFCP 10 – Arbitrary Cycle in its totality. Internal validity is present, as the research tests all elements of the conceptual model and the results match to the expectations. The case extends the validity, as it shows that the results also hold in practice.

Applying the guidelines to the case provides empirical validation of them. The results prove the internal validity of the guidelines, as the resulting diagrams are transformable. As the guidelines are only applied to a single case, external validity of the guidelines is low. Several of the guidelines may apply to general (business process) modeling, but some of them are more specific. For example, alternating functions and events is typical for EPC, and using only while-loops is specific to the transformation to BPEL.

8.2 Limitations

This research knows several limitations. A tight scope is the cause for most of the limitations. The next section reflects this further, as it names alternatives that did not fit within the scope. That section also discusses the effects of not taking the alternatives. Access to some sources was difficult too, as some of the main research on EPC only exists in German. Due to this, some information may be misunderstood or missed altogether. Due to time constraints, there is no further empirical validation of the guidelines. Only a single case from practice illustrates their application.

The mapping of EPC to BPEL in the conceptual model (section 4.4) is not normative. The choice for a specific mapping of constructs that lack clarity causes this limitation. While logic provides reasoning for the choices made, they lack absolute objectivity. It is not possible to overcome this limitation. It will be present for any mapping of constructs lacking clarity.

While the combination of workflow patterns and the BWW model provides a complete conceptual model, the two criteria also overlap on some points for the transformation from EPC to BPEL. For example, the BWW concept lawful transformation overlaps with the workflow patterns where the control flow joins, such as WFCP 3 - Synchronization. Removing the overlap improves (the clarity of) the model.

As section 3.1.2 discusses, the choice for the BWW model to evaluate a business process modeling language is debatable, as it has excess on several parts, and is incomplete on some others. For transformation, this is less of an issue. The combination with workflow patterns solves the incompleteness, as the workflow patterns fill the gaps. Evaluation of both languages solves the excess, as only the comparison is important for transformation.

8.3 Alternatives

This section provides several alternative paths the research could have taken. It did not, because of the limited scope enforced by time constraints. The alternatives are options for further research in addition to those specified in the next section.

No other tool that provides EPC to BPEL transformation was found other than the Oracle BPA Suite, and IDS Scheer's SOA Architect, which is its basis. Other tools may exist, or have

been developed since. Such a tool could serve for a comparison of capabilities and further validation of the conceptual model.

While no other tool was found for direct EPC to BPEL transformation, transformation from EPC to BPEL is also possible by a diversion. It is possible to go from EPC to another language, and then from that other language to BPEL. Theory development shows promise in this area, especially if using EPML (EPC Markup Language) and AML (ARIS Markup Language) as the other language (Mendling & Nüttgens, 2004). Some tools are available that do the individual steps.

The research shows what the Oracle BPA Suite is capable of, and what it cannot do. However, it is also interesting to know why it cannot do certain things. The conceptual model explains most of the difficulties discovered in the pattern transformations, but the difficulties in the case transformation have other causes too. They are specific to the implementation of the transformation in the Oracle BPA Suite. Instead of guidelines for modelers, the research could have offered advice for implementation changes.

8.4 Further Research

As this research deals with a restricted scope, it does not handle several possibly interesting research topics. These are beyond its scope, but have a direct relation to the research.

This research handles transformation from EPC to BPEL. Much research on transformation from BPMN to BPEL also exists. A comparison of the two transformations based on the conceptual model is now a possibility. The comparison could lead to a more founded choice for the use of BPMN or EPC over the other. It may also shed light on general issues of BPEL, which need improvement.

One of the two criteria in this research is the BWW model. As section 3.1.2 discusses, other approaches to evaluate a modeling language are possible too. A comparison of such approaches is an item for further research.

Besides ontology, this research bases its conceptual model on the basic workflow control patterns. Further research can also handle the other workflow patterns in the same manner. This includes the advanced control flow patterns, as well as the data and resource patterns.

Several sources (Mulyar, 2005) use the BPEL link construct much more. The use of this construct allows for a transformation, which is applicable for more cases. This is good for automatic transformation. However, it has several drawbacks. Especially for understandability, the method may be unacceptable.

The main limitation for applying this research in practice is that the Oracle BPA Suite does not deliver executable code. In order to arrive at executable code, the modeler has to provide more than just the modeled control flow. Executable code needs at least a data model, and the interaction with partners. Therefore, a question for further research is “What does transformation to executable code require from the input model?” This research answers it for the control flow.

8.5 Recommendations

Implications of this research for Sogeti exist for both the divisions DSE and A&BS. Professionals in A&BS do most of the modeling. In the BPM Lifecycle (see Figure 5), they are responsible for modeling and simulating. ARIS EPC is one of their competencies. The guidelines provided in chapter 7 apply for the modelers in A&BS, who work with EPC. Several of the guidelines apply always, while some apply when transforming to BPEL only. The modelers generally know and use the guidelines that apply always, already. The case from practice illustrates this. The main recommendation for Sogeti is to use the guidelines when applicable.

On the other side, the professionals in division DSE take care of the development. They work on the BPEL code, which the transformation produces. Within DSE, the Oracle units are the key target. Filling the implementation details, which the tool leaves open, is the developers work. Besides implementation, they take care of deployment and execution, within the BPM Lifecycle. The main lessons learned from this research for the developers are the limitations and workarounds of the transformation from EPC to BPEL in the Oracle BPA Suite. Knowing the limitations helps the developer see where problems arise, so they can solve them. The workarounds are possible solutions to the limitations.

Communication between the modelers and the developers is important. While the plain diagrams cover a lot of information already, they still contain ambiguities. Within the diagrams, annotations and labeling improve communication. However, other means of

communicating are helpful too. As long as the business process diagrams are not directly executable, human interference and interpretation is necessary. Interpretation naturally leads to misunderstanding. Communication has to limit that.

The implications for business differ depending on the point of view taken. From the A&BS (modeler) perspective, the promise of MDE is great. Model transformation prevents certain restraining steps in the BPM Lifecycle. However, this research shows that it is hard to realize the promise completely. The implementation of the Oracle BPA Suite includes several issues, which require interference of a developer. This reduces the benefits of automatic transformation. Additionally, the costs of following the guidelines can be high. The restrictions placed on control flow and decomposition may be unreasonable. On the other hand, the benefits gained from partial automated transformation may already outweigh the costs. The benefits are in reduced human work in both development and communication. The costs are in the license for the tool and the human work required adapting the EPC model for transformation. The balance between costs and benefits depends on the individual case.

From the DSE (developer) perspective, the use of the Oracle BPA Suite results in two reactions. Firstly, if the promise of MDE with fully automated transformation works, then the developers are no longer necessary. Secondly, if the promise does not fully work, gaps exist for the developers to fill. This research indicates that the second case is the current situation. The implementation of the transformation in the Oracle BPA Suite forces the use of developers. For the developers this is an opportunity. They can offer their specialization anytime businesses use the Oracle BPA Suite.

9 Conclusions

This chapter sums up the findings of this research. It does this by handling the answers to the research sub-questions first, as the previous chapters answer them. The answers appear in the order in which chapter 2 presents the questions. Then, section 9.2 answers the main research questions. Finally, section 9.3 shows how the answers to the research questions contribute to the research objectives.

9.1 Answers to research sub-questions

According to literature, business process modeling languages must adhere to the Bunge-Wand-Weber (BWW) representational model (Wand & Weber, 1990). For the case of transforming EPC to BPEL, the model makes several issues clear. These issues are lack of completeness and lack of clarity of the modeling languages. Some parts of these issues can cause difficulties for the transformation. Table 11 in section 4.3 lists these difficulties.

Workflow patterns are the most commonly used patterns in business processes (Aalst, Hofstede, Kiepuszewski, & Barros, 2003). EPC and BPEL are able to represent different sets of these patterns. For transformation, the only situation leading to difficulties is where the source language is able to represent a pattern that the target language cannot. For the case of transforming EPC to BPEL, only workflow control pattern (WFCP) 10 causes difficulties. WFCP 10 is the Arbitrary Cycle.

Section 4.4 presents a conceptual mapping from EPC to BPEL. As several design choices are made, the mapping is not normative and other mappings are possible. Tables 12, 18, and 19 list the mapping. Most noticeable in the mapping is that WFCP 10 is forbidden in general, as it has no mapping for all cases.

The Oracle BPA Suite serves as the tool for transforming EPC to BPEL. Known constraints for transformation include the difficulties encountered in the conceptual model. With the exception of the OR-connector and the Arbitrary Cycle, the Oracle BPA Suite successfully and correctly transformed all patterns. The used version of the tool (10.0.3.4) does not support the OR-connector, which WFCP 6 and 7 require. The Arbitrary Cycle, WFCP 10, has no general mapping from EPC to BPEL. The tool is only able to transform a subset of the pattern, the while-loop.

Workarounds exist for the patterns that failed transformation. In case of the OR-connector, it is always possible to rewrite it to a combination of XOR-connectors and AND-connectors. However, this leads to an exponential increase in constructs. Often, it is better to replace it simply by the XOR-connector. It is possible to rewrite many Arbitrary Cycles to a form BPEL is able to represent. If rewriting only requires restructuring of the model, this is not an issue. However, rewriting often requires the process to be split into sub-processes. This is usually unacceptable, as the model becomes too hard to understand.

Classic EPC diagrams capture only the control flow of the business process. Transformation of the control flow to BPEL requires nothing extra from a diagram, which the modeling of a correct EPC diagram does not already require. Of course, an exception to this are the difficulties encountered before. In order to obtain executable code through transformation, extra data is required. That goes beyond the scope of this research.

Several limitations arose during the transformation of the larger, composite case. Besides the difficulties found before, a lack of structured modeling was the main issue. Workarounds exist for all limitations. The guidelines in Table 24 (section 7.11) capture them. The most important guidelines are to avoid the OR-connector, use only while-loops if using loops at all, and create structured models. Applying the guidelines leads to models that transform successfully.

Guideline 10 provides a means to transform complex and hard-to-transform diagrams. As with some of the other guidelines, this may lead to diagrams that are no longer fit for any other purpose. They may even be too hard to understand, especially for the developer but also for the modeler who decomposed the diagram. Often, this is not acceptable. The balance between the need for (automated) transformation and understandability must be rated on a case-to-case basis.

9.2 Answers to main research questions

Automated transformation from EPC diagrams to BPEL specifications is possible to a large extent. Chapter 4 provides a mapping from EPC to BPEL. This indicates which concepts, constructs, and patterns can transform from EPC to BPEL. According to this theory, several difficulties exist for the transformation. Table 11 in section 4.3 lists these issues. The mapping already deals with most of the issues. For one issue, only a mapping for some cases

exists. This is the Arbitrary Cycle of WFCP 10. The transformation of this pattern is not always possible, according to theory. All other patterns can transform. The mapping also handles all issues that arose during ontological evaluation of the languages. The issues of lack of clarity and lack of completeness of the two languages are either not significant for the transformation, or require a choice to be made. By picking alternatives for these choices, the mapping solves the issues, but is not normative.

The effectiveness of automated transformation in the Oracle BPA Suite is lower than what was expected based on the theory. The theoretical limitations and possibilities, which the conceptual model describes, are put to the test in chapters 5 and 6. The pattern diagrams include all found issues. With the exception of the OR-connector, the results of the pattern transformations are nearly exact what the conceptual model predicted. This version (10.0.3.4) of the Oracle BPA Suite forbids the OR-connector, so a workaround is necessary. Further differences from the expected results arose due to the choices made in the conceptual mapping.

Based on the pattern transformation alone, the Oracle BPA Suite was able to transform nearly everything that is theoretically possible. It only failed in case of the OR-connector. However, the results of transforming a case from practice greatly reduced this effectiveness. Many new issues arose when attempting to transform the composite case. The lack of structured modeling is the main cause for this. Other issues include the use of multiple start events and end events, and loops nested within a parallel flow. Structuring the diagrams is the workaround solving most issues. Usually, splitting diagrams into sub-processes can solve the remaining, complex issues, which structuring diagrams cannot solve. This workaround is often not acceptable, as it results in unreadable, hard-to-understand models.

Chapter 7 presents a list of guidelines in Table 24. Following these guidelines results in EPC diagrams, which the Oracle BPA Suite can automatically transform to BPEL specifications. This provides methodological support for EPC to BPEL transformation feasibility. These BPEL specifications are not directly executable, as implementation details are missing. These details require the addition of data models among other things. That is beyond the scope of this research, which only handles the control flow. As applying guideline 10 may lead to unreadable models, this may only be acceptable in case no humans need to work with the

results anymore. In this research, that is never the case, as a developer still has to complete the implementation details.

9.3 Contributions

This research contributes in three ways. First, it contributes to theory by expanding the knowledge of EPC to BPEL model transformation. The conceptual model in chapter 4 captures this knowledge. It provides a framework for evaluating the possibilities of a transformation between two languages. The BWW model and the workflow control patterns form the basis of the framework. The conceptual mapping from EPC to BPEL and the list of issues for this transformation are a specific instance of the model.

The second contribution lies in the validation and application of the framework for the specific case of EPC to BPEL transformation as done by the Oracle BPA Suite. The transformation results in chapter 5 demonstrate that the conceptual model is correct. Chapter 6 reveals more limitations, by applying the transformations to a case from practice. Some of these limitations are specific to (version 10.0.3.4 of) the Oracle BPA Suite. Together, the theoretical and practical limitations show what to expect during model transformation in practice.

The final contribution is a list of guidelines, in chapter 7. This is a methodological contribution to practice. The guidelines are specific for EPC modeling and models, which are meant for transformation to BPEL. Modelers can apply these guidelines to improve the feasibility of EPC to BPEL transformations. When they apply all the guidelines, the Oracle BPA Suite can transform the models successfully and correctly.

List of References

- Aalst, W. M. (1999). Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41 (10), 639-650.
- Aalst, W. M., & Hofstede, A. H. (2005). YAWL: Yet Another Workflow Language. *Information Systems*, 30 (4), 245-275.
- Aalst, W. M., & Lassen, K. B. (2005). *Translating Workflow Nets to BPEL*. Eindhoven, The Netherlands: Eindhoven University of Technology.
- Aalst, W. M., Barros, A., Hofstede, A. H., & Kiepuszewski, B. (2000). Advanced Workflow Patterns. In O. Etzion, & P. Scheuermann (Ed.), *Proceedings of the 7th International Conference on Cooperative Information Systems* (pp. 18-29). London, UK: Springer-Verlag.
- Aalst, W. M., Dumas, M., Hofstede, A. H., Russell, N., Verbeek, H., & Wohed, P. (2005). Life After BPEL? In *Formal Techniques for Computer Systems and Business Processes* (pp. 35-50). Berlin / Heidelberg, Germany: Springer.
- Aalst, W. M., Hofstede, A. H., Kiepuszewski, B., & Barros, A. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14 (1), 5-51.
- Aalst, W. M., Jorgensen, J., & Lassen, K. B. (2005). Let's Go All the Way: From Requirements Via Colored Workflow Nets to a BPEL Implementation of a New Bank System. In R. Meersman, & Z. Tari (Eds.), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE* (Vol. 3760/2005, pp. 22-39). Berlin / Heidelberg, Germany: Springer-Verlag.
- Aksit, M. (2004). The C'7 for Creating Living Software: A Research Perspective for Quality-Oriented Software Engineering. *Turkish Journal of Electrical Engineering & Computer Sciences*, 12 (2), 61-96.
- Alanen, M., Lilius, J., Porres, I., & Truscan, D. (2003). *Realizing a Model Driven Engineering Process*. Turku, Finland: Turku Centre for Computer Science.
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A pattern language*. New York, USA: Oxford University press.

Becker, J., Rosemann, M., & Uthmann, C. v. (2000). Guidelines of Business Process Modeling. In W. M. Aalst, J. Desel, & A. Oberweis (Eds.), *Business Process Management, LNCS 1806* (pp. 30-49). Berlin / Heidelberg, Germany: Springer-Verlag.

Bézivin, J. (2004). In Search of a Basic Principle for Model Driven Engineering. *UPGRADE* , 5 (2), 21-24.

Bézivin, J., Farcet, N., Jézéquel, J.-M., Langlois, B., & Pollet, D. (2003). Reflective Model Driven Engineering. *Proceedings of the 6th International Conference on the Unified Modeling Language: Modeling Languages and Applications (UML 2003)* (pp. 175-189). San Francisco, USA: Springer.

BiZZdesign. (2006). *Handboek BPEL*. Enschede.

Bodart, F., Patel, A., Sim, M., & Weber, R. (2001). Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests. *Information Systems Research* , 12 (4), 384-405.

Bunge, M. (1977). *Treatise on Basic Philosophy (Volume 3), Ontology I: The Furniture of the World* (Vol. 3). Dordrecht, The Netherlands: Reidel.

Bunge, M. (1979). *Treatise On Basic Philosophy: Volume 4: Ontology II: A World of Systems* (Vol. 4). Dordrecht, The Netherlands: Reidel.

Burton-Jones, A., & Meso, P. N. (2006). Conceptualizing systems for understanding: an empirical test of decomposition principles in object oriented analysis. *Information Systems Research* , 17 (1), 38-60.

Butler Group. (2007). *Oracle BPM*. Hull, UK: Butler Direct Ltd.

Cuntz, N., & Kindler, E. (2005). On the Semantics of EPCs: Efficient Calculation and Simulation. In W. M. Aalst, B. Benatallah, F. Casati, & F. Curbera (Ed.), *Proceedings of the 3rd International Conference on Business Process Management (BPM 2005)* (pp. 398-403). Berlin / Heidelberg: Springer.

Czarnecki, K., & Helsen, S. (2003). Classification of Model Transformation Approaches. *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, (pp. 1-17). Anaheim.

Dongen, B. v., Jansen-Vullers, M., Verbeek, H., & Aalst, W. M. (2007). Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, 58 (6), 578-601.

Everman, J., & Wand, Y. (2001). Towards Ontologically Based Semantics for UML Constructs. In H. Kunii, S. Jajodia, & A. Sølvberg (Eds.), *ER 2001, LNCS 2224* (Vol. 2224, pp. 354-367). Berlin / Heidelberg, Germany: Springer-Verlag.

Falkenberg, E. .., Hesse, W., Lindgreen, P., Nilsson, B., Oei, J., Rolland, C., et al. (1998). *A Framework of Information Systems Concepts*. International Federation for Information Processing.

Fettke, P., & Loos, P. (2003). Ontological evaluation of reference models using the Bunge-Wand-Weber model. *Proceedings of the Ninth Americas Conference on Information Systems*, (pp. 2944-2955). Tampa, USA.

Fondement, F., & Silaghi, R. (2004). Defining Model Driven Engineering Processes. *Proceedings of the 3rd Workshop in Software Model Engineering (WiSME 2004)*.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, USA: Addison-Wesley.

Gardner, T. (2003). UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. *Proceedings of the First European Workshop on Object Orientation and Web Services at ECOOP*. Darmstadt, Germany.

Gehlert, A., & Esswein, W. (2007). Toward a formal research framework for ontological analyses. *Advanced Engineering Informatics*, 21 (2), 119-131.

Gemino, A., & Wand, Y. (2005). Complexity and clarity in conceptual modeling: comparison of mandatory and optional properties. *Data & Knowledge Engineering*, 55 (3), 301-326.

Green, P., & Rosemann, M. (1999). An Ontological Analysis of Integrated Process Modelling. *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE 1999)* (pp. 225-240). London, UK: Springer-Verlag.

Green, P., & Rosemann, M. (2000). Integrated Process Modeling: An Ontological Evaluation. *Information Systems*, 25 (2), 73-87.

- Green, P., Rosemann, M., Indulska, M., & Manning, C. (2007). Candidate interoperability standards: An ontological overlap analysis. *Data & Knowledge Engineering*, 62 (2), 274-291.
- Gregor, S. (2006). The Nature of Theory in Information Systems. *Management Information Systems Quarterly*, 30 (3), 611-642.
- Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5 (2), 199-220.
- Gruhn, V., & Laue, R. (2006). How Style Checking Can Improve Business Process Models. In J. Barjis, U. Ultes-Nitsche, & J. C. Augusto (Ed.), *Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS 2006)* (pp. 47-56). Paphos, Cyprus: INSTICC Press.
- Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. Enschede, The Netherlands: Centre for Telematics and Information Technology, University of Twente.
- Hauser, R., & Koehler, J. (2004). Compiling Process Graphs into Executable Code. *Proceedings of the Third International Conference on Generative Programming And Component Engineerin (GPCE 2004)* (pp. 317-336). Vancouver, Canada: Springer.
- Hinz, S., Schmidt, K., & Stahl, C. (2005). Transforming BPEL to Petri Nets. In W. M. Aalst, B. Benatallah, F. Casati, & F. Curbera (Ed.), *Proceedings of the 3rd International Conference, BPM 2005* (pp. 220-235). Berlin / Heidelberg: Springer-Verlag.
- Hoyer, V., Bucherer, E., & Schnabel, F. (2008). Collaborative e-Business Process Modelling: Transforming Private EPC to Public BPMN Business Process Models. In A. t. Hofstede, B. Benatallah, & H.-Y. Paik (Eds.), *Business Process Management Workshops: BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws, Brisbane, Australia, September 24, 2007 : Revised Selected Papers* (4928/2008 ed., pp. 185-196). Berlin / Heidelberg, Germany: Springer.
- Keller, G., Nüttgens, M., & Scheer, A.-W. (1992). *Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)"*. Saarbrücken, Germany: Institut für Wirtschaftsinformatik Saarbrücken.

Kent, S. (2002). Model Driven Engineering. *Proceedings of the Third International Conference on Integrated Formal Methods* (pp. 286-298). London, UK: Springer-Verlag.

Kiepuszewski, B. (2003). *Expresiveness and Suitability of Languages for Control Flow Modelling in Workflows*, PhD Thesis. Brisbane, Australia: Queensland University of Technology.

Kiepuszewski, B., Hofstede, A. H., & Aalst, W. M. (2003). Fundamentals of control flow in workflows. *Acta Informatica* , 39 (3), 143-209.

Kindler, E. (2006). On the semantics of EPCs: Resolving the vicious circle. *Data & Knowledge Management* , 56 (1), 23-40.

Koehler, J., & Vanhatalo, J. (2007). Process Anti-Patterns: How to Avoid the Common Traps of Business Process Modeling. *IBM WebSphere Developer Technical Journal* , 10 (2 & 4).

Koschmider, A., & Mevius, M. (2005). A Petri Net Based Approach for Process Model Driven Deduction of BPEL Code. In R. Meersman, Z. Tari, & P. Herrero (Eds.), *On the Move to Meaningful Internet Systems 2005: OTM Workshops* (pp. 495-505). Berlin / Heidelberg, Germany: Springer.

Luftman, J., Kempaiah, R., & Nash, E. (2006). Key issues for IT executives 2005. *MIS Quarterly Executive* , 5 (2), 81-99.

Mendling, J., & Nüttgens, M. (2004). Transformation of ARIS Markup Language to EPML. *Proceedings of the 3rd GI Workshop on Event-Driven Process Chains*, (pp. 27-38). Luxembourg.

Mendling, J., & Ziemann, J. (2005). Transformation of BPEL Processes to EPCs. *Proceedings of the 4th GI Workshop on Event-Driven Process Chains (EPK 2005)* , 167, 41-53.

Mendling, J., Lassen, K. B., & Zdun, U. (2007). On the transformation of control flow between block-oriented and graph-oriented process modeling languages. *International Journal of Business Process Integration and Management* (2).

Mendling, J., Lassen, K. B., & Zdun, U. (2006). Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. In F. Lehner, H. Nösekabel, & P.

Kleinschmidt (Ed.), *Multikonferenz Wirtschaftsinformatik 2006, Band 2* (pp. 297-312). Berlin: GITO-Verlag.

Mendling, J., Neumann, G., & Aalst, W. M. (2007). Understanding the Occurrence of Errors in Process Models Based on Metrics. In R. Meersman, & Z. Tari (Ed.), *Proceedings 15th International Conference on Cooperative Information Systems (CoopIS 2007)* (pp. 113-130). Vilamoura, Portugal: Springer-Verlag.

Mendling, J., Neumann, G., & Nüttgens, M. (2005). Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In M. Nüttgens, & J. Mendling (Ed.), *Proceedings of the 2nd Workshop XML4BPM*, (pp. 23-38). Karlsruhe, Germany.

Mendling, J., Reijers, H., & Aalst, W. M. (2008). *Seven Process Modeling Guidelines (7PMG)*.

Moody, D. L. (2005). Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering* , 55 (3), 243-276.

Mulyar, N. A. (2005). *Pattern-based Evaluation of Oracle-BPEL (v.10.1.2)*. BPMcenter.org.

Mulyar, N. A., & Aalst, W. M. (2005). *Patterns in Colored Petri Nets*. Eindhoven: Eindhoven University of Technology.

Mulyar, N. A., Aalst, W. M., Hofstede, A. H., & Russell, N. (2006). *Towards a WPSL: A Critical Analysis of the 20 Classical Workflow Control-flow Patterns*. BPMcenter.org.

Noorman, B. (2008). *White Paper - Pronto: BPM-aanpak Sogeti*. Vianen: Sogeti Nederland B.V.

Nüttgens, M., & Rump, F. J. (2002). Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel, & M. Weske (Ed.), *Proceedings of Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (PROMISE 2002)* (pp. 64-77). Potsdam, Germany: Gesellschaft für Informatik.

Nüttgens, M., Feld, T., & Zimmermann, V. (1998). Business Process Modeling with EPC and UML: Transformation or Integration? (M. Schader, & A. Korthaus, Eds.) *The Unified Modeling Language - Technical Aspects and Applications* , 250-261.

OASIS. (2007). *A Primer: Web Service Business process Execution Language Version 2.0*.

- OASIS. (2003). *Business Process Execution Language for Web Services Version 1.1*.
- OASIS. (2007). *Web Service Business process Execution Language Version 2.0*.
- Object Management Group. (2008). *Business Process Modeling Notation, Version 1.1*.
- Object Management Group. (2003). *MDA Guide Version 1.0.1*.
- Object Management Group. (2008). *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*. Object Management Group.
- Olsen, T. G. (2006). Defining Short and Usable Processes. (K. Johnstun, Ed.) *CROSSTALK The Journal of Defense Software Engineering*, 19 (6), 24-28.
- Opdahl, A. L., & Henderson-Sellers, B. (2002). Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model. *Software and Systems Modeling*, 1 (1), 43-67.
- Oracle. (2006). *Business Process Management and WS-BPEL 2.0. What's next for SOA Orchestration? An Oracle White Paper*. Redwood Shores, USA: Oracle Corporation.
- Oracle. (n.d.). *Oracle Business Process Analysis Suite Data Sheet*.
- Oracle. (2006, December). *Oracle Business Process Analysis Suite: Overview & Product Strategy*. Retrieved October 14, 2008, from Oracle BPA Suite: http://www.oracle.com/technology/products/bpa/ORACLE_BPA_SUITE_OVERVIEW_1206.pdf
- Oracle. (2008). *Oracle Business Process Architect Quick Start Guide, Release 10.1.3.4*.
- Ouyang, C., Aalst, W. M., Dumas, M., & Hofstede, A. H. (2006). Translating BPMN to BPEL. *BPM Center Report BPM-06-02, BPMcenter.org*.
- Ouyang, C., Dumas, M., Aalst, W. M., & Hofstede, A. H. (2006). *From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way*. BPMcenter.org.
- Ouyang, C., Dumas, M., Breutel, S., & Hofstede, A. H. (2006). Translating Standard Process Models to BPEL. In E. Dubois, & K. Pohl (Ed.), *Proceedings of the 18th International Conference on Advance Information Systems (CAiSE'06)* (pp. 417-432). Luxembourg: Springer.

Ouyang, C., Verbeek, E. (., Aalst, W. M., Breutel, S., Dumas, M., & Hofstede, A. H. (2007). Formal semantics and analysis of control flow in WS-BPEL. *Science of Computer Programming* , 67 (2-3), 162-198.

Parsons, J., & Cole, L. (2005). What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. *Data & Knowledge Engineering* , 55 (3), 327-342.

Peppard, J., & Ward, J. (1999). 'Mind the Gap': diagnosing the relationship between the IT organisation and the rest of the business. *Journal of Strategic Information Systems* , 8 (1), 29-60.

Recker, J., & Mendling, J. (2006). On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. *Proceedings 18th International Conference on Advanced Information Systems Engineering* , 521-532.

Recker, J., Rosemann, M., Indulska, M., & Green, P. (2006). *Business process modeling: A maturing discipline*. BPMcenter.org.

Reichert, M. U., & Rinderle, S. B. (2006). On Design Principles for Realizing Adaptive Service Flows with BPEL. *Proceedings of EMISA 2006* (pp. 133-146). Hamburg, Germany: Köllen Verlag.

Reichert, M., Rinderle, S., & Dadam, P. (2004). On the Modeling of Correct Service Flows with BPEL4WS. *Proceedings of EMISA 2004* (pp. 117-128). Köllen Verlag.

Rittgen, P. (1999). *Modified EPCs and Their Formal Semantics. Technical report 99/19*. Koblenz, germany: University of Koblenz-Landau.

Rosemann, M., Recker, J., Indulska, M., & Green, P. (2006). A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. In E. Dubois, & K. Pohl (Ed.), *Proceedings of the 18th Conference of Advanced Information Systems Engineering - CAiSE 2006* (pp. 447-461). Luxembourg: Springer.

Russel, N., Aalst, W. M., & Hofstede, A. H. (2006). Workflow Exception Patterns. *Proceedings of the 18th Conference on Advanced Information Systems Engineering (CAiSE'2006)* (pp. 288-302). Berlin / Heidelberg: Springer.

Russell, N., Aalst, W. M., Hofstede, A. H., & Edmond, D. (2004). Workflow Resource Patterns: Identification, Representation and Tool Support. *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)* (pp. 216-232). Berlin / Heidelberg: Springer.

Russell, N., Hofstede, A. H., Aalst, W. M., & Mulyar, N. A. (2006). *Workflow control-flow patterns: A revised view*. BPMcenter.org.

Russell, N., Hofstede, A. H., Edmond, D., & Aalst, W. M. (2005). Workflow Data Patterns: Identification, Representation and Tool Support. *Proceedings of the 24th International Conference on Conceptual Modeling (ER05)* (pp. 353-368). Berlin / Heidelberg: Springer.

Scheer, A.-W. (1994). *Business Process Engineering, ARIS-Navigator for Reference Models for Industrial Enterprises*. Berlin: Springer-Verlag.

Scheer, A.-W., & Schneider, K. (1992). *ARIS: Architecture of Integrated Information Systems*. Springer.

Scheithauer, G., & Wirtz, G. (2008). Case Study: Applying Business Process Management Systems. *Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE'2008)* (pp. 12-15). San Fransisco, USA: Knowledge Systems Institute Graduate School.

Sendall, S., & Kozaczynski, W. (2003). Model Transformation: The Heart and Soul of Model-Driven Software Development. *IEEE Software* , 20 (5), 42-45.

Sendall, S., & Küster, J. (2004). Taming Model Round-Trip Engineering. *OOPSLA/GPCE: Best Practices for Model-Driven Software Development* .

Shanks, G., Tansley, E., & Weber, R. (2003). Using ontology to validate conceptual models. *Communications of the ACM* , 46 (10), 85-89.

Shapiro, R. (2002). *A Comparison of XPD, BPML and BPEL4WS*. ebPML.org.

Silver, B. (2008). *The BPMS Report: Oracle BPM Solution v10.1.3*. Aptos, USA: Bruce Silver Associates.

Smith, H., & Fingar, P. (2003). *Business Process Management: The Third Wave*. Tampa, USA: Meghan-Kiffer.

Soanes, C., & Hawker, S. (2005). *Compact Oxford English Dictionary of Current English* (3rd ed.). Oxford, UK: Oxford University Press.

Söderström, E., Andersson, B., Johannesson, P., Perjons, E., & Wangler, B. (2002). Towards a Framework for Comparing Process Modelling Languages. *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002)* (pp. 600-611). Berlin / Heidelberg: Springer-Verlag.

Srikarsemsira, W., & Roongruangsuwan, S. (2005). Integration of BPEL Designer in Specific - Vendor UML Modeling Tool. *Proceedings of the Fourth International Conference on eBusiness*, (pp. 22.1-5). Bangkok, Thailand.

Stein, S., & Ivanov, K. (2007). EPK nach BPEL Transformation als Voraussetzung für praktische Umsetzung einer SOA. *Software Engineering*, 105, 75-80.

The Standish Group. (2007). *CHAOS Report 2007: The Laws of CHAOS*. West Yarmouth, USA: The Standish Group International.

The Standish Group. (1995). *The CHAOS Report*. West Yarmouth, USA: The Standish Group International.

Verschuren, P., & Doorewaard, H. (1998). *Het ontwerpen van een onderzoek* (2nd ed.). Utrecht, The Netherlands: LEMMA.

Vries, K. d., & Ommert, O. (2001). Advanced Workflow Patterns in Practice (1): Experiences Based on Pension Processing. *Business Process Magazine*, 7 (6), 15-18.

Vries, K. d., & Ommert, O. (2002). Advanced Workflow Patterns in Practice (2): Experiences Based on Judicial Processes. *Business Process Magazine*, 8 (1), 20-23.

Wagter, R., Van den Berg, M., Luijpers, J., & Van Steenberghe, M. (2001). *DYA® : snelheid en samenhang in business- en ICT-architectuur*. Den Bosch: Tutein Nolthenius.

Wand, Y., & Weber, R. (1989). An Ontological Evaluation of Systems Analysis and Design Methods. In E. Falkenberg, & P. Lindgreen (Ed.), *Information System Concepts: An In-depth Analysis. Proceedings of the IFIP TC 8/WG 8.1 Working Conference on Information System Concepts*, (pp. 79-107). Amsterdam, The Netherlands.

- Wand, Y., & Weber, R. (1990). An Ontological Model of an Information System. *IEEE Transactions on Software Engineering* , 16, 1282-1292.
- Wand, Y., & Weber, R. (1990). Mario Bunge's Ontology as a Formal Foundation for Information Systems Concepts. In G. D. Paul Weingartner, *Studies on Mario Bunge's Treatise* (pp. 132-149). Atlanta: Rodopi.
- Wand, Y., & Weber, R. (2006). On Ontological Foundations of Conceptual Modeling: A Response to Wysusek. (O. Henfridsson, K. Kautz, B. E. Munkvold, & M. Rossi, Eds.) *Scandinavian Journal of Information Systems* , 18 (1), 127-138.
- Wand, Y., & Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modeling—A Research Agenda. *Information Systems Research* , 13 (4), 363-376.
- Weber, R. (2003). Conceptual Modeling and Ontology: Possibilities and Pitfalls. *Journal of Database Management* , 14 (3), 1-20.
- White, S. (2005). Using BPMN to Model a BPEL Process. *BPTrends* , 3 (3), 1-18.
- Wohed, P., Aalst, W. M., Dumas, M., & Hofstede, A. H. (2003). Analysis of Web Services Composition Languages: The Case of BPEL4WS. *Proceedings of the 22nd International Conference on Conceptual Modeling* (pp. 200-215). Chicago, USA: Springer.
- Wohed, P., Aalst, W. M., Dumas, M., Hofstede, A. H., & Russell, N. (2006). On the Suitability of BPMN for Business Process Modelling. In S. Dustdar, J. L. Fiadeiro, & A. Sheth (Ed.), *Proceedings of the 4th International Conference on Business process Management (BPM 2006)* (pp. 161-176). Berlin / Heidelberg: Springer.
- Wohed, P., Aalst, W. M., Dumas, M., Hofstede, A. H., & Russell, N. (2005). Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams. In L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, & O. Pastor (Ed.), *Proceedings of the 24th International Conference on Conceptual Modeling (ER05)* (pp. 63-78). Berlin / Heidelberg: Springer-Verlag.
- Wysusek, B. (2006). On Ontological Foundations of Conceptual Modelling. (O. Henfridsson, K. Kautz, B. E. Munkvold, & M. Rossi, Eds.) *Scandinavian Journal of Information Systems* , 18 (1), 63-80.

Zhao, W., Hauser, R., Bhattacharya, K., Bryant, B., & Cao, F. (2006). Compiling business processes: untangling unstructured loops in irreducible flow graphs. *International Journal of Web and Grid Services*, 2 (1), 68-91.

Ziemann, J., & Mendling, J. (2005). EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach. *Proceedings of the 7th International Conference on the Modern Information Technology in the Innovation Processes of the industrial enterprises*. Genoa, Italy.

Appendix A - Output BPEL diagrams

Diagram 1: WFCP 01 - Sequence (expanded)

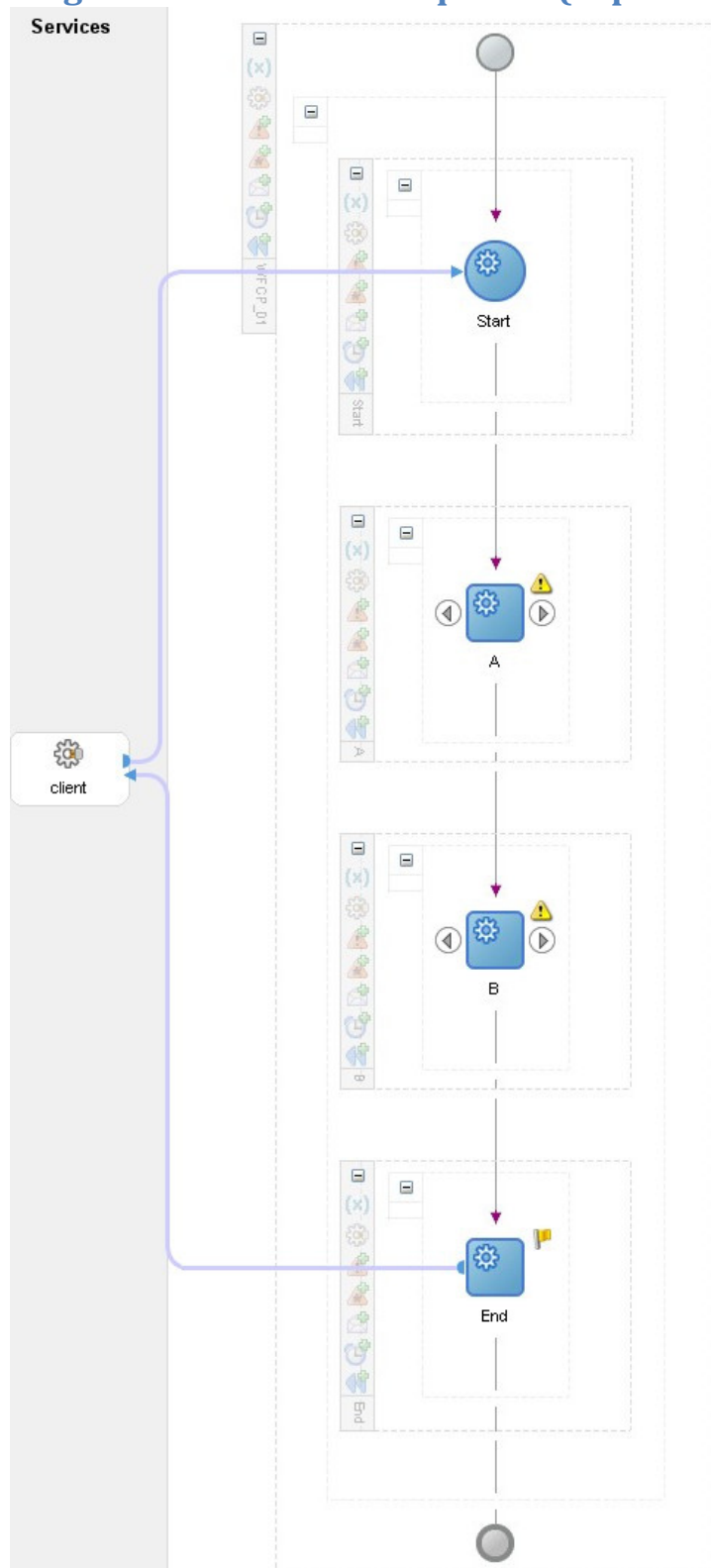


Diagram 1: WFCP 01 – Sequence (collapsed)

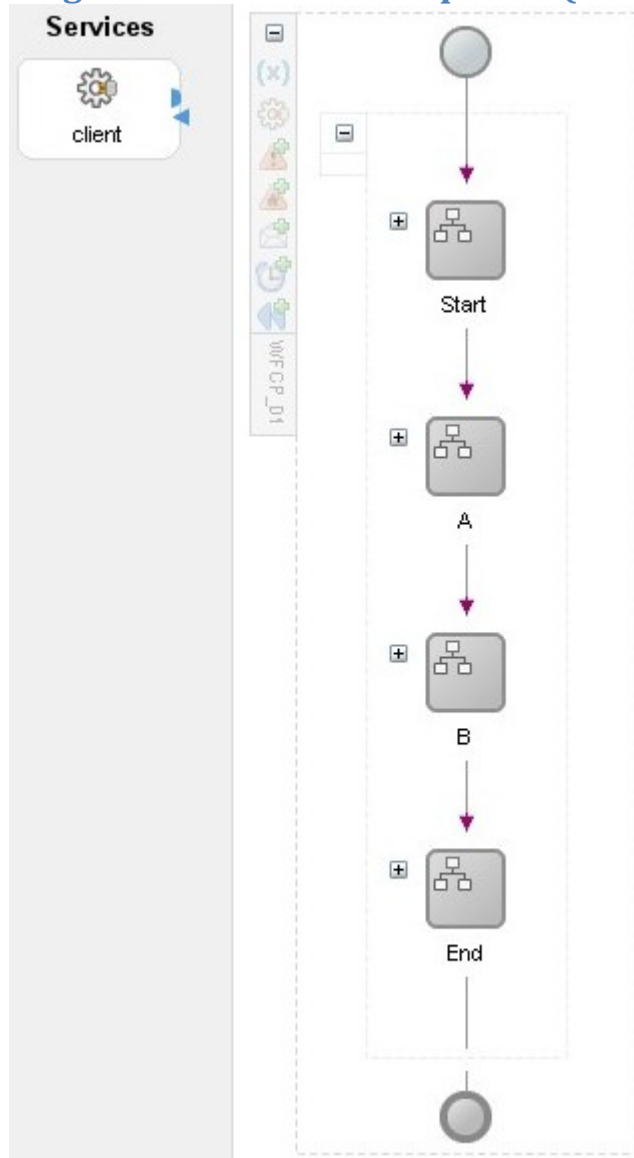


Diagram 2: WFCP 02 – Parallel Split & WFCP 03 – Synchronization (expanded)

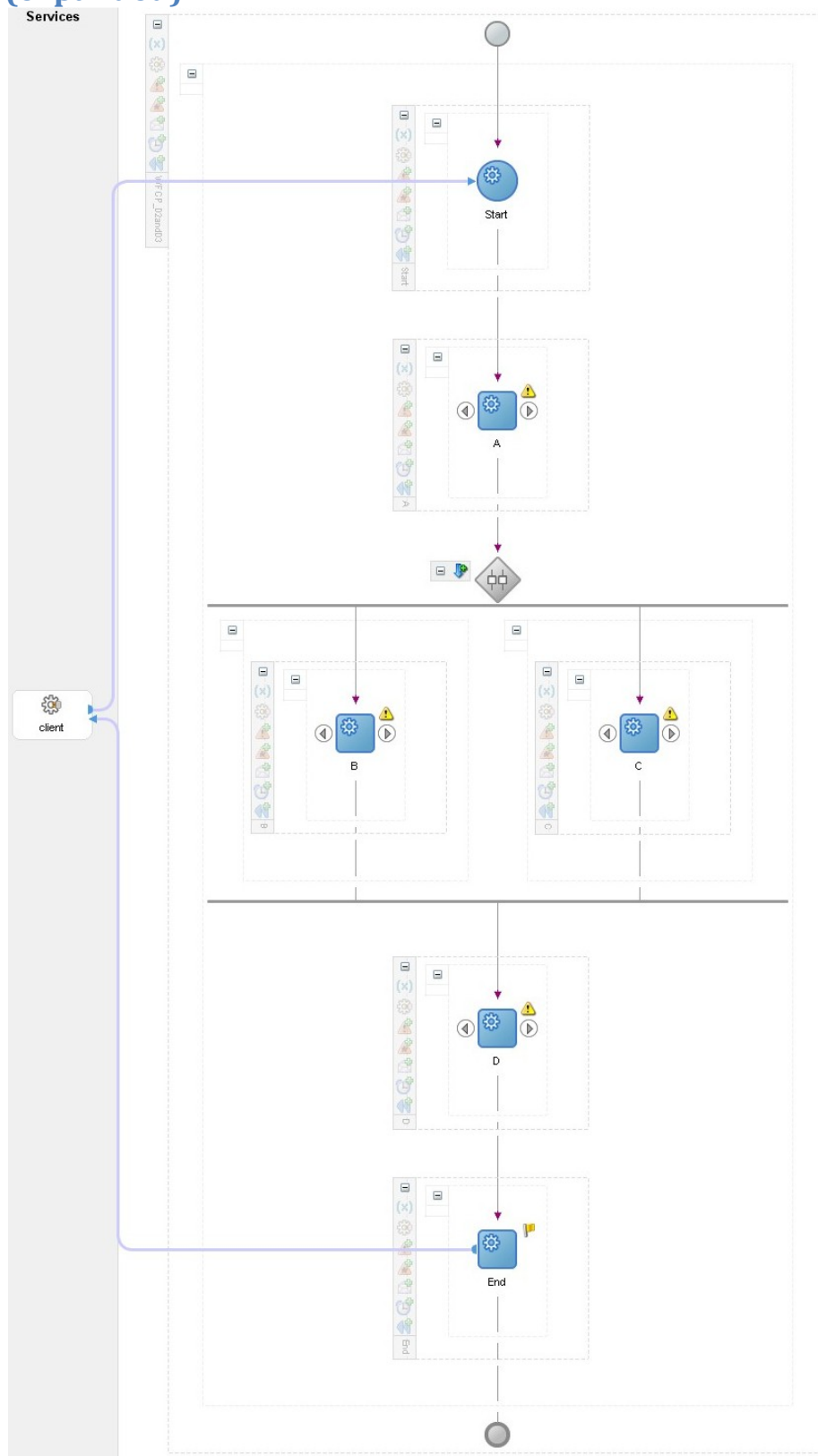


Diagram 2: WFCP 02 – Parallel Split & WFCP 03 – Synchronization (collapsed)

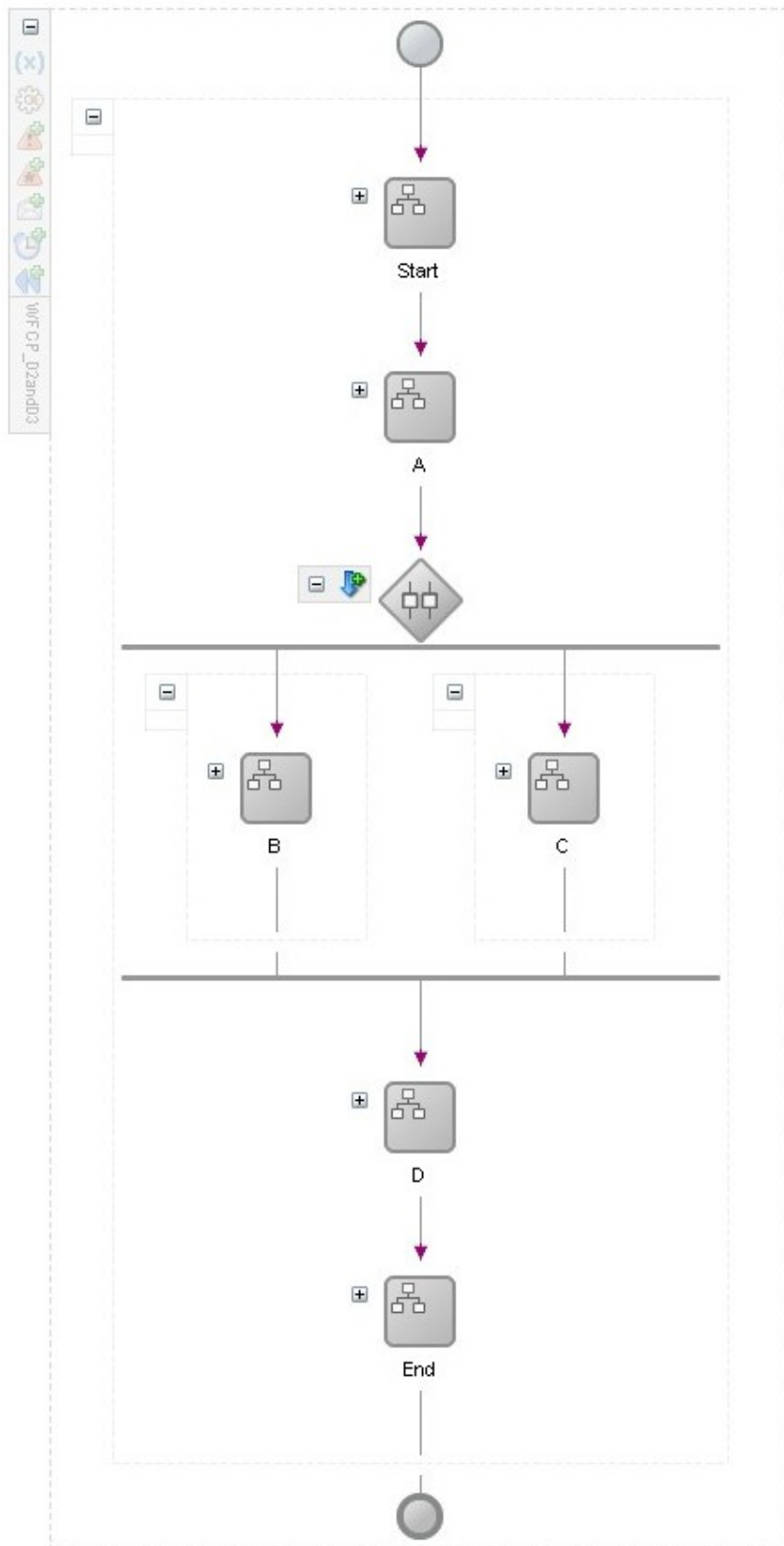


Diagram 3: WFCP 04 – Exclusive Choice & WFCP 05 – Simple Merge (expanded)

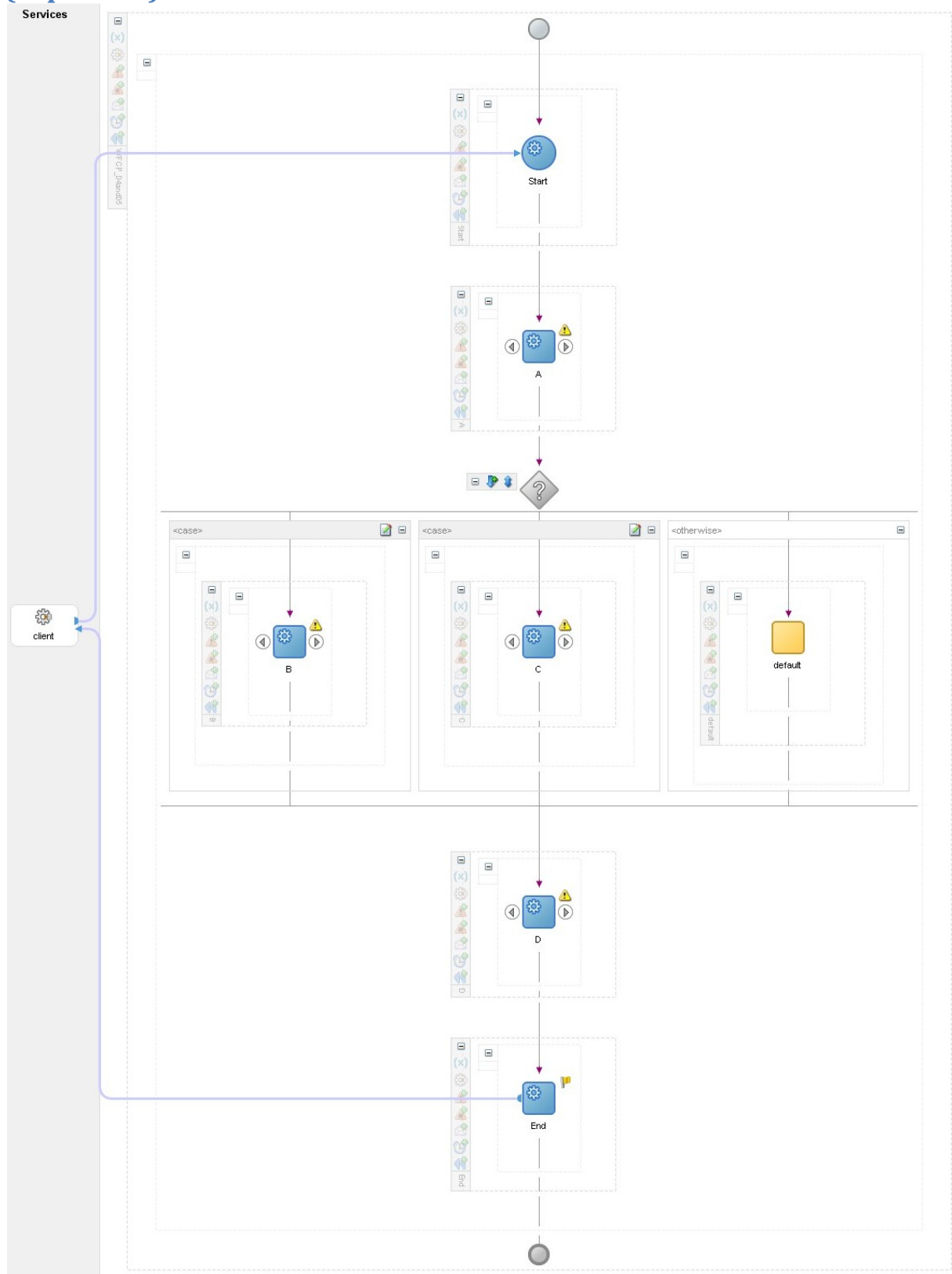


Diagram 3: WFCP 04 – Exclusive Choice & WFCP 05 – Simple Merge (collapsed)

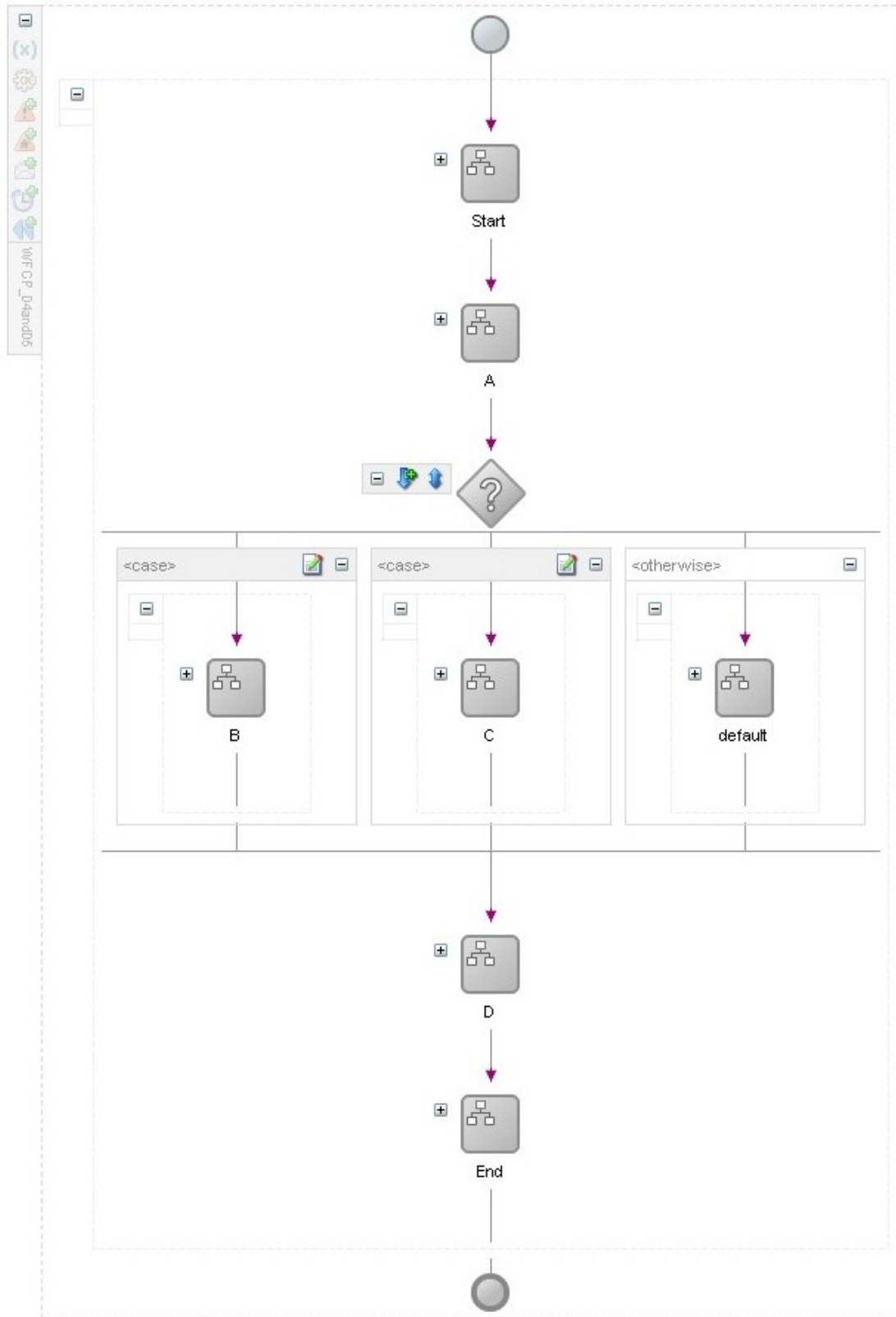


Diagram 5: WFCP 11 - Implicit Termination (expanded)

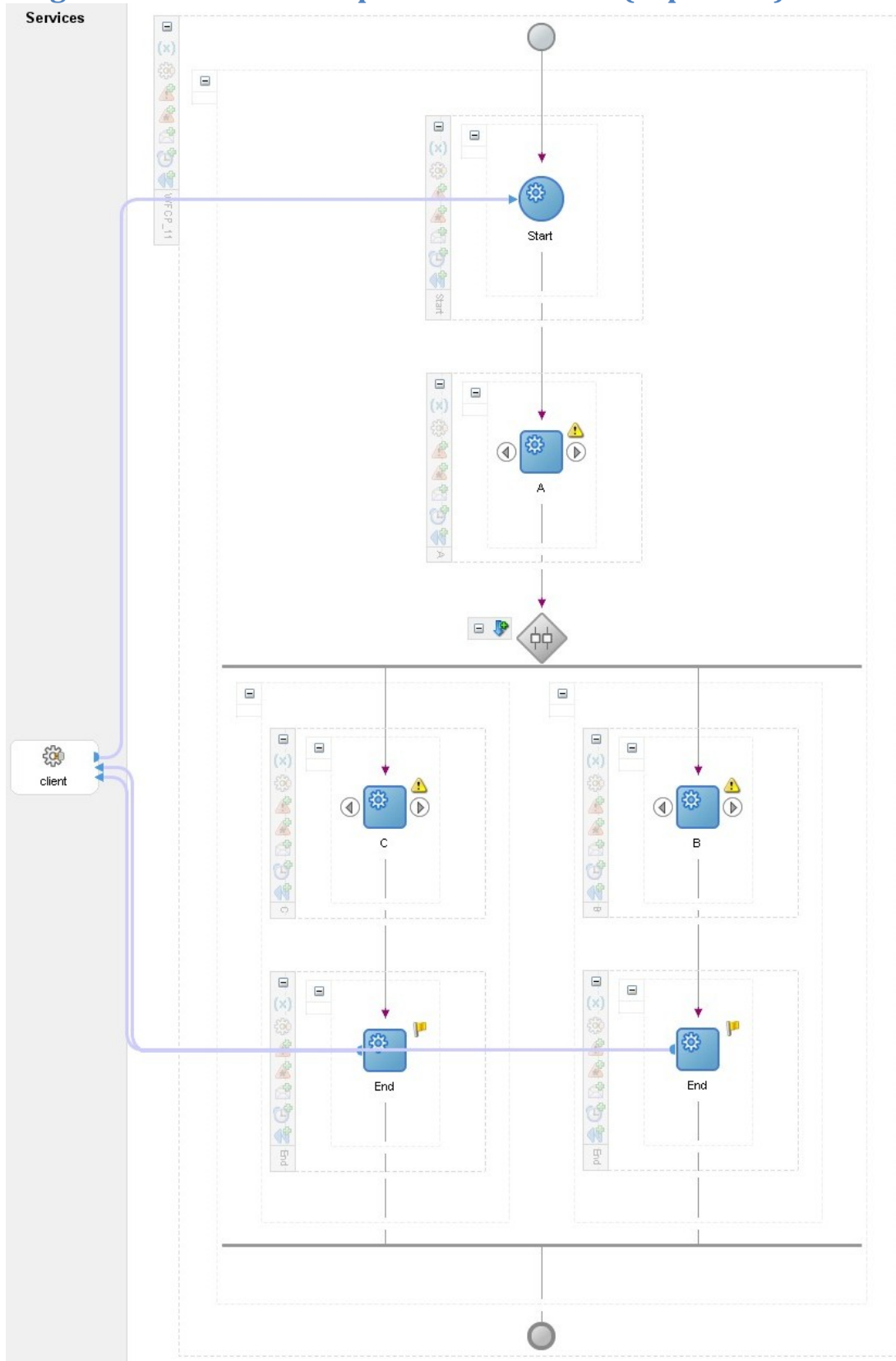


Diagram 5: WFCP 11 - Implicit Termination (collapsed)

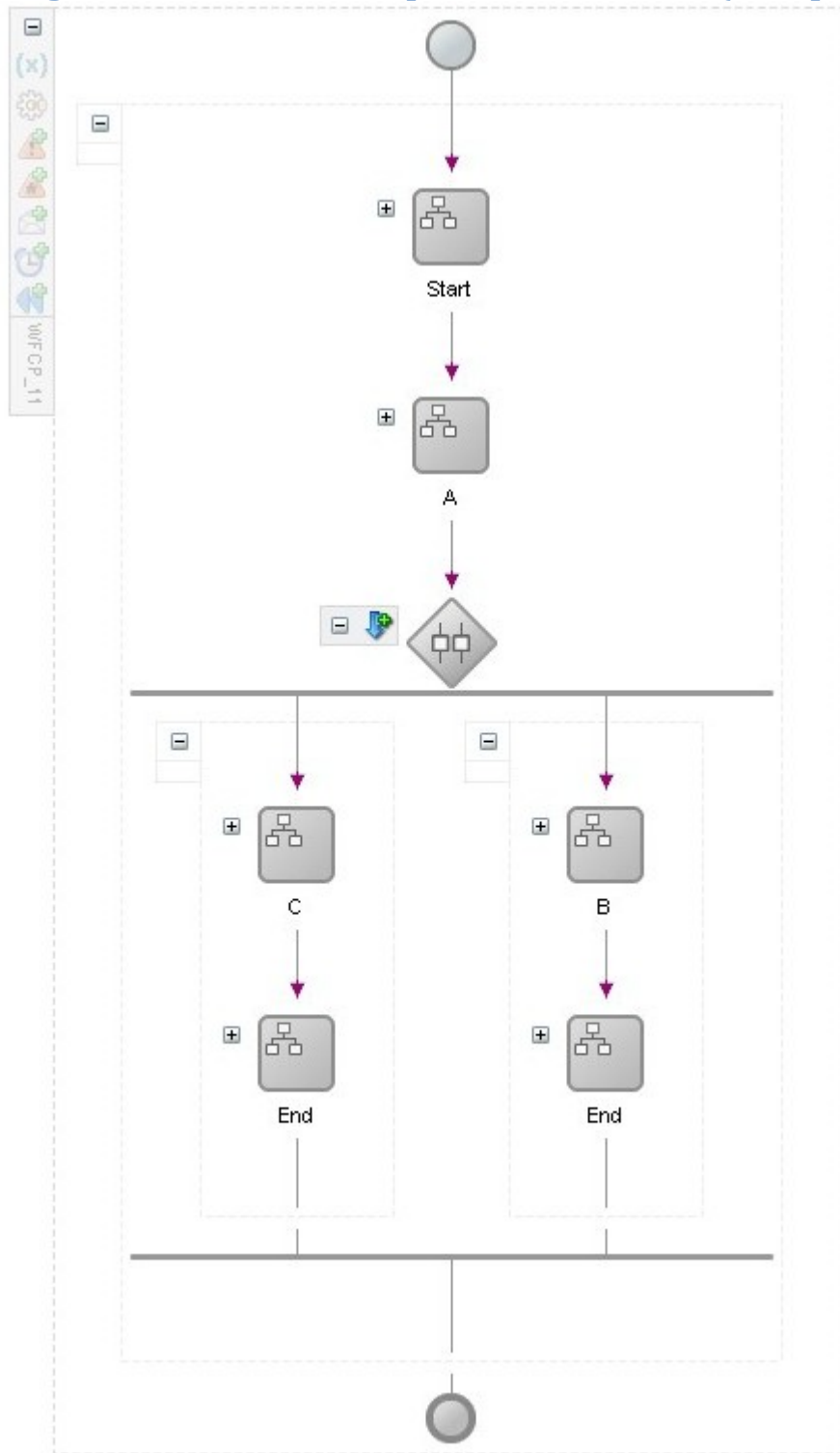


Diagram 6: WFCP 11 - Arbitrary Cycle (expanded)

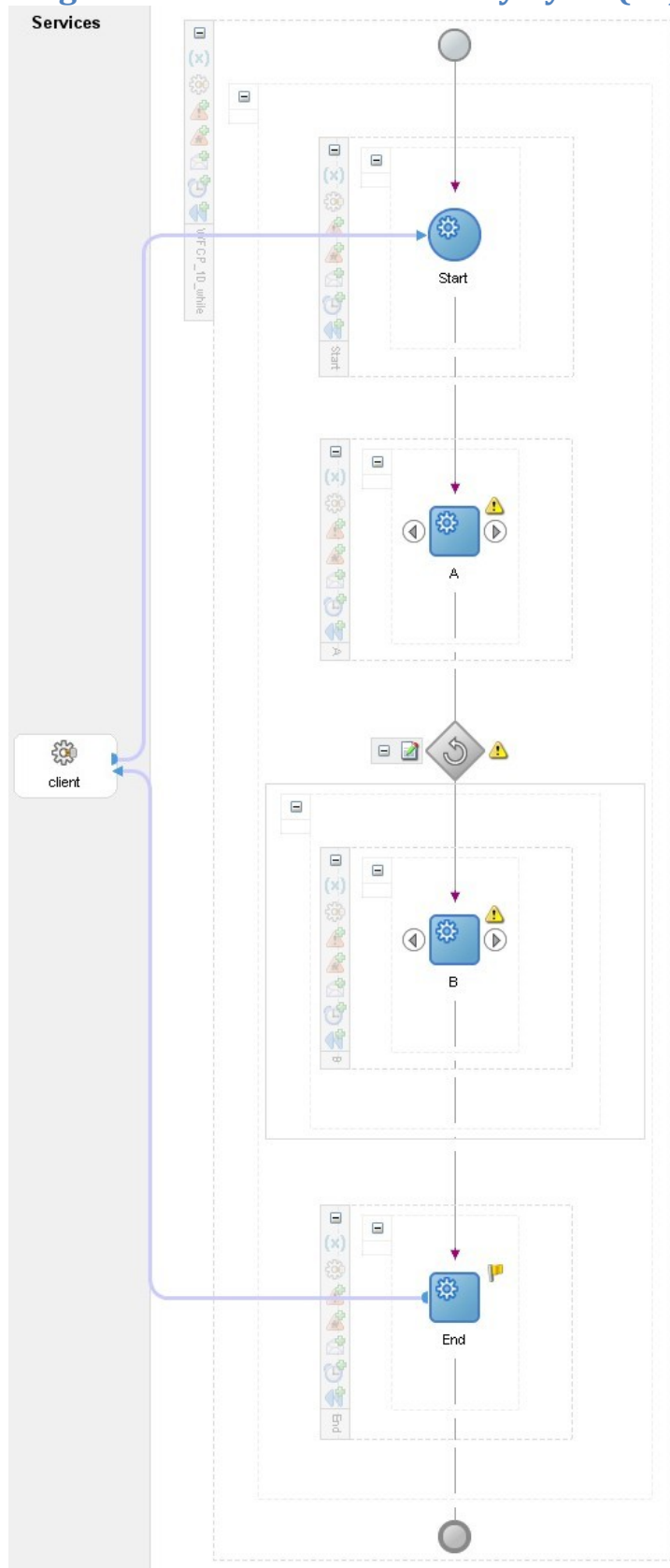
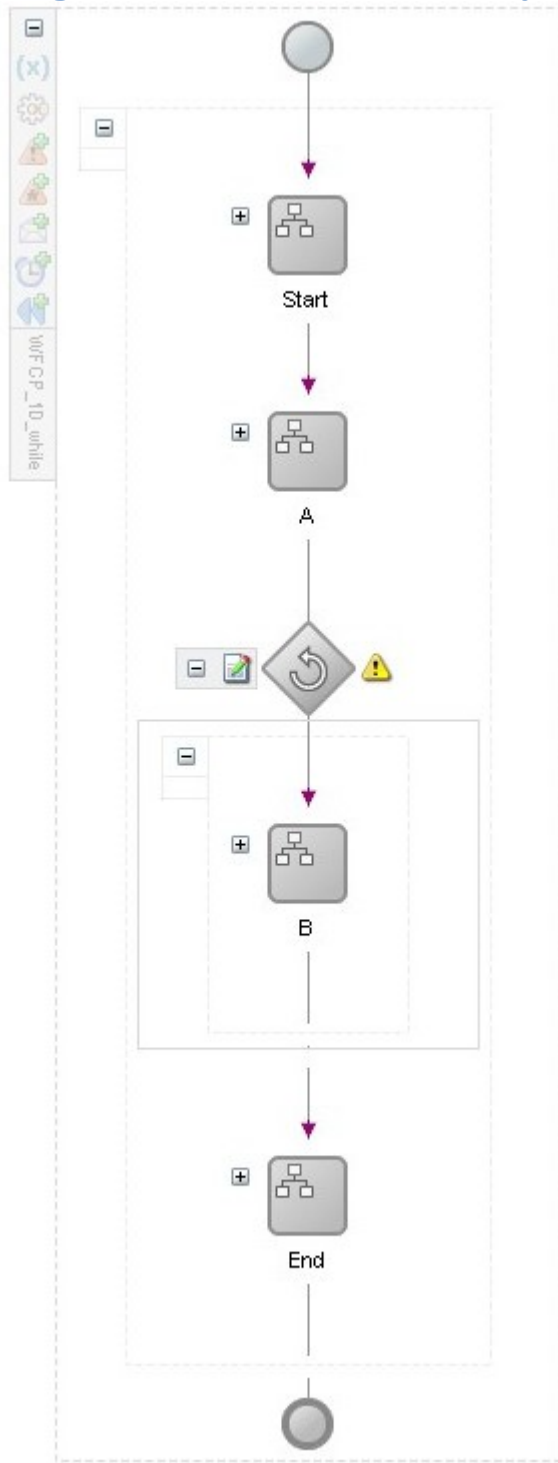


Diagram 6: WFCP 11 - Arbitrary Cycle (collapsed)



Appendix B - Output BPEL code

Diagram 1: WFCP 01 - Sequence

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--Generated by Oracle BPA Suite-->
3  <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:tns="http://xmlns.oracle.com/WFCP_01"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
name="WFCP_01" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://xmlns.oracle.com/WFCP_01">
4    <bpelx:annotation>
5      <bpelx:pattern patternName="bpelx:generated"/>
6      <bpelx:analysis>
7        <bpelx:property name="Description"></bpelx:property>
8        <bpelx:property name="BusinessProcessIdentifier">7400f9c0-c20c-
11dd-23b1-005056c00001</bpelx:property>
9        <bpelx:property name="LastUpdateDate">12/9/08 6:12:22
PM</bpelx:property>
10       <bpelx:property name="BPELProcessIdentifier">862a7d41-c614-11dd-
23b1-005056c00001</bpelx:property>
11       <bpelx:property name="Filter">dd838074-ac29-11d4-85b8-
00005a4053ff</bpelx:property>
12     </bpelx:analysis>
13   </bpelx:annotation>
14   <partnerLinks>
15     <partnerLink myRole="WFCP_01Provider" name="client"
partnerLinkType="tns:WFCP_01" partnerRole="WFCP_01Requester"/>
16   </partnerLinks>
17   <variables>
18     <variable messageType="tns:WFCP_01RequestMessage"
name="inputVariable"/>
19     <variable messageType="tns:WFCP_01ResponseMessage"
name="outputVariable"/>
20   </variables>
21   <sequence>
22     <bpelx:annotation>
23       <bpelx:analysis>
24         <bpelx:property name="BusinessId">Sequence_7400f9c0-c20c-11dd-
23b1-005056c00001</bpelx:property>
25         <bpelx:property name="LastUpdateDate">12/9/08 6:12:17
PM</bpelx:property>
26       </bpelx:analysis>
27     </bpelx:annotation>
28     <scope name="Start">
29       <bpelx:annotation>
```

```

30         <bpelx:pattern patternName="bpelx:StartEvent"/>
31         <bpelx:analysis>
32             <bpelx:property name="BusinessId">Scope_7cd19910-c20c-11dd-
23b1-005056c00001</bpelx:property>
33             <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
34         </bpelx:analysis>
35     </bpelx:annotation>
36     <sequence>
37         <bpelx:annotation>
38             <bpelx:analysis>
39                 <bpelx:property name="BusinessId">Sequence_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
40                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
41                 </bpelx:analysis>
42             </bpelx:annotation>
43             <receive createInstance="yes" name="Start"
operation="initiate" partnerLink="client" portType="tns:WFCP_01"
variable="inputVariable">
44                 <bpelx:annotation>
45                     <bpelx:documentation></bpelx:documentation>
46                     <bpelx:analysis>
47                         <bpelx:property
name="CreateInstance">>true</bpelx:property>
48                         <bpelx:property name="BusinessId">Receive_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
49                         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
50                             <bpelx:property name="Documentation"></bpelx:property>
51                             <bpelx:property name="Label">Start</bpelx:property>
52                         </bpelx:analysis>
53                     </bpelx:annotation>
54                 </receive>
55             </sequence>
56         </scope>
57     <scope name="A">
58         <bpelx:annotation>
59             <bpelx:pattern patternName="bpelx:automated"/>
60             <bpelx:analysis>
61                 <bpelx:property name="BusinessId">Scope_7cd19911-c20c-11dd-
23b1-005056c00001</bpelx:property>
62                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
63                 </bpelx:analysis>
64             </bpelx:annotation>
65         <sequence>

```

```

66         <bpelx:annotation>
67         <bpelx:analysis>
68         <bpelx:property name="BusinessId">Sequence_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
69         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
70         </bpelx:analysis>
71         </bpelx:annotation>
72         <invoke name="A">
73         <bpelx:annotation>
74         <bpelx:documentation></bpelx:documentation>
75         <bpelx:analysis>
76         <bpelx:property name="Documentation"></bpelx:property>
77         <bpelx:property name="BusinessId">Invoke_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
78         <bpelx:property name="Label">A</bpelx:property>
79         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
80         </bpelx:analysis>
81         </bpelx:annotation>
82         </invoke>
83     </sequence>
84 </scope>
85 <scope name="B">
86     <bpelx:annotation>
87     <bpelx:pattern patternName="bpelx:automated"/>
88     <bpelx:analysis>
89     <bpelx:property name="BusinessId">Scope_7cd19915-c20c-11dd-
23b1-005056c00001</bpelx:property>
90     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
91     </bpelx:analysis>
92     </bpelx:annotation>
93     <sequence>
94     <bpelx:annotation>
95     <bpelx:analysis>
96     <bpelx:property name="BusinessId">Sequence_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>
97     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
98     </bpelx:analysis>
99     </bpelx:annotation>
100    <invoke name="B">
101    <bpelx:annotation>
102    <bpelx:documentation></bpelx:documentation>
103    <bpelx:analysis>
104    <bpelx:property name="Documentation"></bpelx:property>

```

```

105         <bpelx:property name="BusinessId">Invoke_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>
106         <bpelx:property name="Label">B</bpelx:property>
107         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
108         </bpelx:analysis>
109         </bpelx:annotation>
110         </invoke>
111     </sequence>
112 </scope>
113 <scope name="End">
114     <bpelx:annotation>
115         <bpelx:pattern patternName="bpelx:automated"/>
116     <bpelx:analysis>
117         <bpelx:property name="BusinessId">Scope_7cd19917-c20c-11dd-
23b1-005056c00001</bpelx:property>
118         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
119         </bpelx:analysis>
120     </bpelx:annotation>
121     <sequence>
122         <bpelx:annotation>
123             <bpelx:analysis>
124                 <bpelx:property name="BusinessId">Sequence_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
125                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
126                 </bpelx:analysis>
127             </bpelx:annotation>
128             <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client" portType="tns:WFCP_01Callback">
129                 <bpelx:annotation>
130                     <bpelx:documentation></bpelx:documentation>
131                 <bpelx:analysis>
132                     <bpelx:property name="Documentation"></bpelx:property>
133                     <bpelx:property name="BusinessId">Invoke_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
134                     <bpelx:property name="Label">End</bpelx:property>
135                     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
136                     </bpelx:analysis>
137                 </bpelx:annotation>
138             </invoke>
139         </sequence>
140     </scope>
141 </sequence>
142 </process>

```

Diagram 2: WFCP 02 – Parallel Split & WFCP 03 – Synchronization

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--Generated by Oracle BPA Suite-->
3  <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:tns="http://xmlns.oracle.com/WFCP_02and03"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
name="WFCP_02and03" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116" targetNamespace="http://xmlns.oracle.com/WFCP_02and03">
4    <bpelx:annotation>
5      <bpelx:pattern patternName="bpelx:generated"/>
6      <bpelx:analysis>
7        <bpelx:property name="Description"></bpelx:property>
8        <bpelx:property name="BusinessProcessIdentifier">7af06080-c20d-
11dd-23b1-005056c00001</bpelx:property>
9        <bpelx:property name="LastUpdateDate">12/9/08 6:13:16
PM</bpelx:property>
10       <bpelx:property name="BPELProcessIdentifier">a78c0583-c614-11dd-
23b1-005056c00001</bpelx:property>
11       <bpelx:property name="Filter">dd838074-ac29-11d4-85b8-
00005a4053ff</bpelx:property>
12     </bpelx:analysis>
13   </bpelx:annotation>
14   <partnerLinks>
15     <partnerLink myRole="WFCP_02and03Provider" name="client"
partnerLinkType="tns:WFCP_02and03" partnerRole="WFCP_02and03Requester"/>
16   </partnerLinks>
17   <variables>
18     <variable messageType="tns:WFCP_02and03ResponseMessage"
name="outputVariable"/>
19     <variable messageType="tns:WFCP_02and03RequestMessage"
name="inputVariable"/>
20   </variables>
21   <sequence>
22     <bpelx:annotation>
23       <bpelx:analysis>
24         <bpelx:property name="BusinessId">Sequence_7af06080-c20d-11dd-
23b1-005056c00001</bpelx:property>
25         <bpelx:property name="LastUpdateDate">12/9/08 6:13:11
PM</bpelx:property>
26       </bpelx:analysis>
27     </bpelx:annotation>
28     <scope name="Start">
29       <bpelx:annotation>
30         <bpelx:pattern patternName="bpelx:StartEvent"/>
31       </bpelx:annotation>

```



```

32         <bpelx:property name="BusinessId">Scope_7cd19910-c20c-11dd-
23b1-005056c00001</bpelx:property>
33         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
34     </bpelx:analysis>
35 </bpelx:annotation>
36 <sequence>
37     <bpelx:annotation>
38     <bpelx:analysis>
39         <bpelx:property name="BusinessId">Sequence_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
40         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
41     </bpelx:analysis>
42 </bpelx:annotation>
43 <receive createInstance="yes" name="Start "
operation="initiate" partnerLink="client" portType="tns:WFCP_02and03"
variable="inputVariable">
44     <bpelx:annotation>
45     <bpelx:documentation></bpelx:documentation>
46     <bpelx:analysis>
47     <bpelx:property name="BusinessId">Receive_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
48     <bpelx:property
name="CreateInstance">true</bpelx:property>
49     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
50     <bpelx:property name="Documentation"></bpelx:property>
51     <bpelx:property name="Label">Start</bpelx:property>
52 </bpelx:analysis>
53 </bpelx:annotation>
54 </receive>
55 </sequence>
56 </scope>
57 <scope name="A">
58     <bpelx:annotation>
59     <bpelx:pattern patternName="bpelx:automated"/>
60     <bpelx:analysis>
61     <bpelx:property name="BusinessId">Scope_7cd19911-c20c-11dd-
23b1-005056c00001</bpelx:property>
62     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
63     </bpelx:analysis>
64 </bpelx:annotation>
65 <sequence>
66     <bpelx:annotation>
67     <bpelx:analysis>

```

```

68         <bpelx:property name="BusinessId">Sequence_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
69         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
70         </bpelx:analysis>
71     </bpelx:annotation>
72     <invoke name="A">
73         <bpelx:annotation>
74             <bpelx:documentation></bpelx:documentation>
75             <bpelx:analysis>
76                 <bpelx:property name="Documentation"></bpelx:property>
77                 <bpelx:property name="BusinessId">Invoke_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
78                 <bpelx:property name="Label">A</bpelx:property>
79                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
80             </bpelx:analysis>
81         </bpelx:annotation>
82     </invoke>
83 </sequence>
84 </scope>
85 <flow name="AND_rule">
86     <bpelx:annotation>
87         <bpelx:pattern patternName="Flow"/>
88     <bpelx:analysis>
89         <bpelx:property name="BusinessId">Flow_7cd1991d-c20c-11dd-
23b1-005056c00001</bpelx:property>
90         <bpelx:property name="Label">AND_rule</bpelx:property>
91         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
92     </bpelx:analysis>
93 </bpelx:annotation>
94 <sequence>
95     <bpelx:annotation>
96         <bpelx:analysis>
97             <bpelx:property name="BusinessId">Sequence_7cd19927-c20c-
11dd-23b1-005056c00001</bpelx:property>
98             <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
99         </bpelx:analysis>
100    </bpelx:annotation>
101    <scope name="B">
102        <bpelx:annotation>
103            <bpelx:pattern patternName="bpelx:automated"/>
104        <bpelx:analysis>
105            <bpelx:property name="BusinessId">Scope_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>

```

```

106         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
107         </bpelx:analysis>
108     </bpelx:annotation>
109     <sequence>
110         <bpelx:annotation>
111             <bpelx:analysis>
112                 <bpelx:property name="BusinessId">Sequence_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
113                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
114                 </bpelx:analysis>
115             </bpelx:annotation>
116             <invoke name="B">
117                 <bpelx:annotation>
118                     <bpelx:documentation></bpelx:documentation>
119                 </bpelx:analysis>
120                 <bpelx:property
name="Documentation"></bpelx:property>
121                 <bpelx:property name="BusinessId">Invoke_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
122                 <bpelx:property name="Label">B</bpelx:property>
123                 <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
124                 </bpelx:analysis>
125             </bpelx:annotation>
126         </invoke>
127     </sequence>
128 </scope>
129 </sequence>
130 <sequence>
131     <bpelx:annotation>
132         <bpelx:analysis>
133             <bpelx:property name="BusinessId">Sequence_7cd19934-c20c-
11dd-23b1-005056c00001</bpelx:property>
134             <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
135             </bpelx:analysis>
136         </bpelx:annotation>
137         <scope name="C">
138             <bpelx:annotation>
139                 <bpelx:pattern patternName="bpelx:automated"/>
140             <bpelx:analysis>
141                 <bpelx:property name="BusinessId">Scope_7cd19931-c20c-
11dd-23b1-005056c00001</bpelx:property>
142                 <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>

```

```

143         </bpelx:analysis>
144     </bpelx:annotation>
145     <sequence>
146         <bpelx:annotation>
147             <bpelx:analysis>
148                 <bpelx:property name="BusinessId">Sequence_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
149                 <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
150             </bpelx:analysis>
151         </bpelx:annotation>
152         <invoke name="C">
153             <bpelx:annotation>
154                 <bpelx:documentation></bpelx:documentation>
155             <bpelx:analysis>
156                 <bpelx:property
name="Documentation"></bpelx:property>
157                 <bpelx:property name="BusinessId">Invoke_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
158                 <bpelx:property name="Label">C</bpelx:property>
159                 <bpelx:property name="LastUpdateDate">12/4/08
3:16:49 PM</bpelx:property>
160             </bpelx:analysis>
161         </bpelx:annotation>
162     </invoke>
163 </sequence>
164 </scope>
165 </sequence>
166 </flow>
167 <scope name="D">
168     <bpelx:annotation>
169         <bpelx:pattern patternName="bpelx:automated"/>
170     <bpelx:analysis>
171         <bpelx:property name="BusinessId">Scope_7cd1992a-c20c-11dd-
23b1-005056c00001</bpelx:property>
172         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
173     </bpelx:analysis>
174 </bpelx:annotation>
175 <sequence>
176     <bpelx:annotation>
177         <bpelx:analysis>
178             <bpelx:property name="BusinessId">Sequence_7cd1992a-c20c-
11dd-23b1-005056c00001</bpelx:property>
179             <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
180         </bpelx:analysis>

```

```

181         </bpelx:annotation>
182         <invoke name="D">
183             <bpelx:annotation>
184                 <bpelx:documentation></bpelx:documentation>
185                 <bpelx:analysis>
186                     <bpelx:property name="Documentation"></bpelx:property>
187                     <bpelx:property name="BusinessId">Invoke_7cd1992a-c20c-
11dd-23b1-005056c00001</bpelx:property>
188                     <bpelx:property name="Label">D</bpelx:property>
189                     <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
190                 </bpelx:analysis>
191             </bpelx:annotation>
192         </invoke>
193     </sequence>
194 </scope>
195 <scope name="End">
196     <bpelx:annotation>
197         <bpelx:pattern patternName="bpelx:automated"/>
198     <bpelx:analysis>
199         <bpelx:property name="BusinessId">Scope_7cd19917-c20c-11dd-
23b1-005056c00001</bpelx:property>
200         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
201     </bpelx:analysis>
202 </bpelx:annotation>
203 <sequence>
204     <bpelx:annotation>
205     <bpelx:analysis>
206         <bpelx:property name="BusinessId">Sequence_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
207         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
208     </bpelx:analysis>
209 </bpelx:annotation>
210     <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client"
portType="tns:WFCP_02and03Callback">
211         <bpelx:annotation>
212             <bpelx:documentation></bpelx:documentation>
213             <bpelx:analysis>
214                 <bpelx:property name="Documentation"></bpelx:property>
215                 <bpelx:property name="BusinessId">Invoke_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
216                 <bpelx:property name="Label">End</bpelx:property>
217                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>

```

```

218         </bpelx:analysis>
219         </bpelx:annotation>
220     </invoke>
221 </sequence>
222 </scope>
223 </sequence>
224 </process>

```

Diagram 3: WFCP 04 – Exclusive Choice & WFCP 05 – Simple Merge

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--Generated by Oracle BPA Suite-->
3  <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:tns="http://xmlns.oracle.com/WFCP_04and05"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
name="WFCP_04and05" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116" targetNamespace="http://xmlns.oracle.com/WFCP_04and05">
4      <bpelx:annotation>
5          <bpelx:pattern patternName="bpelx:generated"/>
6          <bpelx:analysis>
7              <bpelx:property name="Description"></bpelx:property>
8              <bpelx:property name="BusinessProcessIdentifier">40350490-c20e-
11dd-23b1-005056c00001</bpelx:property>
9              <bpelx:property name="LastUpdateDate">12/9/08 6:14:35
PM</bpelx:property>
10             <bpelx:property name="BPELProcessIdentifier">d758a7a1-c614-11dd-
23b1-005056c00001</bpelx:property>
11             <bpelx:property name="Filter">dd838074-ac29-11d4-85b8-
00005a4053ff</bpelx:property>
12         </bpelx:analysis>
13     </bpelx:annotation>
14     <partnerLinks>
15         <partnerLink myRole="WFCP_04and05Provider" name="client"
partnerLinkType="tns:WFCP_04and05" partnerRole="WFCP_04and05Requester"/>
16     </partnerLinks>
17     <variables>
18         <variable messageType="tns:WFCP_04and05ResponseMessage"
name="outputVariable"/>
19         <variable messageType="tns:WFCP_04and05RequestMessage"
name="inputVariable"/>
20     </variables>
21     <sequence>
22         <bpelx:annotation>
23             <bpelx:analysis>
24                 <bpelx:property name="BusinessId">Sequence_40350490-c20e-11dd-
23b1-005056c00001</bpelx:property>

```

```

25         <bpelx:property name="LastUpdateDate">12/9/08 6:14:31
PM</bpelx:property>
26         </bpelx:analysis>
27     </bpelx:annotation>
28     <scope name="Start">
29         <bpelx:annotation>
30             <bpelx:pattern patternName="bpelx:StartEvent"/>
31             <bpelx:analysis>
32                 <bpelx:property name="BusinessId">Scope_7cd19910-c20c-11dd-
23b1-005056c00001</bpelx:property>
33                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
34                 </bpelx:analysis>
35             </bpelx:annotation>
36         <sequence>
37             <bpelx:annotation>
38                 <bpelx:analysis>
39                     <bpelx:property name="BusinessId">Sequence_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
40                     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
41                     </bpelx:analysis>
42                 </bpelx:annotation>
43             <receive createInstance="yes" name="Start"
operation="initiate" partnerLink="client" portType="tns:WFCP_04and05"
variable="inputVariable">
44                 <bpelx:annotation>
45                     <bpelx:documentation></bpelx:documentation>
46                 <bpelx:analysis>
47                     <bpelx:property name="BusinessId">Receive_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
48                     <bpelx:property
name="CreateInstance">true</bpelx:property>
49                     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
50                     <bpelx:property name="Documentation"></bpelx:property>
51                     <bpelx:property name="Label">Start</bpelx:property>
52                 </bpelx:analysis>
53             </bpelx:annotation>
54         </receive>
55     </sequence>
56 </scope>
57 <scope name="A">
58     <bpelx:annotation>
59         <bpelx:pattern patternName="bpelx:automated"/>
60     <bpelx:analysis>

```

```

61         <bpelx:property name="BusinessId">Scope_7cd19911-c20c-11dd-
23b1-005056c00001</bpelx:property>
62         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
63         </bpelx:analysis>
64     </bpelx:annotation>
65 <sequence>
66     <bpelx:annotation>
67     <bpelx:analysis>
68         <bpelx:property name="BusinessId">Sequence_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
69         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
70         </bpelx:analysis>
71     </bpelx:annotation>
72 <invoke name="A">
73     <bpelx:annotation>
74     <bpelx:documentation></bpelx:documentation>
75     <bpelx:analysis>
76         <bpelx:property name="Documentation"></bpelx:property>
77         <bpelx:property name="BusinessId">Invoke_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
78         <bpelx:property name="Label">A</bpelx:property>
79         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
80         </bpelx:analysis>
81     </bpelx:annotation>
82 </invoke>
83 </sequence>
84 </scope>
85 <switch name="Switch">
86     <bpelx:annotation>
87     <bpelx:pattern patternName="XOR"/>
88     <bpelx:analysis>
89         <bpelx:property name="BusinessId">Switch_7cd1993a-c20c-11dd-
23b1-005056c00001</bpelx:property>
90         <bpelx:property name="Label">Switch</bpelx:property>
91         <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
92         </bpelx:analysis>
93     </bpelx:annotation>
94 <case condition="Event">
95     <bpelx:annotation>
96     <bpelx:pattern patternName="Case"/>
97     <bpelx:analysis>
98         <bpelx:property name="BusinessId">Case_573402e2-c20e-11dd-
23b1-005056c00001</bpelx:property>

```



```

99         <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
100         <bpelx:property
name="ConditionExpression">Event</bpelx:property>
101         </bpelx:analysis>
102     </bpelx:annotation>
103     <sequence>
104         <bpelx:annotation>
105             <bpelx:analysis>
106                 <bpelx:property name="BusinessId">Sequence_573402e2-
c20e-11dd-23b1-005056c00001</bpelx:property>
107                 <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
108             </bpelx:analysis>
109         </bpelx:annotation>
110     <scope name="B">
111         <bpelx:annotation>
112             <bpelx:pattern patternName="bpelx:automated"/>
113         <bpelx:analysis>
114             <bpelx:property name="BusinessId">Scope_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>
115             <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
116         </bpelx:analysis>
117     </bpelx:annotation>
118     <sequence>
119         <bpelx:annotation>
120             <bpelx:analysis>
121                 <bpelx:property name="BusinessId">Sequence_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
122                 <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
123             </bpelx:analysis>
124         </bpelx:annotation>
125     <invoke name="B">
126         <bpelx:annotation>
127             <bpelx:documentation></bpelx:documentation>
128         <bpelx:analysis>
129             <bpelx:property
name="Documentation"></bpelx:property>
130             <bpelx:property name="BusinessId">Invoke_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
131             <bpelx:property name="Label">B</bpelx:property>
132             <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
133         </bpelx:analysis>
134     </bpelx:annotation>

```

```

135         </invoke>
136     </sequence>
137 </scope>
138 </sequence>
139 </case>
140 <case condition="Event">
141     <bpelx:annotation>
142         <bpelx:pattern patternName="Case"/>
143         <bpelx:analysis>
144             <bpelx:property name="BusinessId">Case_573402f1-c20e-11dd-
23b1-005056c00001</bpelx:property>
145             <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
146             <bpelx:property
name="ConditionExpression">Event</bpelx:property>
147         </bpelx:analysis>
148     </bpelx:annotation>
149 <sequence>
150     <bpelx:annotation>
151     <bpelx:analysis>
152         <bpelx:property name="BusinessId">Sequence_573402f1-
c20e-11dd-23b1-005056c00001</bpelx:property>
153         <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
154     </bpelx:analysis>
155     </bpelx:annotation>
156 <scope name="C">
157     <bpelx:annotation>
158         <bpelx:pattern patternName="bpelx:automated"/>
159     <bpelx:analysis>
160         <bpelx:property name="BusinessId">Scope_7cd19931-c20c-
11dd-23b1-005056c00001</bpelx:property>
161         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
162     </bpelx:analysis>
163     </bpelx:annotation>
164 <sequence>
165     <bpelx:annotation>
166     <bpelx:analysis>
167         <bpelx:property name="BusinessId">Sequence_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
168         <bpelx:property name="LastUpdateDate">12/4/08
3:16:49 PM</bpelx:property>
169     </bpelx:analysis>
170     </bpelx:annotation>
171     <invoke name="C">
172     <bpelx:annotation>

```

```

173             <bpelx:documentation></bpelx:documentation>
174             <bpelx:analysis>
175                 <bpelx:property
name="Documentation"></bpelx:property>
176                 <bpelx:property name="BusinessId">Invoke_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
177                 <bpelx:property name="Label">C</bpelx:property>
178                 <bpelx:property name="LastUpdateDate">12/4/08
3:16:49 PM</bpelx:property>
179             </bpelx:analysis>
180         </bpelx:annotation>
181     </invoke>
182 </sequence>
183 </scope>
184 </sequence>
185 </case>
186 <otherwise>
187     <bpelx:annotation>
188         <bpelx:pattern patternName="Case"/>
189         <bpelx:analysis>
190             <bpelx:property name="BusinessId">7cd1993a-c20c-11dd-23b1-
005056c00001_Case_default</bpelx:property>
191             <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
192             <bpelx:property name="ConditionExpression">STALE,
WITHDRAWN, ERRORED, EXPIRED</bpelx:property>
193         </bpelx:analysis>
194     </bpelx:annotation>
195 <sequence>
196     <bpelx:annotation>
197         <bpelx:analysis>
198             <bpelx:property
name="BusinessId">Sequence_default</bpelx:property>
199             <bpelx:property name="LastUpdateDate">12/4/08 3:19:41
PM</bpelx:property>
200         </bpelx:analysis>
201     </bpelx:annotation>
202 <scope name="default">
203     <bpelx:annotation>
204         <bpelx:pattern patternName="bpelx:automated"/>
205     <bpelx:analysis>
206         <bpelx:property name="BusinessId">Scope_d827b771-c614-
11dd-23b1-005056c00001</bpelx:property>
207     </bpelx:analysis>
208 </bpelx:annotation>
209 </sequence>
210 <bpelx:annotation>

```

```

211         <bpelx:analysis>
212             <bpelx:property name="BusinessId">Sequence_d827b771-
c614-11dd-23b1-005056c00001</bpelx:property>
213             <bpelx:property name="LastUpdateDate">12/9/08
6:14:32 PM</bpelx:property>
214         </bpelx:analysis>
215     </bpelx:annotation>
216     <empty name="default">
217     <bpelx:annotation>
218     <bpelx:analysis>
219         <bpelx:property name="BusinessId">Empty_d827b771-
c614-11dd-23b1-005056c00001</bpelx:property>
220         <bpelx:property name="LastUpdateDate">12/9/08
6:14:32 PM</bpelx:property>
221     </bpelx:analysis>
222 </bpelx:annotation>
223 </empty>
224 </sequence>
225 </scope>
226 </sequence>
227 </otherwise>
228 </switch>
229 <scope name="D">
230     <bpelx:annotation>
231         <bpelx:pattern patternName="bpelx:automated"/>
232     <bpelx:analysis>
233         <bpelx:property name="BusinessId">Scope_7cd1992a-c20c-11dd-
23b1-005056c00001</bpelx:property>
234         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
235     </bpelx:analysis>
236 </bpelx:annotation>
237 <sequence>
238     <bpelx:annotation>
239     <bpelx:analysis>
240         <bpelx:property name="BusinessId">Sequence_7cd1992a-c20c-
11dd-23b1-005056c00001</bpelx:property>
241         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
242     </bpelx:analysis>
243 </bpelx:annotation>
244 <invoke name="D">
245     <bpelx:annotation>
246         <bpelx:documentation></bpelx:documentation>
247     <bpelx:analysis>
248         <bpelx:property name="Documentation"></bpelx:property>

```

```

249         <bpelx:property name="BusinessId">Invoke_7cd1992a-c20c-
11dd-23b1-005056c00001</bpelx:property>
250         <bpelx:property name="Label">D</bpelx:property>
251         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
252         </bpelx:analysis>
253         </bpelx:annotation>
254         </invoke>
255     </sequence>
256 </scope>
257 <scope name="End">
258     <bpelx:annotation>
259         <bpelx:pattern patternName="bpelx:automated"/>
260     <bpelx:analysis>
261         <bpelx:property name="BusinessId">Scope_7cd19917-c20c-11dd-
23b1-005056c00001</bpelx:property>
262         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
263         </bpelx:analysis>
264     </bpelx:annotation>
265     <sequence>
266         <bpelx:annotation>
267             <bpelx:analysis>
268                 <bpelx:property name="BusinessId">Sequence_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
269                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
270                 </bpelx:analysis>
271             </bpelx:annotation>
272             <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client"
portType="tns:WFCP_04and05Callback">
273                 <bpelx:annotation>
274                     <bpelx:documentation></bpelx:documentation>
275                 <bpelx:analysis>
276                     <bpelx:property name="Documentation"></bpelx:property>
277                     <bpelx:property name="BusinessId">Invoke_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
278                     <bpelx:property name="Label">End</bpelx:property>
279                     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
280                 </bpelx:analysis>
281             </bpelx:annotation>
282         </invoke>
283     </sequence>
284 </scope>
285 </sequence>

```

286 </process>

Diagram 6: WFCP 10 - Arbitrary Cycle (while-loop)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--Generated by Oracle BPA Suite-->
3 <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:tns="http://xmlns.oracle.com/WFCP_10_while"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
name="WFCP_10_while" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116" targetNamespace="http://xmlns.oracle.com/WFCP_10_while">
4 <bpelx:annotation>
5 <bpelx:pattern patternName="bpelx:generated" />
6 <bpelx:analysis>
7 <bpelx:property name="Description"></bpelx:property>
8 <bpelx:property name="BusinessProcessIdentifier">75302ad0-c605-
11dd-23b1-005056c00001</bpelx:property>
9 <bpelx:property name="LastUpdateDate">12/9/08 5:05:53
PM</bpelx:property>
10 <bpelx:property name="BPELProcessIdentifier">3c1ebc11-c60b-11dd-
23b1-005056c00001</bpelx:property>
11 <bpelx:property name="Filter">dd838074-ac29-11d4-85b8-
00005a4053ff</bpelx:property>
12 </bpelx:analysis>
13 </bpelx:annotation>
14 <partnerLinks>
15 <partnerLink myRole="WFCP_10_whileProvider" name="client"
partnerLinkType="tns:WFCP_10_while" partnerRole="WFCP_10_whileRequester" />
16 </partnerLinks>
17 <variables>
18 <variable messageType="tns:WFCP_10_whileResponseMessage"
name="outputVariable" />
19 <variable messageType="tns:WFCP_10_whileRequestMessage"
name="inputVariable" />
20 </variables>
21 <sequence>
22 <bpelx:annotation>
23 <bpelx:analysis>
24 <bpelx:property name="BusinessId">Sequence_75302ad0-c605-11dd-
23b1-005056c00001</bpelx:property>
25 <bpelx:property name="LastUpdateDate">12/9/08 5:05:49
PM</bpelx:property>
26 </bpelx:analysis>
27 </bpelx:annotation>
28 <scope name="Start">
29 <bpelx:annotation>
30 <bpelx:pattern patternName="bpelx:StartEvent" />
31 <bpelx:analysis>
32 <bpelx:property name="BusinessId">Scope_7cd19910-c20c-11dd-
23b1-005056c00001</bpelx:property>
33 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
34 </bpelx:analysis>
35 </bpelx:annotation>
36 </sequence>
37 <bpelx:annotation>
38 <bpelx:analysis>
39 <bpelx:property name="BusinessId">Sequence_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
```

```

40         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
41     </bpelx:analysis>
42 </bpelx:annotation>
43     <receive createInstance="yes" name="Start "
operation="initiate" partnerLink="client" portType="tns:WFCP_10_while"
variable="inputVariable">
44         <bpelx:annotation>
45         <bpelx:documentation></bpelx:documentation>
46         <bpelx:analysis>
47         <bpelx:property
name="CreateInstance">true</bpelx:property>
48         <bpelx:property name="BusinessId">Receive_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
49         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
50         <bpelx:property name="Documentation"></bpelx:property>
51         <bpelx:property name="Label">Start</bpelx:property>
52     </bpelx:analysis>
53 </bpelx:annotation>
54 </receive>
55 </sequence>
56 </scope>
57 <scope name="A">
58     <bpelx:annotation>
59     <bpelx:pattern patternName="bpelx:automated"/>
60     <bpelx:analysis>
61     <bpelx:property name="BusinessId">Scope_7cd19911-c20c-11dd-
23b1-005056c00001</bpelx:property>
62     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
63     </bpelx:analysis>
64 </bpelx:annotation>
65 <sequence>
66     <bpelx:annotation>
67     <bpelx:analysis>
68     <bpelx:property name="BusinessId">Sequence_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
69     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
70     </bpelx:analysis>
71 </bpelx:annotation>
72 <invoke name="A">
73     <bpelx:annotation>
74     <bpelx:documentation></bpelx:documentation>
75     <bpelx:analysis>
76     <bpelx:property name="Documentation"></bpelx:property>
77     <bpelx:property name="BusinessId">Invoke_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
78     <bpelx:property name="Label">A</bpelx:property>
79     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
80     </bpelx:analysis>
81 </bpelx:annotation>
82 </invoke>
83 </sequence>
84 </scope>
85 <while name="XOR_rule">
86     <bpelx:annotation>
87     <bpelx:pattern patternName="While"/>
88     <bpelx:analysis>

```

```

89         <bpelx:property name="BusinessId">While_a818b2f4-c605-11dd-
23b1-005056c00001</bpelx:property>
90         <bpelx:property name="Label">XOR_rule</bpelx:property>
91         <bpelx:property name="LastUpdateDate">12/9/08 4:28:50
PM</bpelx:property>
92     </bpelx:analysis>
93 </bpelx:annotation>
94 <sequence>
95     <bpelx:annotation>
96     <bpelx:analysis>
97         <bpelx:property name="BusinessId">Sequence_a818b2f4-c605-
11dd-23b1-005056c00001</bpelx:property>
98         <bpelx:property name="LastUpdateDate">12/9/08 4:28:50
PM</bpelx:property>
99     </bpelx:analysis>
100 </bpelx:annotation>
101 <scope name="B">
102     <bpelx:annotation>
103     <bpelx:pattern patternName="bpelx:automated" />
104     <bpelx:analysis>
105         <bpelx:property name="BusinessId">Scope_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>
106         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
107     </bpelx:analysis>
108 </bpelx:annotation>
109 <sequence>
110     <bpelx:annotation>
111     <bpelx:analysis>
112         <bpelx:property name="BusinessId">Sequence_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
113         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
114     </bpelx:analysis>
115 </bpelx:annotation>
116 <invoke name="B">
117     <bpelx:annotation>
118     <bpelx:documentation></bpelx:documentation>
119     <bpelx:analysis>
120     <bpelx:property
name="Documentation"></bpelx:property>
121     <bpelx:property name="BusinessId">Invoke_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
122     <bpelx:property name="Label">B</bpelx:property>
123     <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
124     </bpelx:analysis>
125 </bpelx:annotation>
126 </invoke>
127 </sequence>
128 </scope>
129 </sequence>
130 </while>
131 <scope name="End">
132     <bpelx:annotation>
133     <bpelx:pattern patternName="bpelx:automated" />
134     <bpelx:analysis>
135         <bpelx:property name="BusinessId">Scope_7cd19917-c20c-11dd-
23b1-005056c00001</bpelx:property>
136         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>

```



```

137         </bpelx:analysis>
138     </bpelx:annotation>
139     <sequence>
140         <bpelx:annotation>
141             <bpelx:analysis>
142                 <bpelx:property name="BusinessId">Sequence_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
143                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
144             </bpelx:analysis>
145         </bpelx:annotation>
146         <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client"
portType="tns:WFCP_10_whileCallback">
147             <bpelx:annotation>
148                 <bpelx:documentation></bpelx:documentation>
149             <bpelx:analysis>
150                 <bpelx:property name="Documentation"></bpelx:property>
151                 <bpelx:property name="BusinessId">Invoke_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
152                 <bpelx:property name="Label">End</bpelx:property>
153                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
154             </bpelx:analysis>
155         </bpelx:annotation>
156     </invoke>
157 </sequence>
158 </scope>
159 </sequence>
160 </process>

```

Diagram 5: WFCP 11 - Implicit Termination

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--Generated by Oracle BPA Suite-->
3  <process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:tns="http://xmlns.oracle.com/WFCP_11"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
name="WFCP_11" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
targetNamespace="http://xmlns.oracle.com/WFCP_11">
4      <bpelx:annotation>
5          <bpelx:pattern patternName="bpelx:generated"/>
6          <bpelx:analysis>
7              <bpelx:property name="Description"></bpelx:property>
8              <bpelx:property name="BusinessProcessIdentifier">f31b3480-c20e-
11dd-23b1-005056c00001</bpelx:property>
9              <bpelx:property name="LastUpdateDate">12/9/08 6:19:31
PM</bpelx:property>
10             <bpelx:property name="BPELProcessIdentifier">883d8951-c615-11dd-
23b1-005056c00001</bpelx:property>
11             <bpelx:property name="Filter">dd838074-ac29-11d4-85b8-
00005a4053ff</bpelx:property>
12         </bpelx:analysis>

```

```

13     </bpelx:annotation>
14     <partnerLinks>
15         <partnerLink myRole="WFCP_11Provider" name="client"
partnerLinkType="tns:WFCP_11" partnerRole="WFCP_11Requester"/>
16     </partnerLinks>
17     <variables>
18         <variable messageType="tns:WFCP_11RequestMessage"
name="inputVariable"/>
19         <variable messageType="tns:WFCP_11ResponseMessage"
name="outputVariable"/>
20     </variables>
21     <sequence>
22         <bpelx:annotation>
23             <bpelx:analysis>
24                 <bpelx:property name="BusinessId">Sequence_f31b3480-c20e-11dd-
23b1-005056c00001</bpelx:property>
25                 <bpelx:property name="LastUpdateDate">12/9/08 6:19:28
PM</bpelx:property>
26             </bpelx:analysis>
27         </bpelx:annotation>
28         <scope name="Start">
29             <bpelx:annotation>
30                 <bpelx:pattern patternName="bpelx:StartEvent"/>
31             </bpelx:analysis>
32                 <bpelx:property name="BusinessId">Scope_7cd19910-c20c-11dd-
23b1-005056c00001</bpelx:property>
33                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
34             </bpelx:analysis>
35         </bpelx:annotation>
36     </sequence>
37     <bpelx:annotation>
38         <bpelx:analysis>
39             <bpelx:property name="BusinessId">Sequence_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
40             <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
41         </bpelx:analysis>
42     </bpelx:annotation>
43     <receive createInstance="yes" name="Start"
operation="initiate" partnerLink="client" portType="tns:WFCP_11"
variable="inputVariable">
44         <bpelx:annotation>
45             <bpelx:documentation></bpelx:documentation>
46         </bpelx:analysis>
47         <bpelx:property
name="CreateInstance">true</bpelx:property>

```

```

48         <bpelx:property name="BusinessId">Receive_7cd19910-c20c-
11dd-23b1-005056c00001</bpelx:property>
49         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
50         <bpelx:property name="Documentation"></bpelx:property>
51         <bpelx:property name="Label">Start</bpelx:property>
52     </bpelx:analysis>
53 </bpelx:annotation>
54 </receive>
55 </sequence>
56 </scope>
57 <scope name="A">
58     <bpelx:annotation>
59         <bpelx:pattern patternName="bpelx:automated"/>
60     <bpelx:analysis>
61         <bpelx:property name="BusinessId">Scope_7cd19911-c20c-11dd-
23b1-005056c00001</bpelx:property>
62         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
63     </bpelx:analysis>
64 </bpelx:annotation>
65 <sequence>
66     <bpelx:annotation>
67         <bpelx:analysis>
68             <bpelx:property name="BusinessId">Sequence_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
69             <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
70         </bpelx:analysis>
71     </bpelx:annotation>
72 <invoke name="A">
73     <bpelx:annotation>
74         <bpelx:documentation></bpelx:documentation>
75     <bpelx:analysis>
76         <bpelx:property name="Documentation"></bpelx:property>
77         <bpelx:property name="BusinessId">Invoke_7cd19911-c20c-
11dd-23b1-005056c00001</bpelx:property>
78         <bpelx:property name="Label">A</bpelx:property>
79         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
80     </bpelx:analysis>
81 </bpelx:annotation>
82 </invoke>
83 </sequence>
84 </scope>
85 <flow name="AND_rule">
86     <bpelx:annotation>

```

```

87         <bpelx:pattern patternName="Flow"/>
88         <bpelx:analysis>
89             <bpelx:property name="BusinessId">Flow_f4be3626-c20e-11dd-
23b1-005056c00001</bpelx:property>
90             <bpelx:property name="Label">AND_rule</bpelx:property>
91             <bpelx:property name="LastUpdateDate">12/4/08 3:23:42
PM</bpelx:property>
92         </bpelx:analysis>
93     </bpelx:annotation>
94     <sequence>
95         <bpelx:annotation>
96             <bpelx:analysis>
97                 <bpelx:property name="BusinessId">Sequence_f4be3633-c20e-
11dd-23b1-005056c00001</bpelx:property>
98                 <bpelx:property name="LastUpdateDate">12/4/08 3:23:42
PM</bpelx:property>
99             </bpelx:analysis>
100         </bpelx:annotation>
101         <scope name="C">
102             <bpelx:annotation>
103                 <bpelx:pattern patternName="bpelx:automated"/>
104                 <bpelx:analysis>
105                     <bpelx:property name="BusinessId">Scope_7cd19931-c20c-
11dd-23b1-005056c00001</bpelx:property>
106                     <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
107                 </bpelx:analysis>
108             </bpelx:annotation>
109         </scope>
110     </bpelx:annotation>
111     <bpelx:analysis>
112         <bpelx:property name="BusinessId">Sequence_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
113         <bpelx:property name="LastUpdateDate">12/4/08 3:16:49
PM</bpelx:property>
114     </bpelx:analysis>
115 </bpelx:annotation>
116 <invoke name="C">
117     <bpelx:annotation>
118         <bpelx:documentation></bpelx:documentation>
119     <bpelx:analysis>
120         <bpelx:property
name="Documentation"></bpelx:property>
121         <bpelx:property name="BusinessId">Invoke_7cd19931-
c20c-11dd-23b1-005056c00001</bpelx:property>
122         <bpelx:property name="Label">C</bpelx:property>

```

```

123             <bpelx:property name="LastUpdateDate">12/4/08
3:16:49 PM</bpelx:property>
124             </bpelx:analysis>
125             </bpelx:annotation>
126             </invoke>
127             </sequence>
128         </scope>
129         <scope name="End">
130             <bpelx:annotation>
131                 <bpelx:pattern patternName="bpelx:automated"/>
132                 <bpelx:analysis>
133                     <bpelx:property name="BusinessId">Scope_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>
134                     <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
135                     </bpelx:analysis>
136                     </bpelx:annotation>
137                 <sequence>
138                     <bpelx:annotation>
139                         <bpelx:analysis>
140                             <bpelx:property name="BusinessId">Sequence_7cd19917-
c20c-11dd-23b1-005056c00001</bpelx:property>
141                             <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
142                             </bpelx:analysis>
143                             </bpelx:annotation>
144                             <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client" portType="tns:WFCP_11Callback">
145                                 <bpelx:annotation>
146                                     <bpelx:documentation></bpelx:documentation>
147                                     <bpelx:analysis>
148                                         <bpelx:property
name="Documentation"></bpelx:property>
149                                         <bpelx:property name="BusinessId">Invoke_7cd19917-
c20c-11dd-23b1-005056c00001</bpelx:property>
150                                         <bpelx:property name="Label">End</bpelx:property>
151                                         <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
152                                         </bpelx:analysis>
153                                         </bpelx:annotation>
154                                     </invoke>
155                                 </sequence>
156                             </scope>
157                         </sequence>
158                     <sequence>
159                         <bpelx:annotation>
160                             <bpelx:analysis>

```

```

161         <bpelx:property name="BusinessId">Sequence_f4be362d-c20e-
11dd-23b1-005056c00001</bpelx:property>
162         <bpelx:property name="LastUpdateDate">12/4/08 3:23:42
PM</bpelx:property>
163     </bpelx:analysis>
164 </bpelx:annotation>
165 <scope name="B">
166     <bpelx:annotation>
167         <bpelx:pattern patternName="bpelx:automated"/>
168     <bpelx:analysis>
169         <bpelx:property name="BusinessId">Scope_7cd19915-c20c-
11dd-23b1-005056c00001</bpelx:property>
170         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
171     </bpelx:analysis>
172 </bpelx:annotation>
173 <sequence>
174     <bpelx:annotation>
175     <bpelx:analysis>
176         <bpelx:property name="BusinessId">Sequence_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
177         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
178     </bpelx:analysis>
179 </bpelx:annotation>
180 <invoke name="B">
181     <bpelx:annotation>
182         <bpelx:documentation></bpelx:documentation>
183     <bpelx:analysis>
184         <bpelx:property
name="Documentation"></bpelx:property>
185         <bpelx:property name="BusinessId">Invoke_7cd19915-
c20c-11dd-23b1-005056c00001</bpelx:property>
186         <bpelx:property name="Label">B</bpelx:property>
187         <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
188     </bpelx:analysis>
189 </bpelx:annotation>
190 </invoke>
191 </sequence>
192 </scope>
193 <scope name="End">
194     <bpelx:annotation>
195         <bpelx:pattern patternName="bpelx:automated"/>
196     <bpelx:analysis>
197         <bpelx:property name="BusinessId">Scope_7cd19917-c20c-
11dd-23b1-005056c00001</bpelx:property>

```

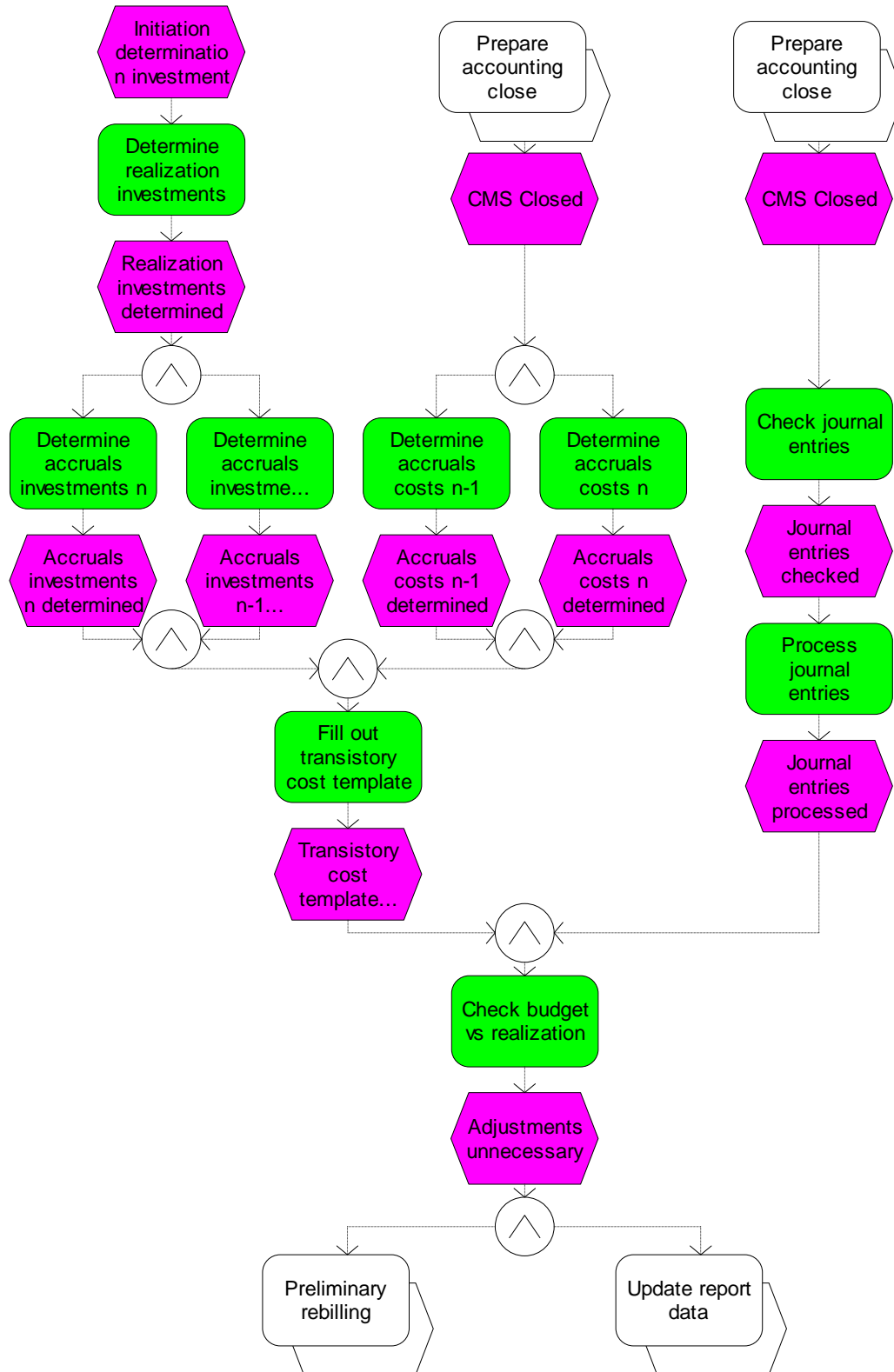
```

198         <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
199         </bpelx:analysis>
200     </bpelx:annotation>
201     <sequence>
202         <bpelx:annotation>
203             <bpelx:analysis>
204                 <bpelx:property name="BusinessId">Sequence_7cd19917-
c20c-11dd-23b1-005056c00001</bpelx:property>
205                 <bpelx:property name="LastUpdateDate">12/4/08 3:11:10
PM</bpelx:property>
206                 </bpelx:analysis>
207             </bpelx:annotation>
208             <invoke inputVariable="outputVariable" name="End"
operation="onResult" partnerLink="client" portType="tns:WFCP_11Callback">
209                 <bpelx:annotation>
210                     <bpelx:documentation></bpelx:documentation>
211                 </bpelx:annotation>
212                 <bpelx:property
name="Documentation"></bpelx:property>
213                 <bpelx:property name="BusinessId">Invoke_7cd19917-
c20c-11dd-23b1-005056c00001</bpelx:property>
214                 <bpelx:property name="Label">End</bpelx:property>
215                 <bpelx:property name="LastUpdateDate">12/4/08
3:11:10 PM</bpelx:property>
216                 </bpelx:analysis>
217             </bpelx:annotation>
218         </invoke>
219     </sequence>
220 </scope>
221 </sequence>
222 </flow>
223 </sequence>
224 </process>

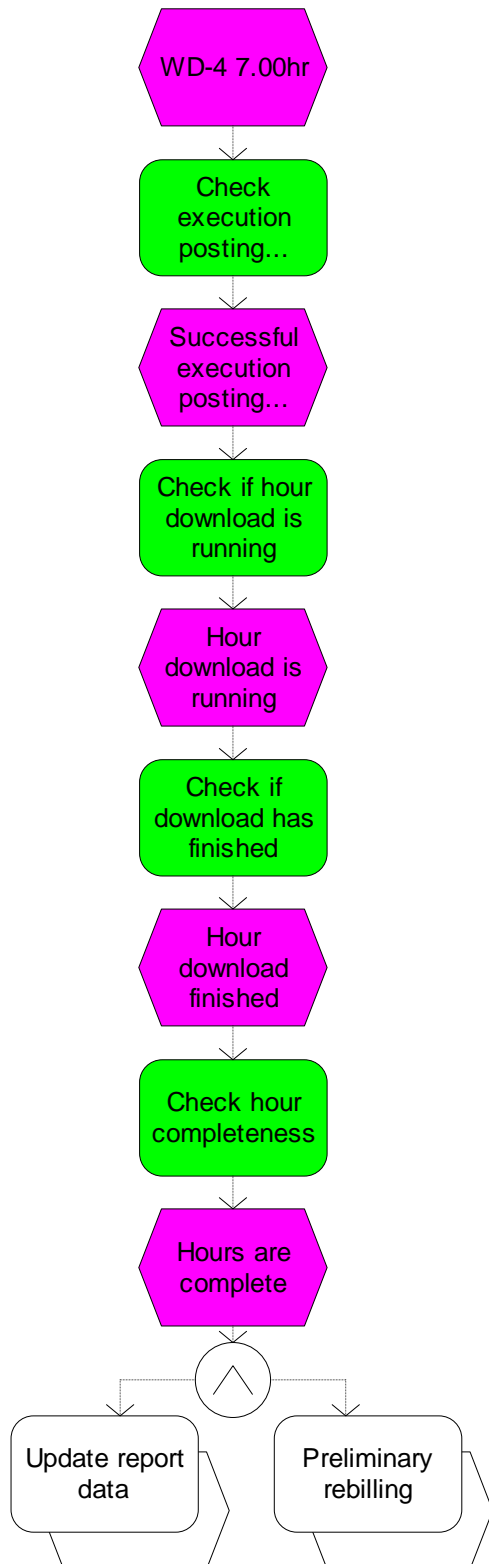
```

Appendix C - Original case EPC diagrams

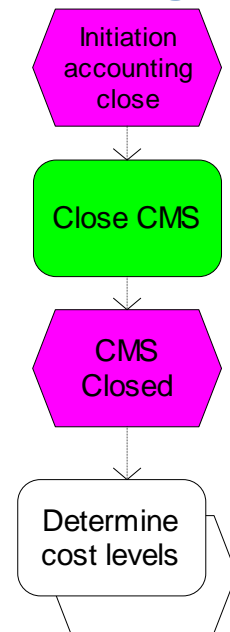
Determine cost levels



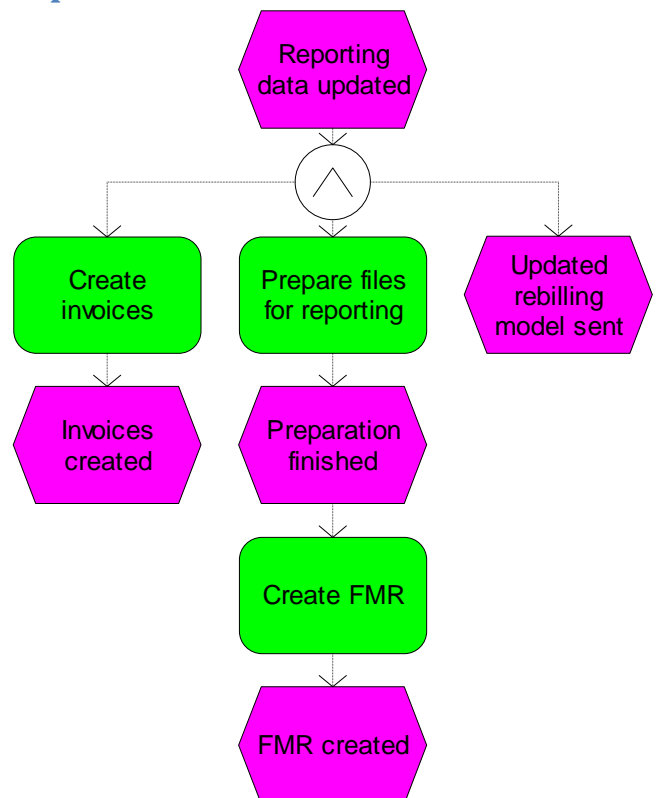
Check final hour download



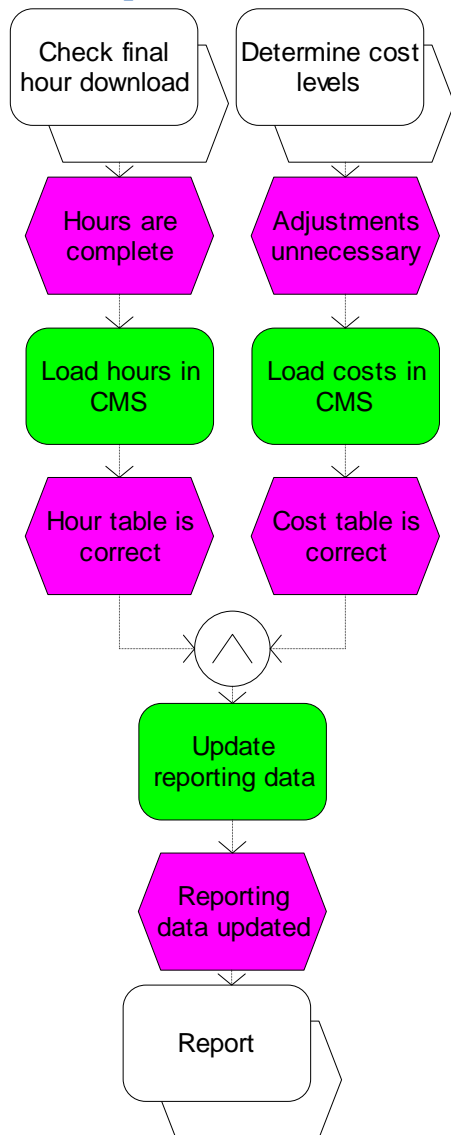
Prepare accounting close



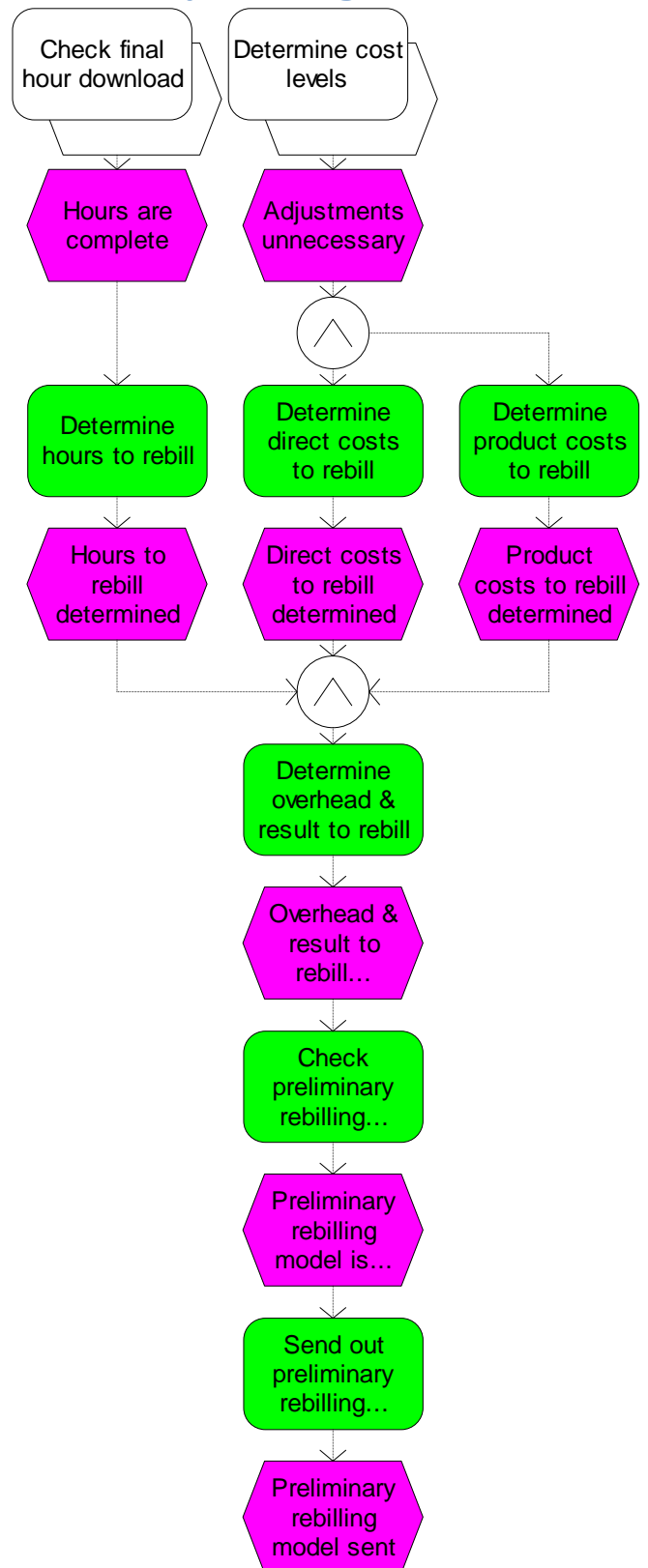
Report



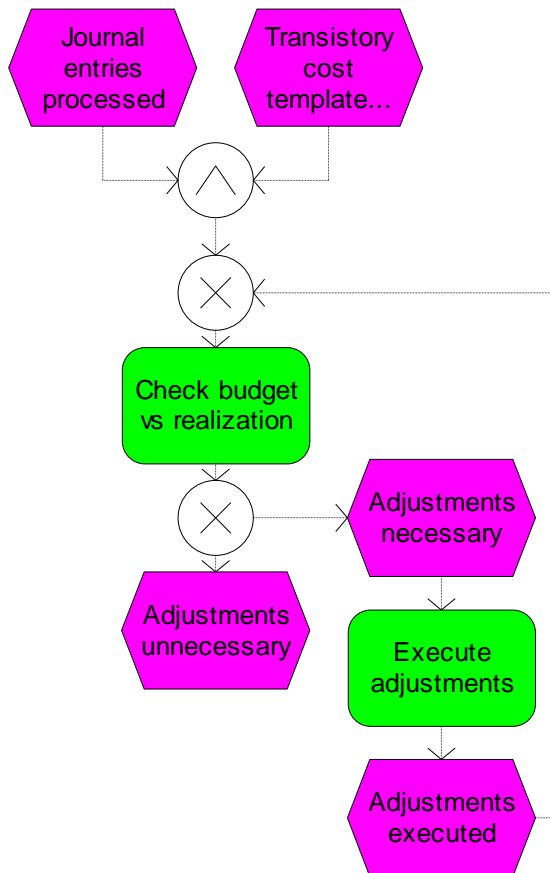
Update report data



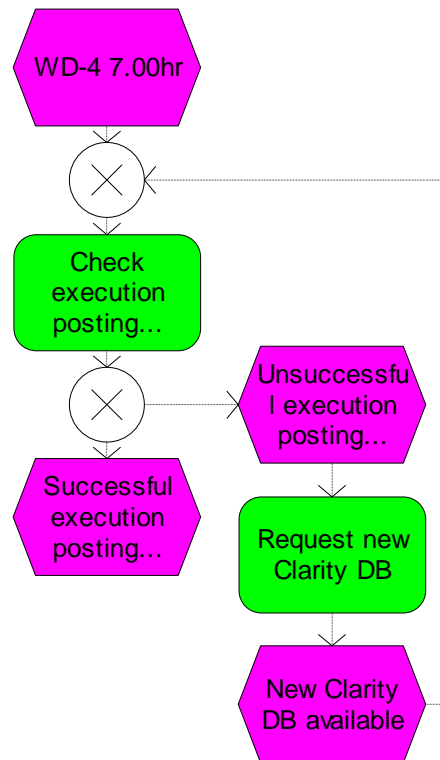
Preliminary rebilling



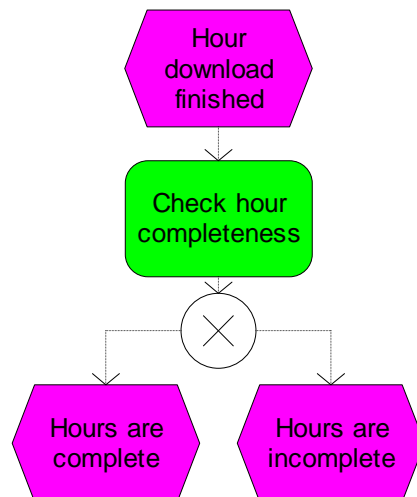
Check budget vs realization (while-loop alternative)



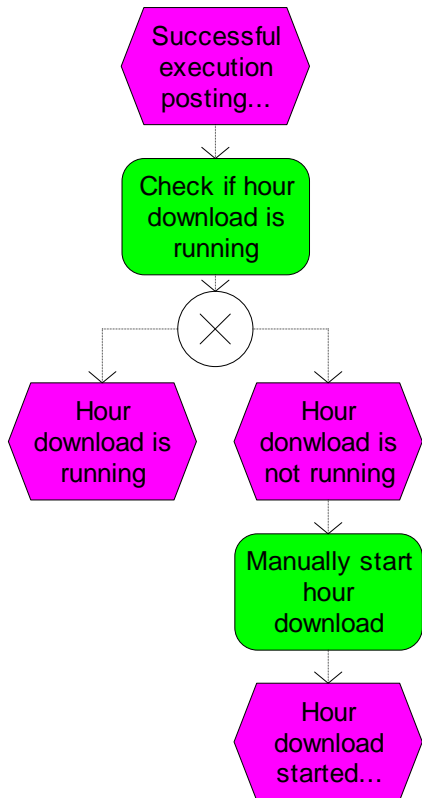
Check execution posting process (while-loop alternative)



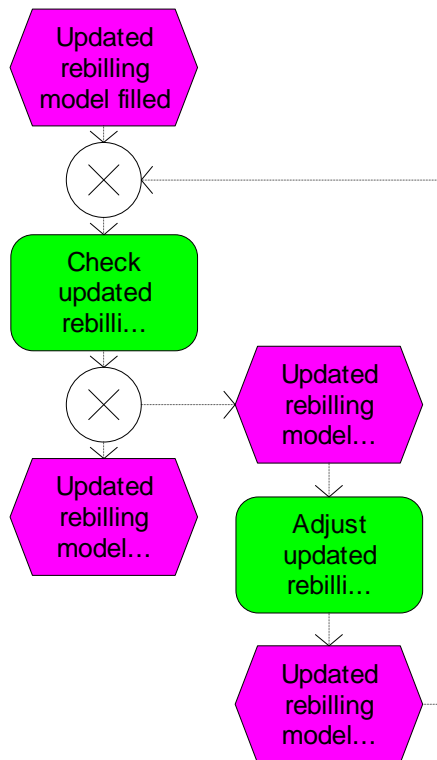
Check hour completeness (choice alternative)



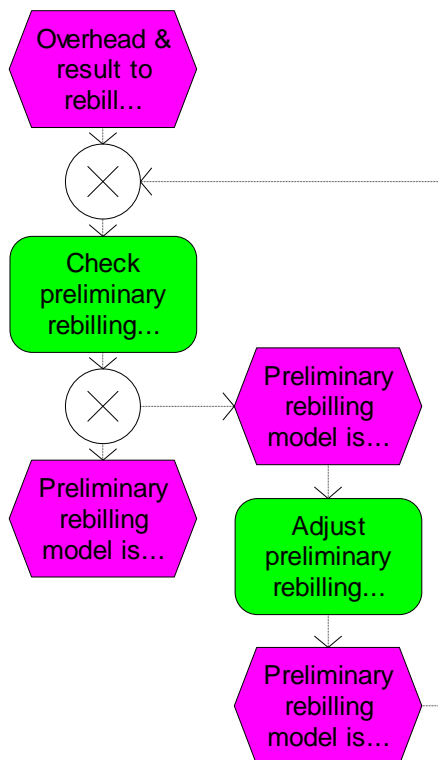
Check if hour download is running (choice alternative)



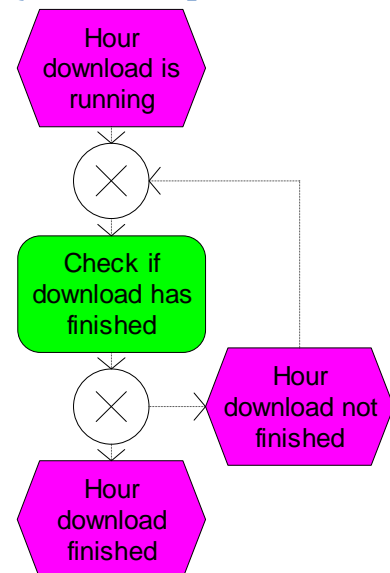
Check updated rebilling model (while-loop alternative)



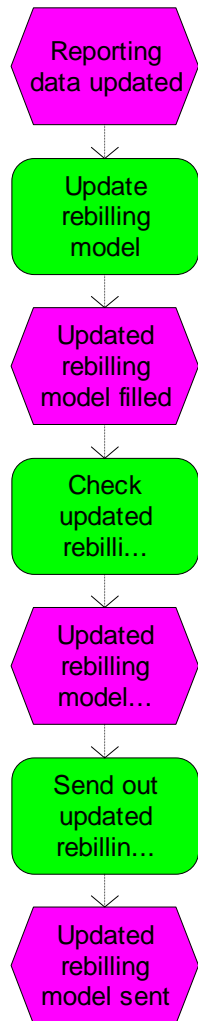
Check preliminary rebilling model (while-loop alternative)



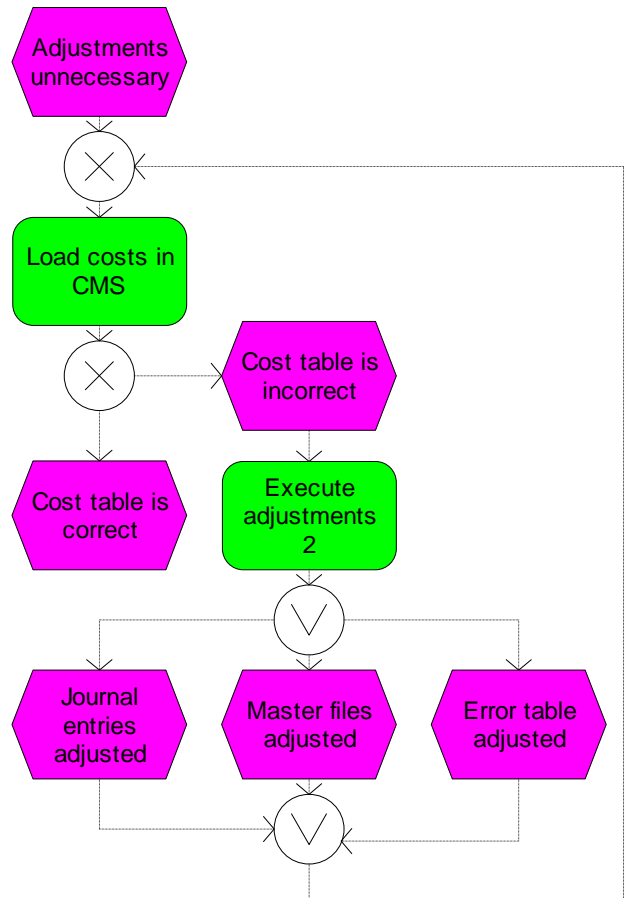
Check if download has finished (while-loop alternative)



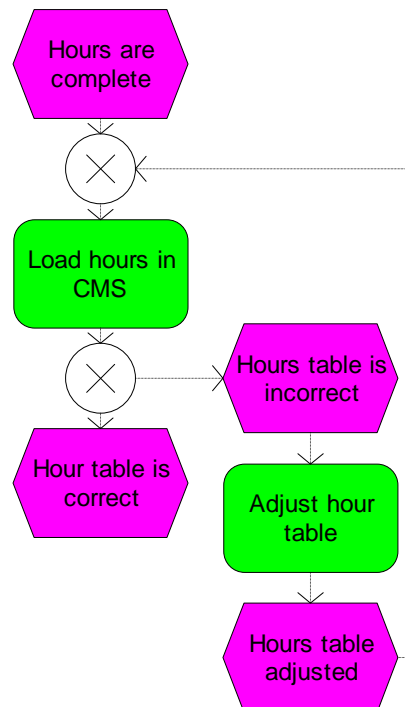
Create and send updated rebilling model (sequence alternative)



Load costs in CMS (while-loop alternative)



Load hours in CMS (while-loop alternative)



Appendix D - Composite case EPC diagrams

Value Added Chain diagram

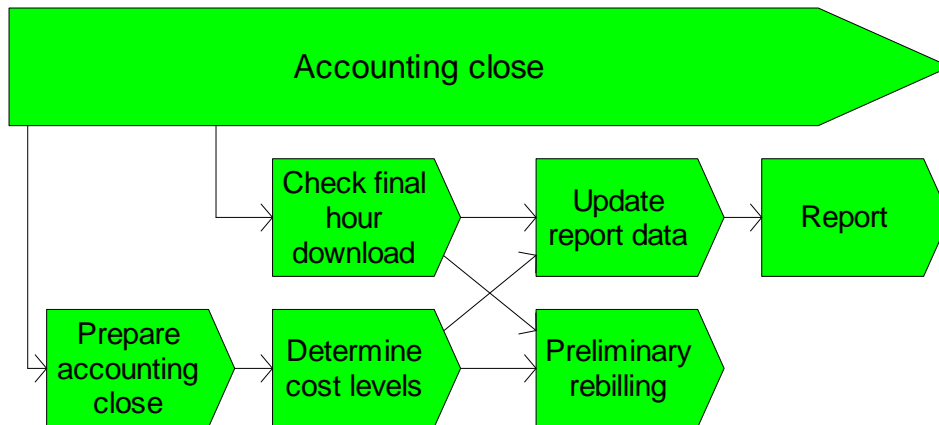


Diagram 8: Prepare accounting close

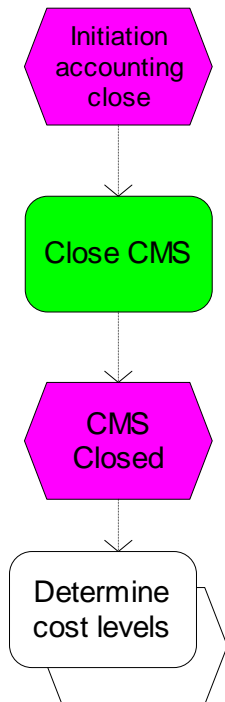


Diagram 9: Determine cost levels

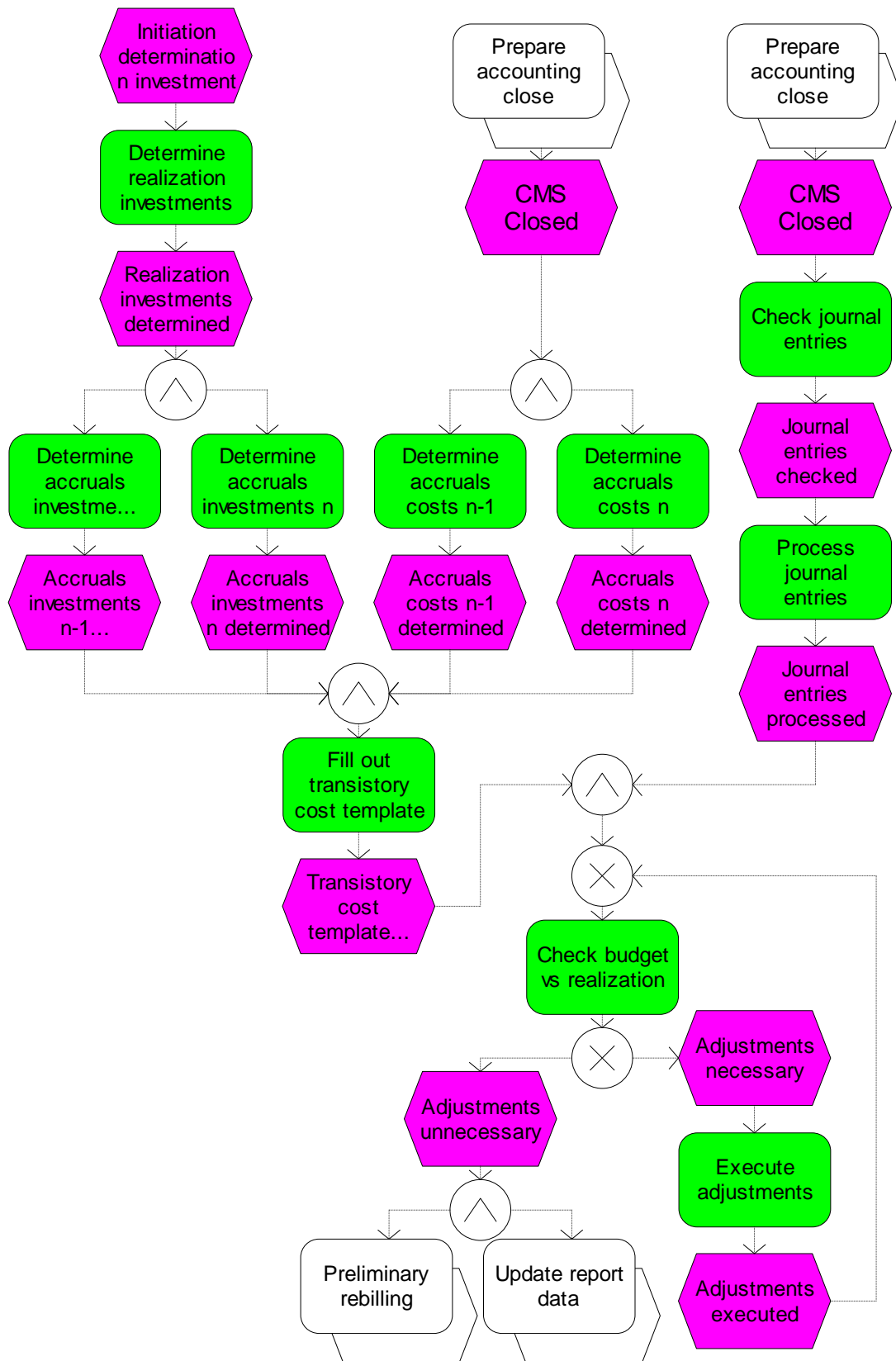


Diagram 10: Check final hour download

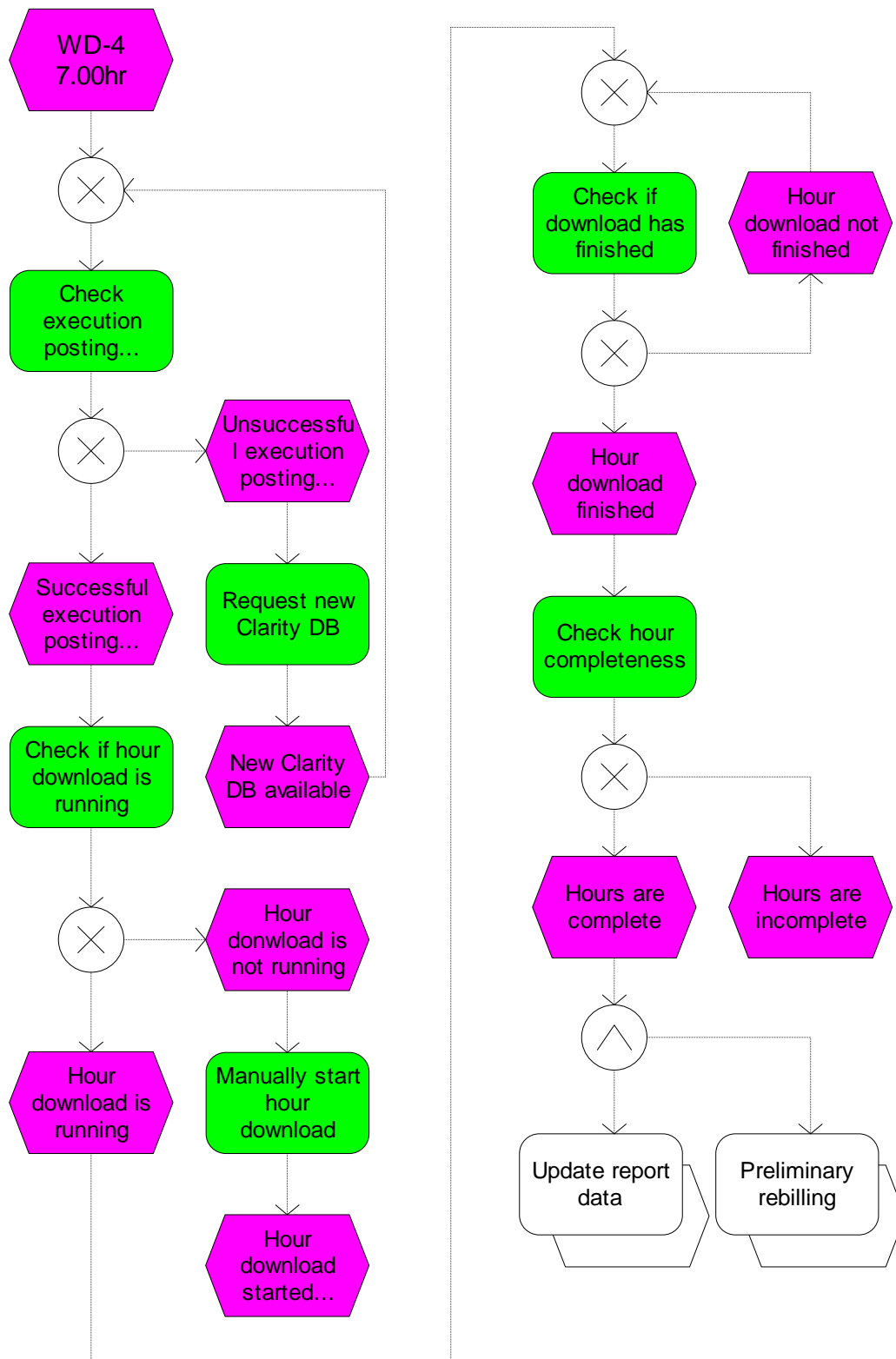


Diagram 11: Preliminary rebilling

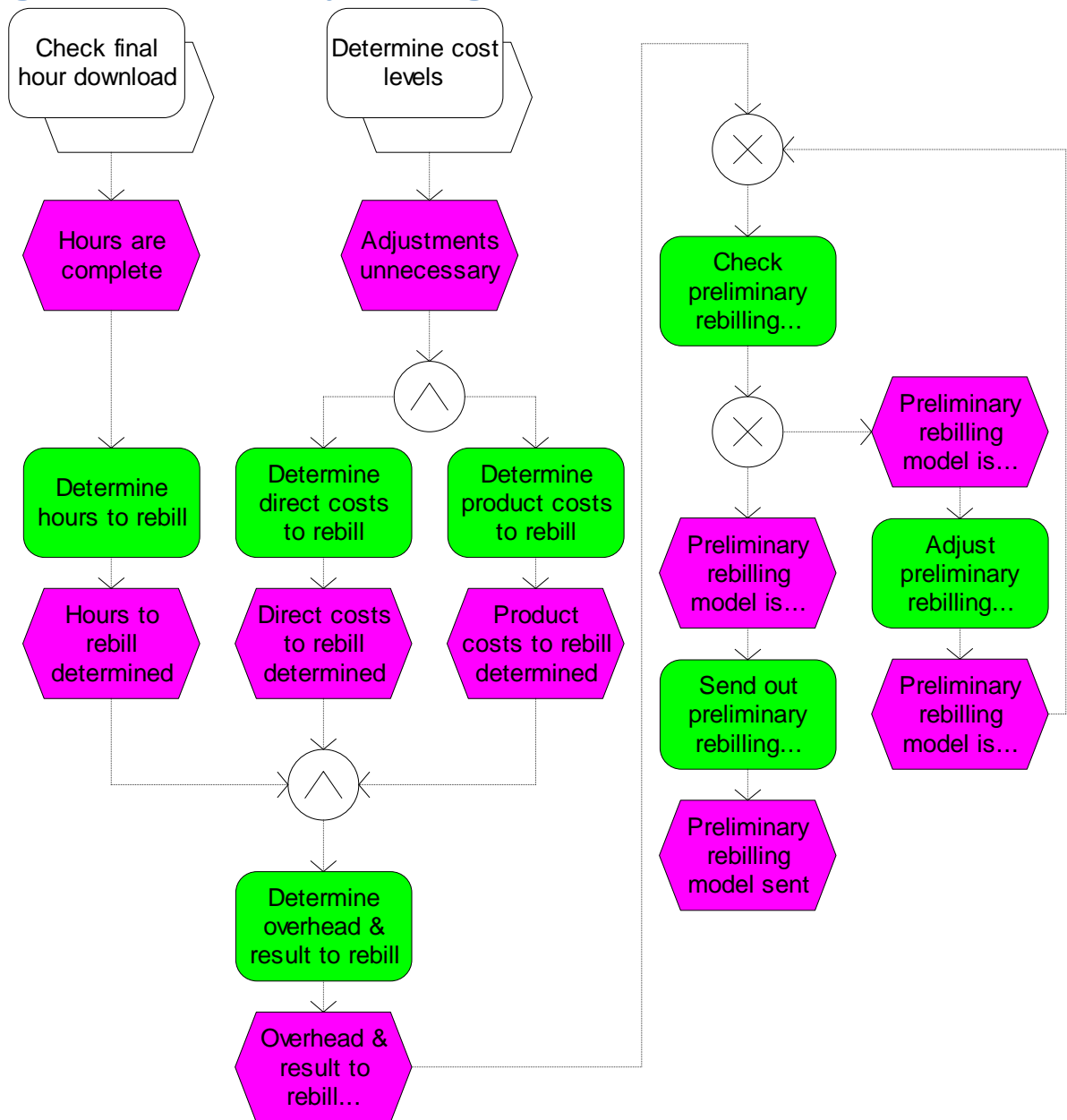


Diagram 12: Update report data

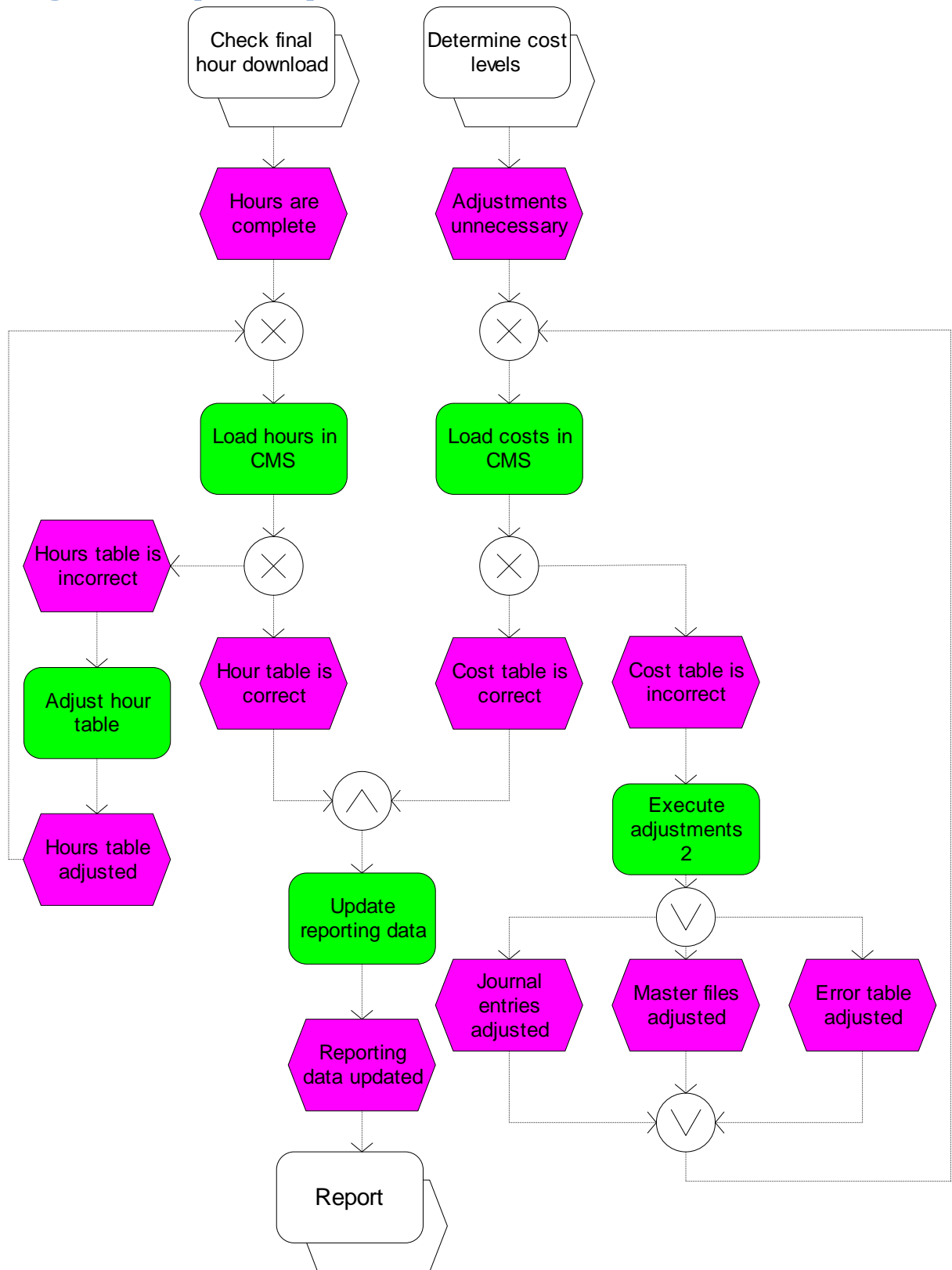
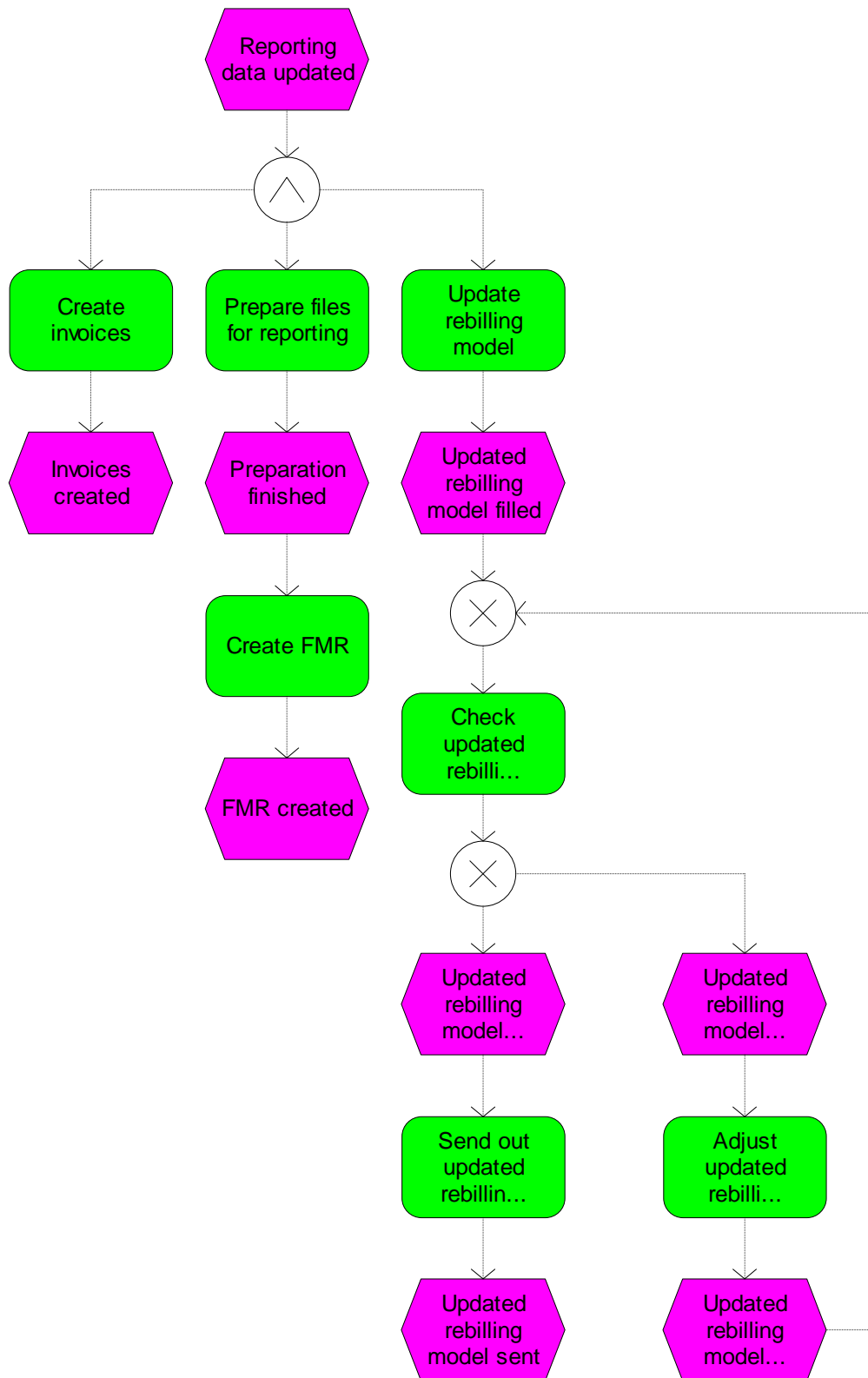


Diagram 13: Report



Appendix E - EPC diagrams of full case

Value Added Chain Diagram

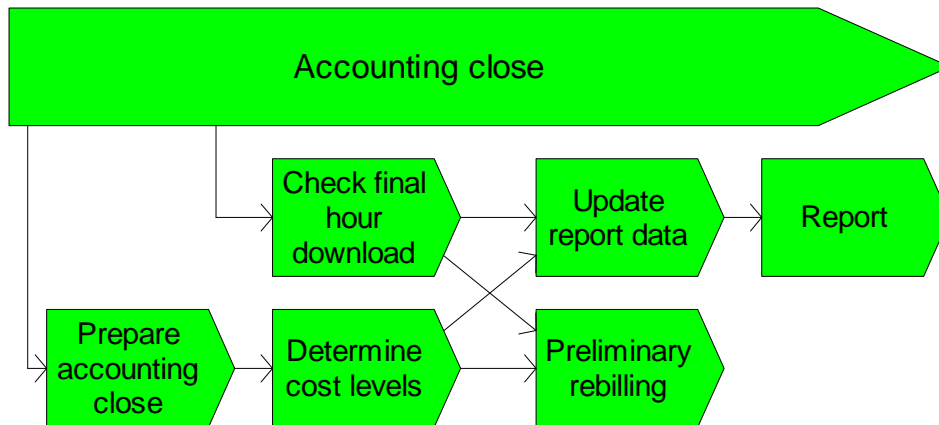
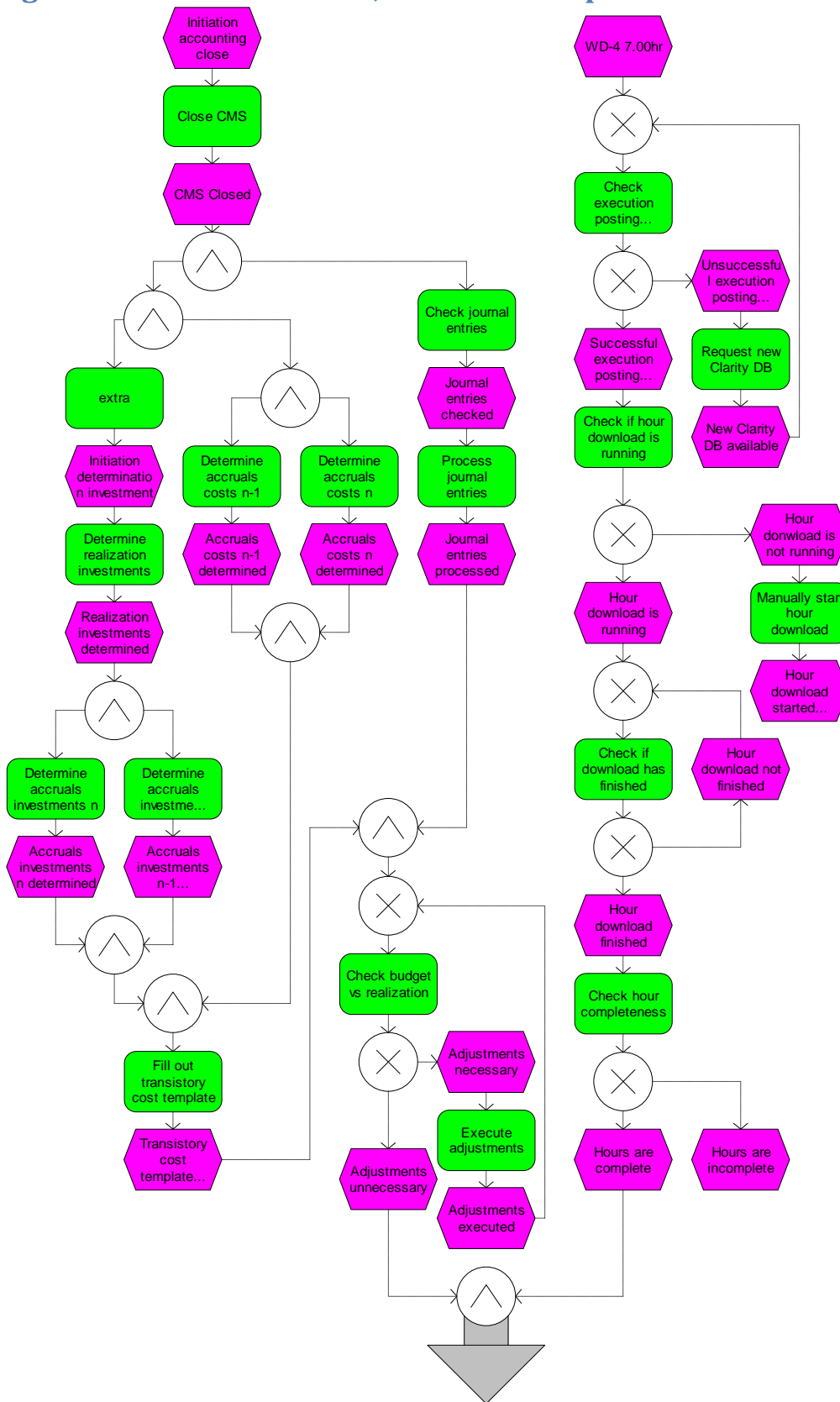


Diagram 14: Transformable, modified composition



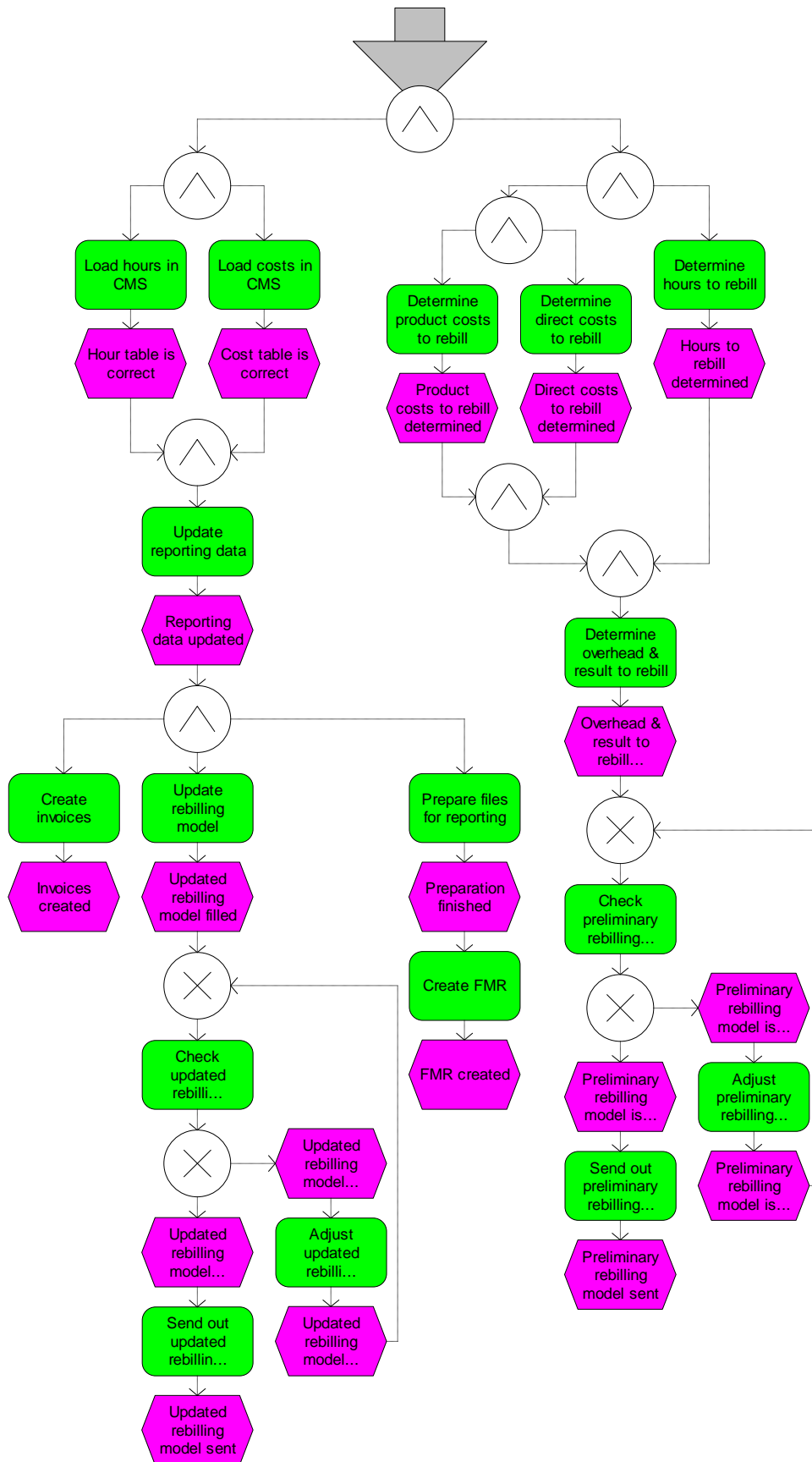
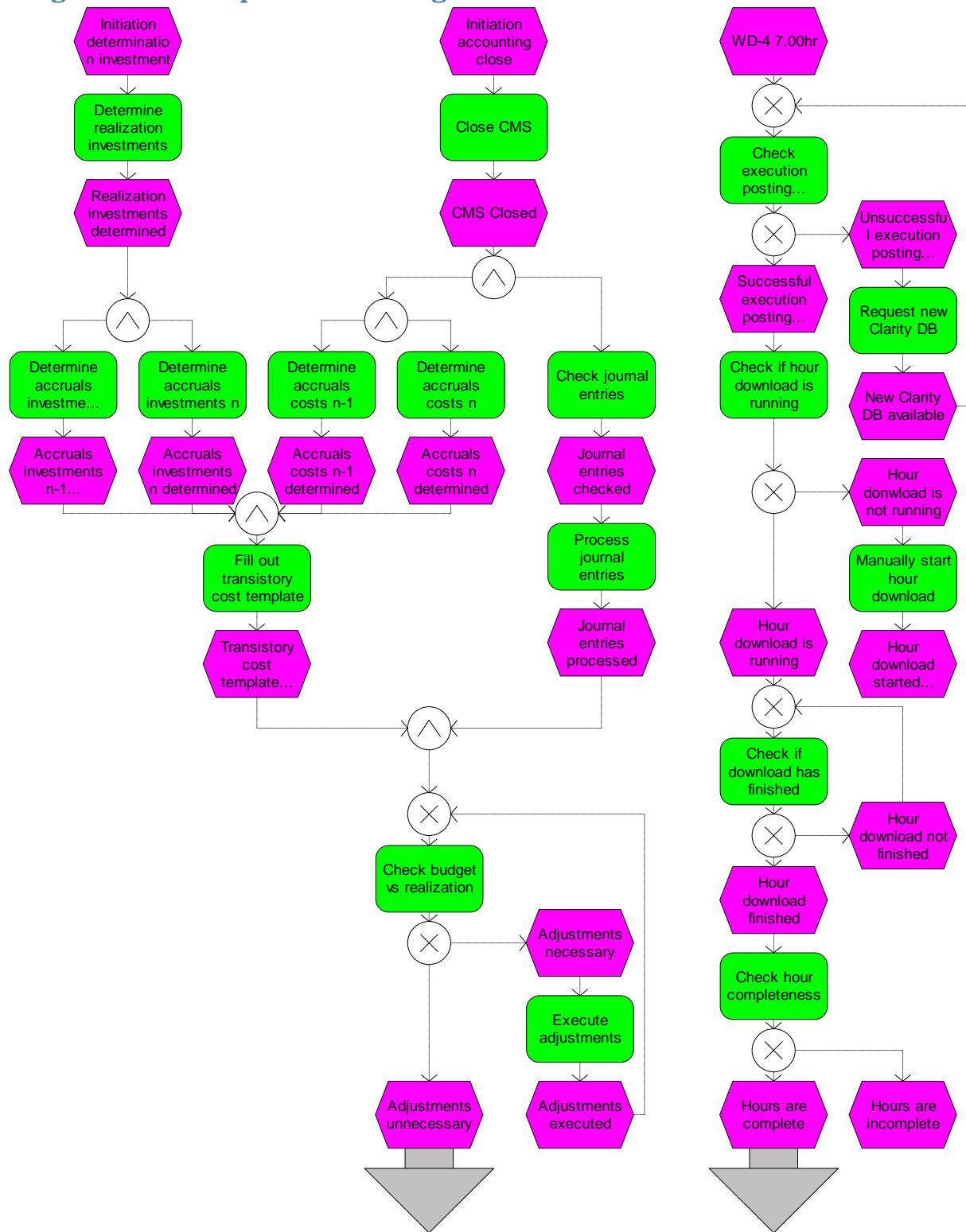
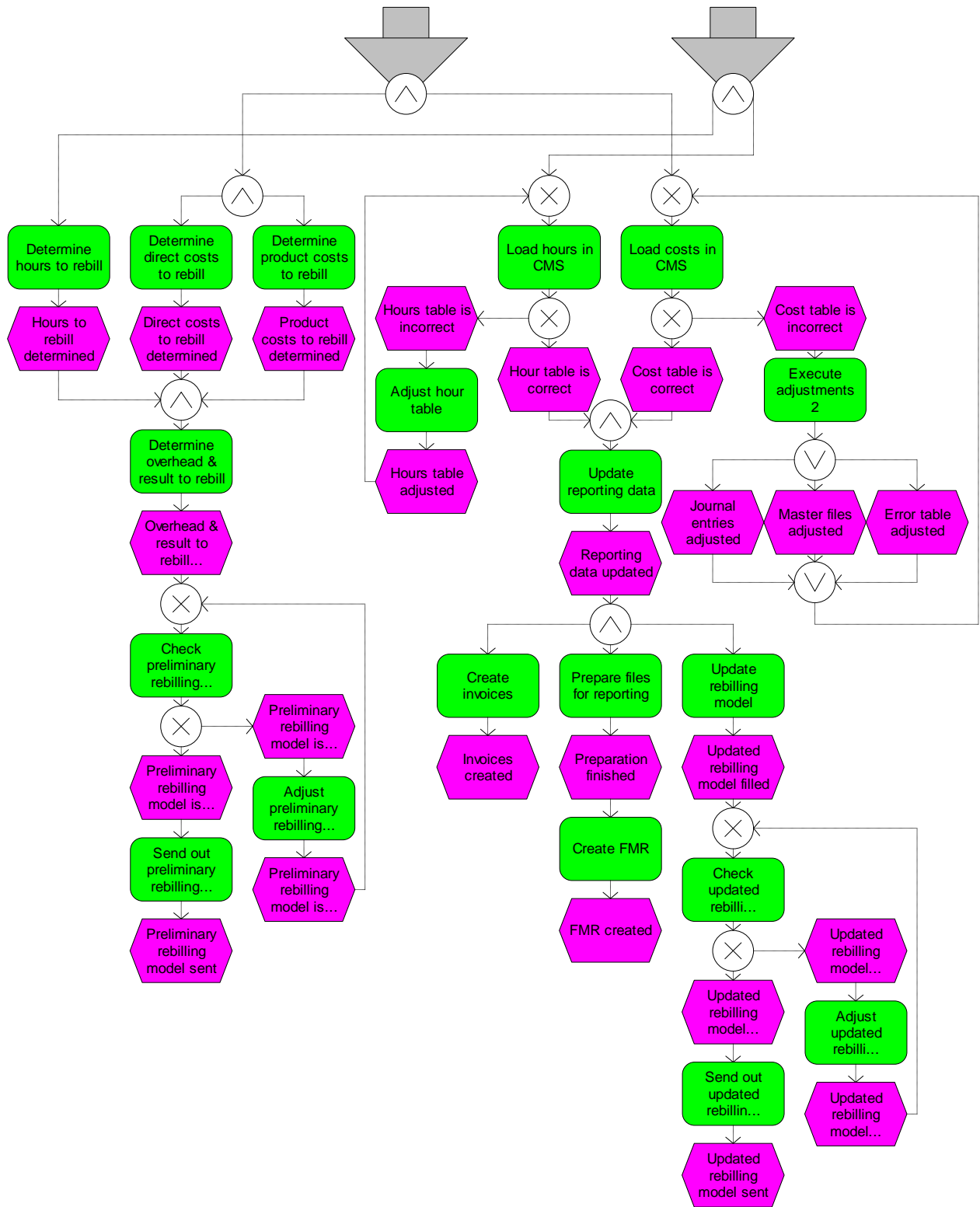


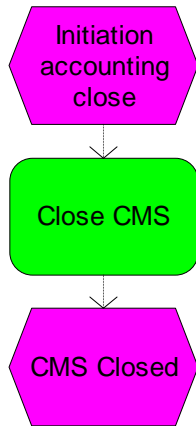
Diagram 15: Composition of original model



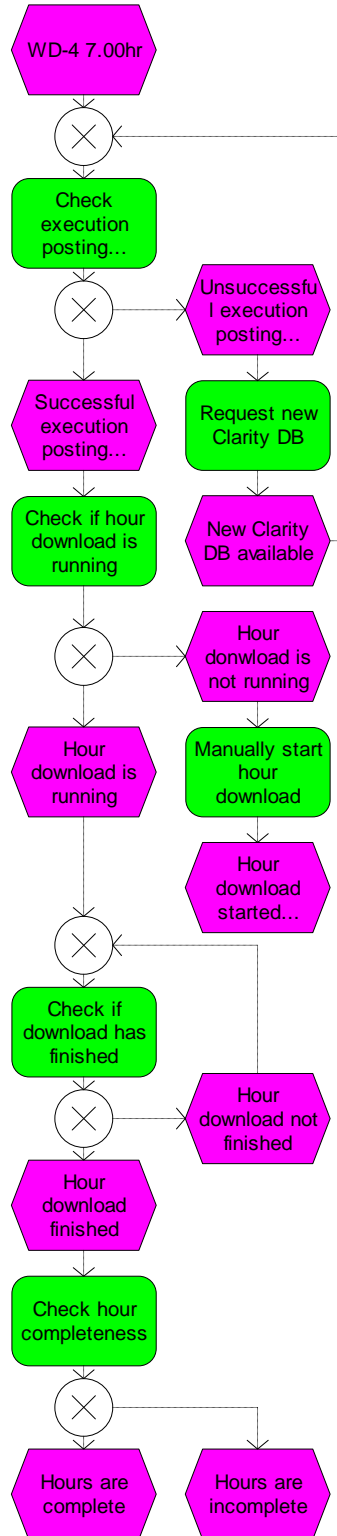


Appendix F - Transformable, modified EPC diagrams

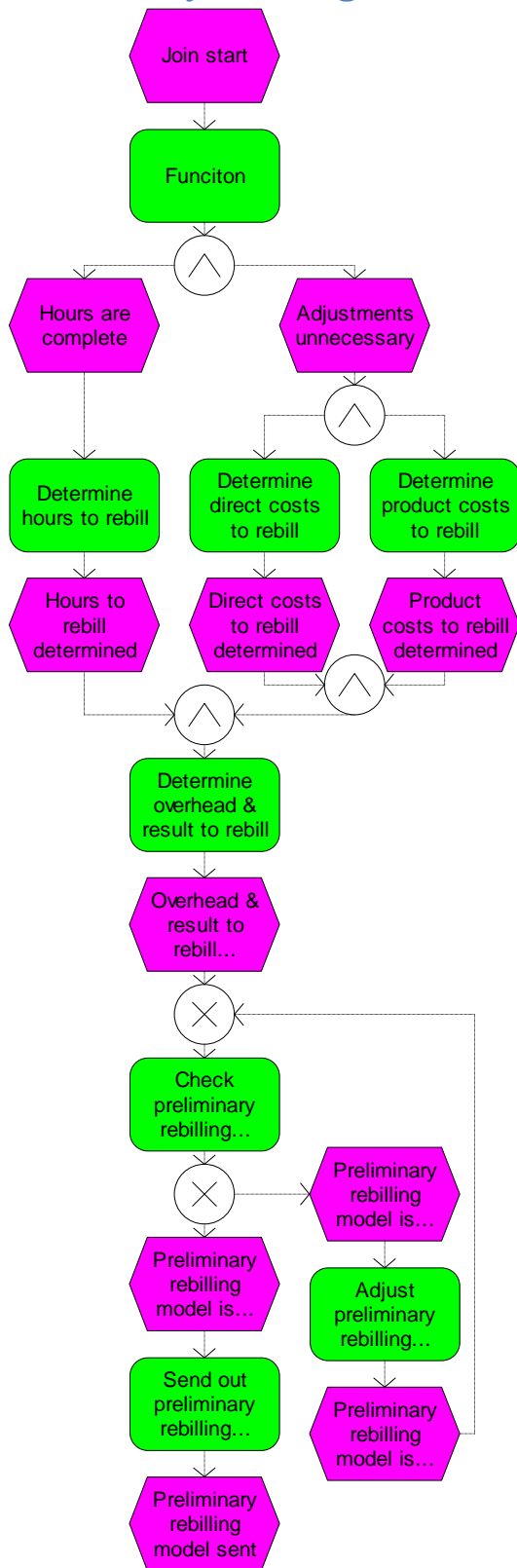
Modified diagram 8: Prepare accounting close



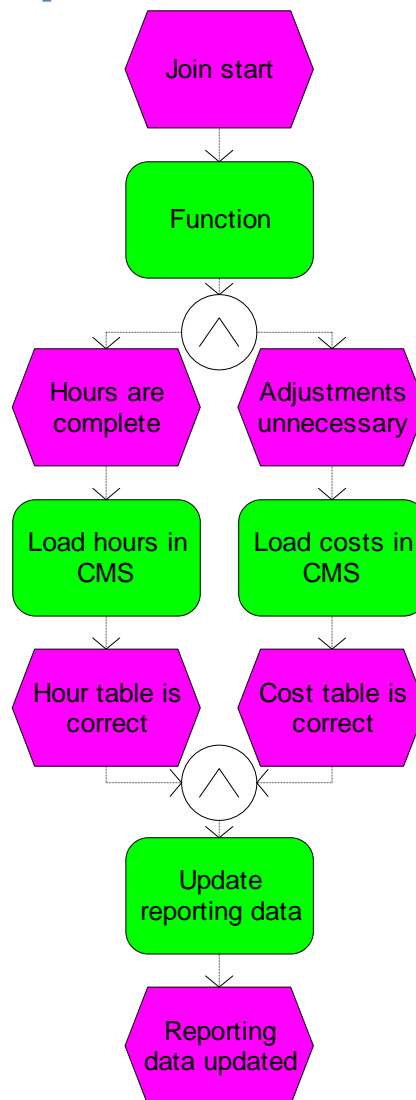
Modified diagram 10: Check final hour download



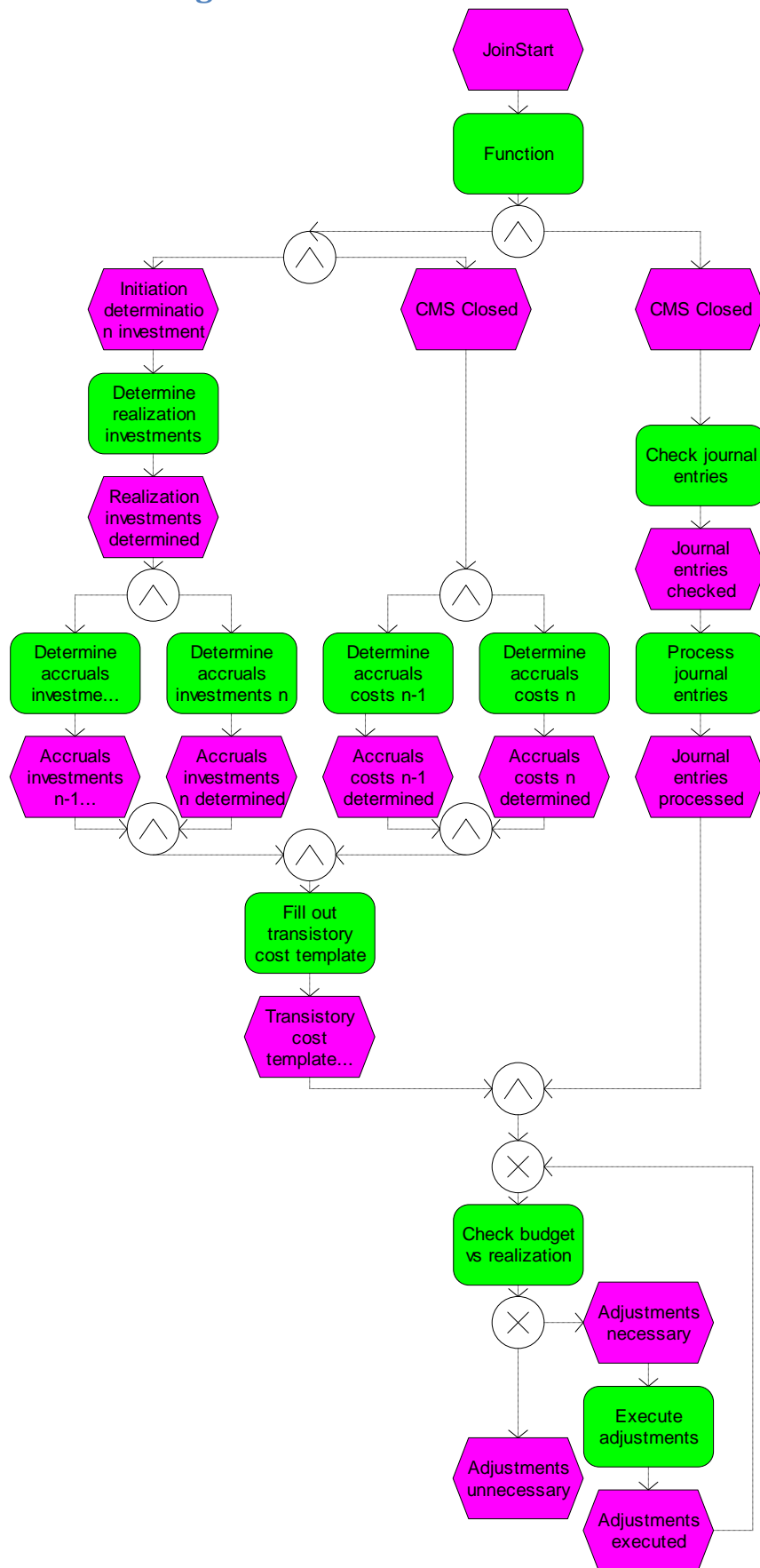
**Modified diagram 11:
Preliminary rebilling**



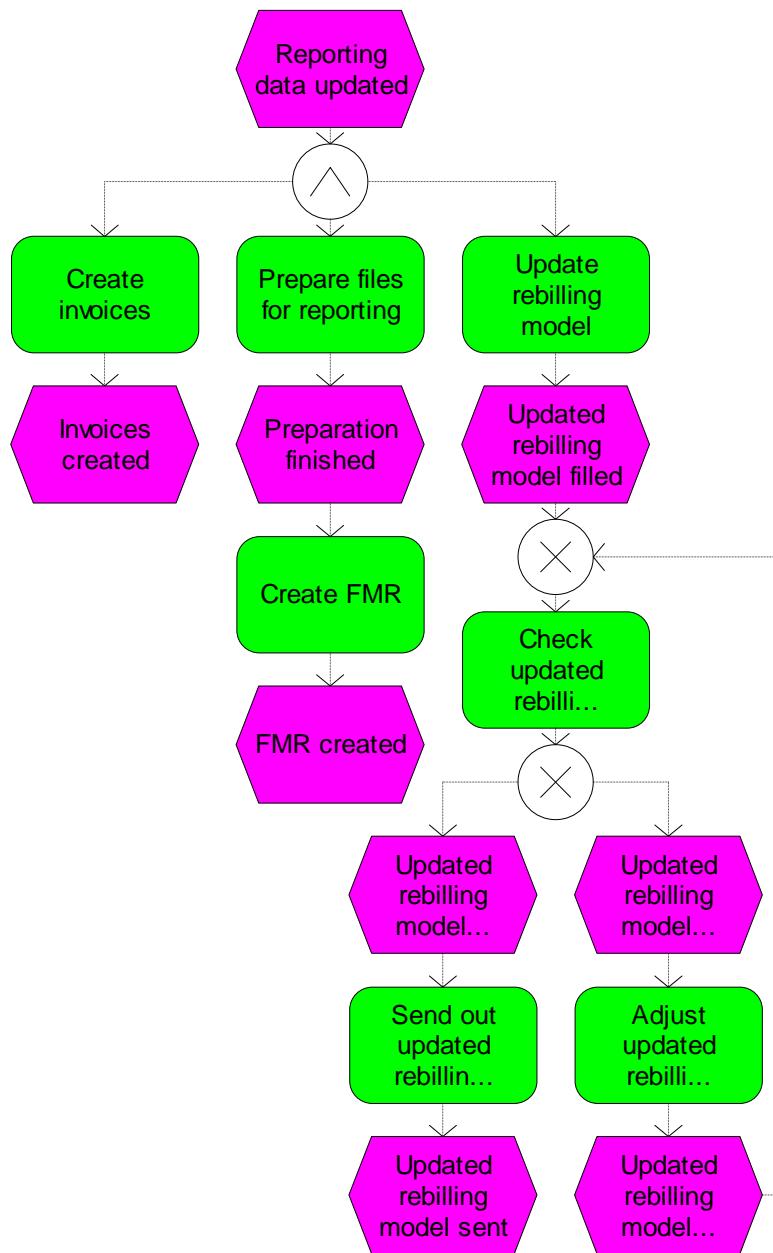
**Modified diagram 12: Update
report data**



Modified diagram 9: Determine cost levels

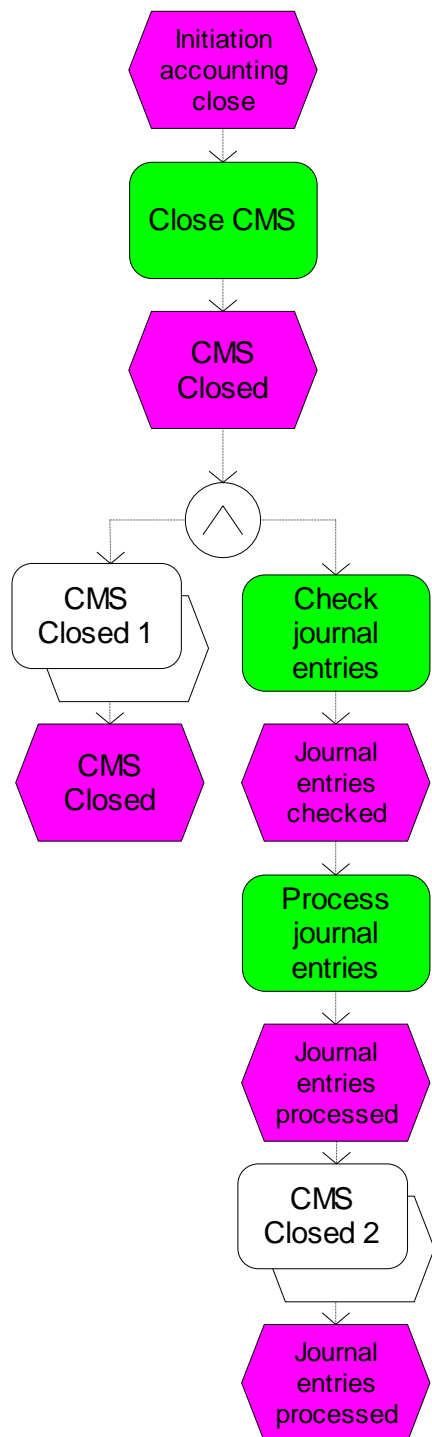


Unmodified diagram 13: Report

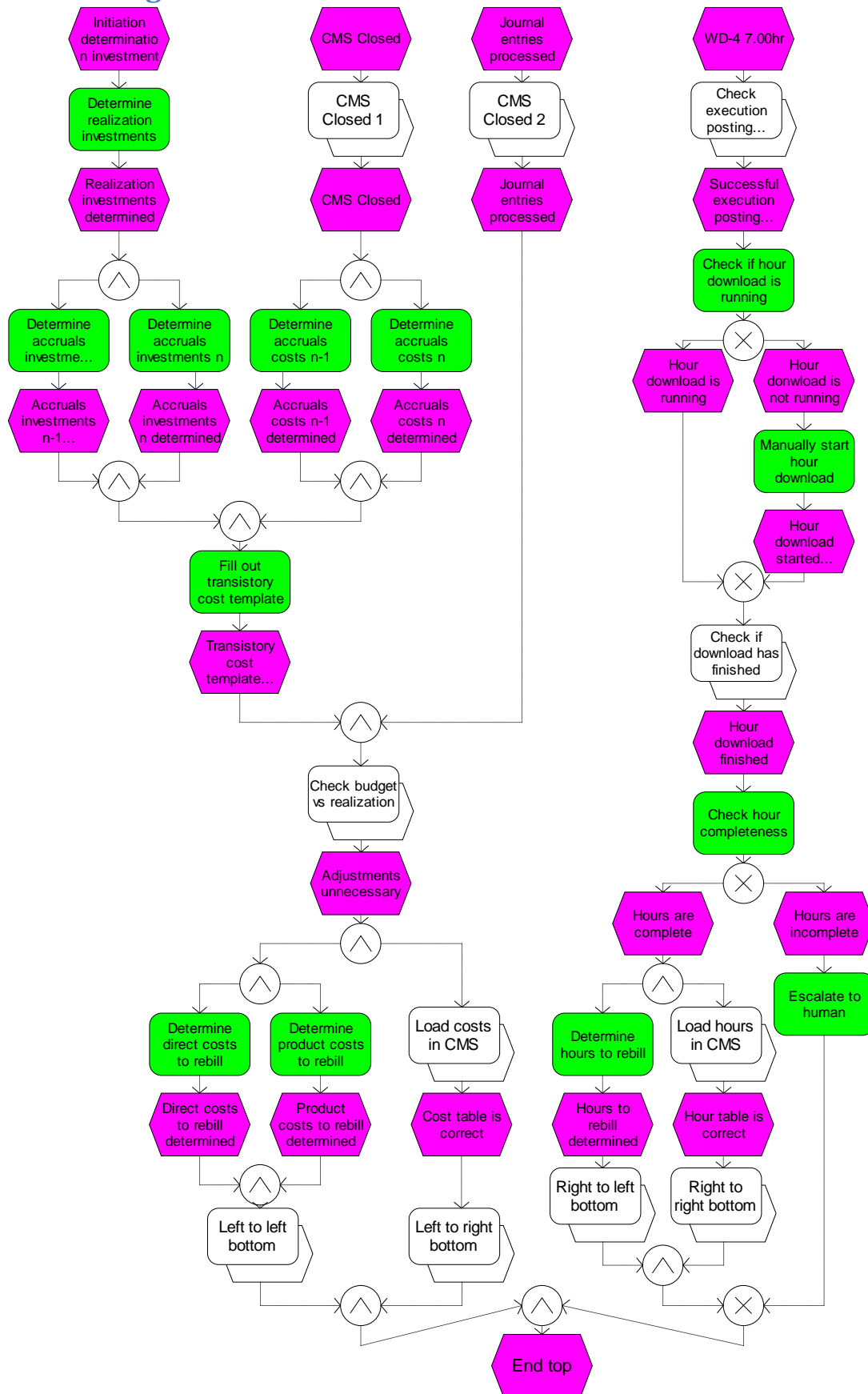


Appendix G - EPC diagrams decomposed from full model

Top diagram



Middle diagram



Bottom

