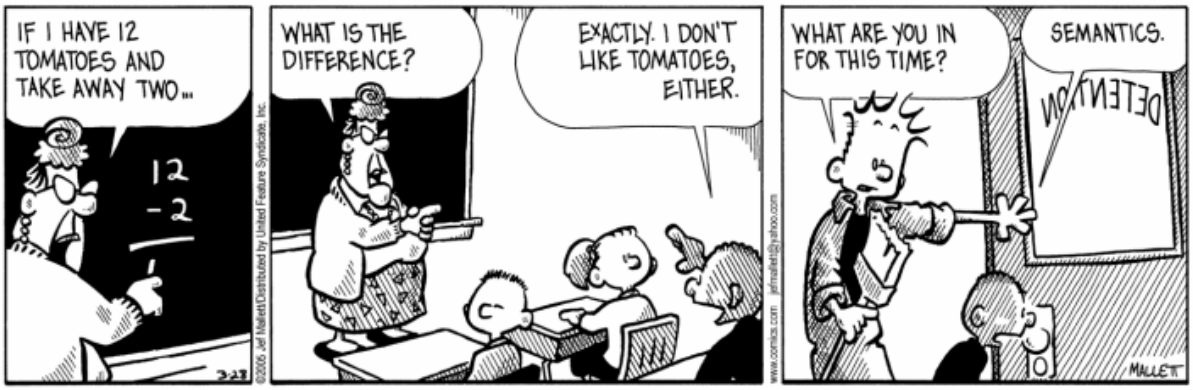
UNIVERSITEIT TWENTE.

MASTER THESIS

A structured approach to identify and resolve semantic conflicts between independently developed information systems



August 2011

Bjorn Bos

Master Thesis Bjorn Bos

A structured approach to identify and resolve semantic conflicts between independently developed information systems

Place and date

Amsterdam, August 26th 2011

Author

B.S. Bos

Programme	Industrial Engineering and Management
	School of Management and Governance
Student number	0068799
Email	b.s.bos@student.utwente.nl

Graduation committee

Ir. E.J.A. Folmer

Department	University of Twente, School of Management and Governance

Dr. M.L. Ehrenhard

Department	University of Twente, School of Management and Governance

Management Summary

In today's world an enterprise's success not only depends on its internal productivity and performance, but also on its ability to partner with others. In order to remain competitive, enterprises thus need to share information with buyers and suppliers so that processes can be aligned, and there is maximum information availability to support business decisions. Interoperability is the concept at which organizations have achieved such connectivity.

Semantic conflicts are an important barrier to overcome when aiming for interoperability. When two or more independently developed information systems are connected, semantic conflicts occur. These differences in the meaning and understanding of exchanged information can lead to wrong business decisions and have high impact, and thus have to be avoided. In this research we make a first attempt to create a methodology that guides the problem holder in this process of semantic conflict identification and resolution.

The methodology we developed consists of four stages. In the first stage the problem holder formalizes the objectives of the interoperability project and defines the concepts to be exchanged. In the second step these concepts are isolated in each participating information system and expressed in an Entity Relationship diagram. In the third step the concepts in the different systems are compared at four different levels: the entity-, attribute-, data format-, and data value level. At each level we indicate the potential semantic conflicts and provide tools to identify them. In the fourth stage the user creates a visual overview of all discovered conflicts. Finally, we propose conflict resolution techniques for each conflict identified by the methodology.

To validate the usability of the methodology in practice, we applied it to a data integration project of *Dienst Uitvoering Onderwijs* and the *SUWI Gegevensregister* in the Netherlands. The results indicate that the methodology is well capable of identifying semantic conflicts between two systems. Compared to the findings from the case holder itself, we discovered similar semantic relationships and conflicts. A few differences indicate suggestions for improvement, most importantly a confirmation of the results after each stage with a domain expert. Further validation was performed by an expert review to measure the general belief in the usefulness of the methodology. Results indicate that the general structure of the method was found to be useful, but that further development is needed to increase its ability to recognize semantically similar concepts in the different systems.

This research makes a first attempt to develop a standard approach for the identification of semantic conflicts, and thereby contributes to the framework for interoperability by targeting the conceptual barriers at the service level. It also provides a new way to categorize semantic conflicts. Instead of segregating by the characteristics of the conflict, we categorize by the entity-, attribute-, data format-, and data value level. Furthermore, the methodology presented in this research help organizations aiming for interoperability to identify semantic conflicts in a more efficient way, and provides suggestions for how to resolve each type of conflict. Finally, we suggest further research for the development of instrument guidelines and tools to support the user in the use of the methodology.

Table of Contents

Preface

This thesis is the last milestone in my studies Industrial Engineering and Management. The research has been very challenging, and learned me a lot about the great complexity of interoperability problems. Although sending information around the world has become enormously easier over the last two decades, the problem of local semantics remain.

Trying to solve a piece of the interoperability puzzle was a daunting task, which I could not have achieved without the help of my university supervisors, Erwin Folmer and Michel Ehrenhard. I want to thank them for the effort and time they put in helping me to accomplish this master thesis. There have been several times in which I got lost in the great complexity of this assignment, and they have always been of great assistance to get me back on the right track. The great knowledge on interoperability from Erwin, with the methodological expertise from Michel, proved to be the perfect mixture to guide me to this end result.

Secondly, I want to thank Dennis Krukkert from *TNO* for the materials and knowledge he has provided me with to conduct the case study found in this report. Additionally, a word of thanks goes out to *Dienst Uitvoering Onderwijs* and *Bureau Keteninformatisering Werk & Inkomen* for providing me permission to study their data integration project.

Thirdly, I want to thank both Fred van Blommestein and Wout Hofman for taking a critical look at this study in the expert review. Their comments provided me with great new insights into the applicability of the proposed methodology in real-life scenarios.

Finally, I want to thank the people from *M4N* for providing me with an inspiring work place to conduct part of this research.

1. Research Context

The Ministry of Economic Affairs in the Netherlands (2007) state that information sharing and connecting business processes, within but most of all between organizations in value chains and networks, more and more becomes a necessity for viable commercial trading. In order to accomplish this, organizations need to be interoperable. This competitive need for interoperability is affirmed by Daclin et al. (2008): "The competitiveness of an enterprise depends not only on its internal productivity and performance, but also on its ability to set up and carry out a partnership with others […] Thus, the concept of interoperability has emerged and aims at *supporting and improving communication and interaction of these partnerships while respecting the constraints imposed by the context in which enterprises evolve*."

Enterprises acknowledge the need for interoperability to remain competitive. Vernadat (1996) concludes that interoperability is one of the key concerns in the enterprise domain. However, even while the basic infrastructure seems to be in place, we have not yet achieved sufficient interoperability (Ralyté, et al. 2008).

So why haven't we yet achieved interoperability? Current research has mostly focused on finding theoretical and/or technical solutions to given specific interoperability problems (Daclin et al, 2008). Also, traditional methods have not managed to solve the interoperability problem as they do not suit the complexity and multifacetedness of the field (Ralyté et al., 2008). As a result, method knowledge related to the information systems interoperability domain still needs to be formalized, managed and evaluated (Ralyté et al., 2008).

This research aims to formalize a methodology that contributes to the interoperability domain by identifying semantic conflicts and by providing guidelines to avoid the conflicts to take place in a live environment. March et al. (2000) discussed semantic interoperability to be one of the most important research issues and technical challenges in heterogeneous and distributed environments. This research topic is relevant for every business that wants to connect or integrate its information system with another independently developed system. Semantic interoperability is a precondition for useful exchange of information. We will now further explain this problem space.

The problem space

The automatic exchange of information from one organization's information system to another is not an easy task. Not only is there a need for a protocol that can transmit the data from one system to another, the hardest part is to make sure that both the transmitter and the receiver give the same meaning to the information. This problem space is defined as the syntactic- and semantic barriers. The semantic barriers relate to the meaning of terms and concepts, the syntactic barriers involve the language to express the terms and concepts.

Semantic problems are not limited to the field of information technology. In fact, semantic misunderstandings have been causing problems for centuries. In the year 1805 the Austrian and Russian emperors agreed to join forces to fight Napoleon's army. They made an agreement to combine their forces on October 20th, in the town of Bavaria. Their plan failed as the Russian forces arrived ten days later than the Austrian forces, giving Napoleon the chance to surround the Austrian army and force surrender on October 21. The reason for the different time of arrival was the use of a

different calendar. While the Austrians were using the Gregorian calendar, the Russians operated the Julian calendar, lagging 10 days behind.

Research Question and Goals

Most interoperability projects cope with two or more independently designed information systems. In the design of each of these systems semantic choices have been made about how to store real world concepts in the information system. The choices made often differ, resulting in semantic conflicts when information is being exchanged. The goal of this research is to identify these conflicts before they actually take place, so that actions can be undertaken to prevent them from happening. Hence, we come to the following research question:

How to identify and resolve semantic conflicts between independently developed information systems by means of a structured approach?

We want to present the result of the research in such a way that it is ready to be used by the problem holder, thereby contributing to a more efficient interoperability project. A methodology is a good way to achieve that goal as "the use of a methodology results in the involvement of less people, less time and effort, and lower costs compared to when no methodology is used in the system development process" (Chatzoglou, 1997), and "it is evident that there is a consensus among many that the use of methodologies is positive and well-advised" (Jenkins et al., 1984).

Before we start developing the methodology, we have to define its requirements. What aspects do we need to include in the methodology? What requirements does the solution have to meet? What (other) factors contribute to a successful fulfillment of our research goal? These and related questions will be examined to achieve our first research goal:

1) *Define the requirements of the methodology to develop*

Once we have clearly defined the restrictions, requirements and goals of the research, we start working on the methodology creation process. To do so, we make use of the Information Engineering Methodology (IEM) Description Model described by Heym and Österle (1992). The model is described in chapter two.

Our second research goal is derived directly from the model. The starting point of our methodology creation process is to define the different stages the user has to go through:

2) *Define the different stages of the methodology and their critical success factors*

Each *stage* produces *deliverables* that form the input to other stages. We therefore need to define the deliverables for each stage, and what input is needed before we can move to a new stage.

Each stage is composed of one or more *tasks,* which are often subdivided into *subtasks*. Each task offers the user *practical guidelines* by providing *techniques* to produce the deliverables, and has associated *rules* and *conventions* for the representation of those deliverables. *Concepts* (such as entity types, attributes, and relationships) model the elementary components a technique deals with.

Our third research goal is to design the tasks that lead to the accomplishment of each stage:

3) *Define the tasks for each stage, and describe the techniques that could be used to accomplish those.*

The next step is to provide validation for the developed methodology. We research the possible validation methods, and apply the most suitable to our research. This is translated into the fourth research goal:

4) Use a sound scientific research method to validate the developed methodology

Findings from this last research goal are then described, and conclusions about the applicability of the developed methodology in practice are explained.

To place this research contribution into the existing body of knowledge of interoperability, we first explain the theoretical framework.

Theoretical Framework

The Framework for Interoperability (CEN/ISO 11354) by Chen et al. (2008) takes into account the basic concepts of several existing frameworks (EIF, 2004) (NEHTA, 2006) (IDEAS, 2003) (ATHENA, 2003). The framework structures the concepts around interoperability, and defines three basic dimensions: interoperability barriers, interoperability concerns, and interoperability approaches (Figure 1-1).

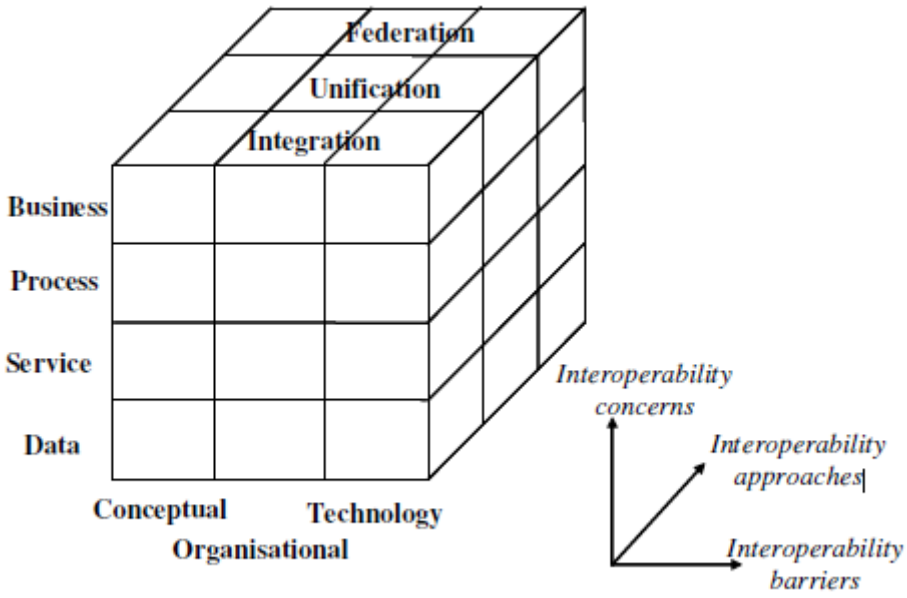


Figure 1-1: Interoperability Framework

There are three categories of problems in the *Interoperability Barriers* dimension: conceptual, technological, and organizational. Conceptual barriers are related to the problems of syntactic and semantic of information to be exchanged. Organizational barriers are related to the definition of responsibilities and authority so that interoperability can take place under good conditions. Technological barriers are related to the standards that are used to present, store, exchange, process, and communicate data through the use of computers.

Interoperability can take place at four *Enterprise Levels*: the business level, the process level, the service level, and the data level. The business level concerns the enterprises abilities to work with each other despite the differences in for example the modes of decision-making, the culture, and commercial approaches. The process level refers to the ability to connect processes from different enterprises to create a common process. The service level aims at solving the syntactic and semantic differences amongst organizations. The data level refers to making different data models and different query languages working together.

Interoperability Approaches concerns the three approaches to remove the barriers. The integrated approach is best suited for mergers between enterprises. With this approach all models are developed according to one standard format. With the unified approach, semantic equivalence is developed with one common meta-model that allows mapping between diverse models. The federated approach results in the lowest level of interoperability. With this approach, systems are dynamically connected on an individual basis.

Research Contribution

This research involves the semantic differences in data and database structures. On the *barriers* dimension this can be placed under the conceptual barriers. These are considered to be the most important barriers because they are concerned with the presentation and representation of concepts to use for enterprise business and operations (Ullberg et al., 2009). At the different *Enterprise Levels* this research covers the *service level*. Table 1-1 displays the set of subdomains of the interoperability research domain, and marks the subdomain of this research. We thereby contribute to this field of research as "a piece of knowledge is considered as relevant to interoperability if it contributes to remove at least one barrier at one level" (Chen et al, 2008). Removing the conceptual barriers at the service level could be performed by using any of the three previously described approaches. It depends on the organization's wishes and requirements which of these is most suitable. When choosing the integrated approach, our methodology assists in the process of transferring information from the current system to the new integrated system. When using the unified approach, the methodology identifies and solves semantic conflicts between the common meta-model and the local system. For the federated approach the methodology can be used to compare the local semantics of the two systems.

		Barriers		
		Conceptual	Technological	Organizational
Levels (concerns)	Business	*subdomain*	*subdomain*	*subdomain*
	Process	*subdomain*	*subdomain*	*subdomain*
	Service	**Research Contribution**	*subdomain*	*subdomain*
	Data	*subdomain*	*subdomain*	*subdomain*

Table 1-1: Research Contribution in the interoperability domain

Research Outline

The research is structured around the Design Science Research Process (DSRP) model by Peffers et al (2006) (Figure 1-2). We choose this model as it is not only consistent with earlier work (Archer, 1984; Takeda et al, 1990; Eekels and Roozenburg, 1991; Nunamaker et al, 1991; Walls et al, 1992; Rossi et al, 2003; Hevner et al, 2004), but also provides a nominal process for doing design science research.

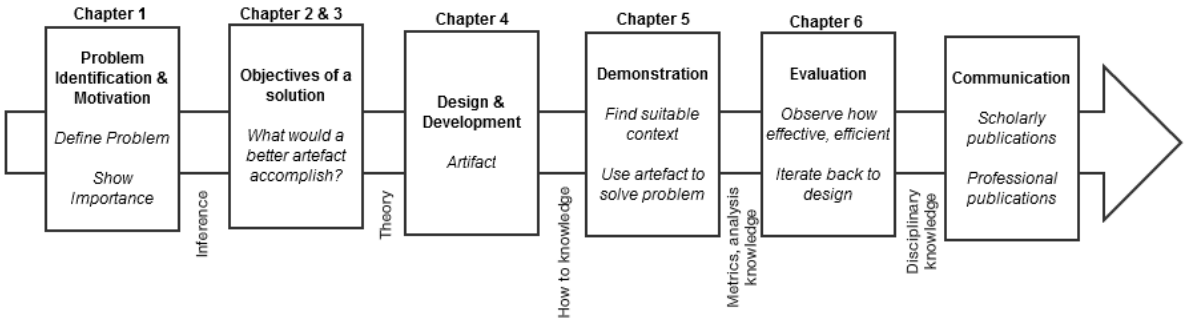


Figure 1-2: Design Science Research Process (Peffers et al, 2006)

In this chapter we defined the research question and objectives, identified the problem relevance and placed this research within the Framework for Interoperability. In the next chapter we look at the objectives of the artifact that is designed during this research. What is necessary for the methodology to be useful in practice? What do we know about rigorous methodology design? What exactly should the methodology be able to accomplish? We also introduce the research model that forms the basis for the methodology to develop. This part of the research is followed by a comprehensive literature review in chapter three, and describes the various concepts subject to this study. We start with a summary of the different meanings given to the term Interoperability and choose a definition to use in this research. When then continue with the different categorizations of semantic conflicts by different authors, and describe the findings for each of the subject areas.

In the fourth chapter we use the knowledge from the previous chapters to build the methodology. We come up with a new semantic conflict categorization that forms the basis of the method. We define the different stages to go through, their deliverables, and the tasks and techniques to produce them.

In chapter five we demonstrate the methodology by performing a case study. Results are then discussed in chapter six, and implications for the method explained. Finally, we present our conclusions and provide suggestions for further research.

2. Research Design

In this chapter we present our research design. We start by addressing the first research goal. We specify the definition of methodology, and the requirements of a useful methodology. The chapter then describes the concepts used in the construction of the methodology, and the research method we use to evaluate the artifact. The chapter ends with a graphical representation of the structure of this paper.

Objectives of the Methodology

In this paper the words method and methodology are often exchanged. The terms can be read as synonyms, referring to the following definition (Brinkkemper, 1996):

> "A method is an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systemic way in development activities with corresponding development products."

In the method development process, we borrow knowledge from the field of method engineering. With method engineering we refer to (Brinkkemper, 1996):

> "Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems."

As method engineering is a new research area in the field of interoperability, no significant research about the success factors for interoperability implementation methods has been published. However, plenty of research has been published about the requirements that methodologies should meet in the field of systems design. System Development Methodologies (SDMs) are, simply defined, a way to develop an information system (Roberts Jr. et al., 1998). The many similarities between the field of interoperability and systems design, provides the opportunity to make use of different SDMs studies, and see how they relate to our research field.

Catchpole (1986) states that "a methodology must be capable of representing the users' requirements in formal terms, and be capable of providing verification of the models constructed in order to check for any inaccuracies, inconsistencies or incompleteness". The verification can be performed in multiple ways such as group- and interview sessions, and scenario mapping with the end users.

Tozer (1984) states that "a methodology should be divided into a series of identifiable, logical stages, with the required outputs from each stage being rigorously defined". This is in line with our second research goals that aims at defining the steps to be taken. With each step, we will have to clearly define the output in terms of results and formal documents. Bantleman concludes after a survey of 150 development methodology users that "preferable the outputs from one stage of a methodology form the inputs to the next stage". So not only should each stage produce a clearly defined output, we want the output to be used in following stages.

Interoperability problems differentiate severely per case. Ralyte at al. (2008) argues that "interoperability is an emerging problem and hence we can only see and analyze the problems as they occur in their organizational and business contexts. This means that there can be no one solution to the problem, which can be captured in a single method." This problem is not limited to

the domain of interoperability. Chatzoglou (1997) points out that many authors suggest that there is no best methodology for all situations. The assumption is shared by Curtis et al. (1988) and Avison et al. (1988) who argue the one size fits all presumption, and state that “due considerations needs to be given to the contingencies of each development situation”.

This leads to a tight playing field for method development. On the one hand we want to increase the strength and problem solving capabilities of the method by providing concrete and well defined steps to be taken, while on the other hand preserving the applicability of the method in diverse situations. A proposed solution is offered in situational method engineering. A situational method is an information systems development method tuned to the situation of the project at hand (Harmsen et al, 1994). Engineering a situational method requires standardized building blocks and guide-lines, so-called meta-methods, to assemble these building blocks (Brinkkemper, 1996).

This means that the methodology must be capable of conflict detection, independently from the type of situation and systems at hand. The challenge involved was earlier identified by Park and Ram (2004): “The design of a semantically interoperable system environment that manages various semantic conflicts among different systems is a daunting task. It should provide the capability of detecting and resolving incompatibilities in data semantics and structures, as well as a standard query language for accessing information on a global basis”.

As explained in the first chapter, the methodology we develop is one of those building blocks in the interoperability method from Daclin et al (2008). We restrict ourselves to the semantic conflicts in this process, thereby providing one of the method fragments that can be used in situational methods (Brinkkemper, 1996).

When looking at user experience, we know that widespread adoption of interoperability will only be achieved with a methodology that does not require its users to have expert knowledge in the field of interoperability. Techniques should provide the means of expressing the users’ problems, and thus they should be easy to use, understand and learn (Tozer, 1984). We want to create a method that is understandable by information systems managers in any organization involved in multi-organizational networks. It provides concrete tools with a structured approach for execution.

Summarized, we come to the following list of methodology requirements:

- One output of the methodology should represent the user’s requirements in formal terms. This document has to provide the opportunity to check the models constructed for any inaccuracies, inconsistencies or incompleteness.
- The methodology needs to define several identifiable, logical stages, where the output of each stage is clearly defined
- The output from one stage preferably forms the input of the next stage.
- The methodology must be useful as a method fragment in situational method engineering. Therefore, it should be able to work together with many different other method fragments in the total interoperability process.
- The method should be easy to use, understand and learn.

Designing the Methodology

The logical structure of the methodology we propose is derived from the Information Engineering Methodology (IEM) Description Model described by Heym and Österle (1992) (Figure 2-1). According to the authors "the representation model provides a standard specification model for Information Systems Development knowledge in order to perform an engineering approach to methodology modeling". The model provides the perfect basis to start building our methodology, as it describes the key concepts that should be included, and how these are related to each other.

The model consist of several *stages* that each produce *deliverables* that form the input to other stages. Each stage is composed of one or more *tasks,* which are often subdivided into *subtasks*. Each task offers the user *practical guidelines* by providing *techniques* to produce the deliverables, and has associated *rules* and *conventions* for the representation of those deliverables. *Concepts* (such as entity types, attributes, and relationships) model the elementary components a technique deals with.

The center part of the model represents the construction part of this research. The goal is to design several stages, each addressing part of the semantic conflicts problem. For each stage we create a list of tasks that have to be performed. These tasks are generated by combining existing techniques and concepts as found in the literature study in Chapter three.

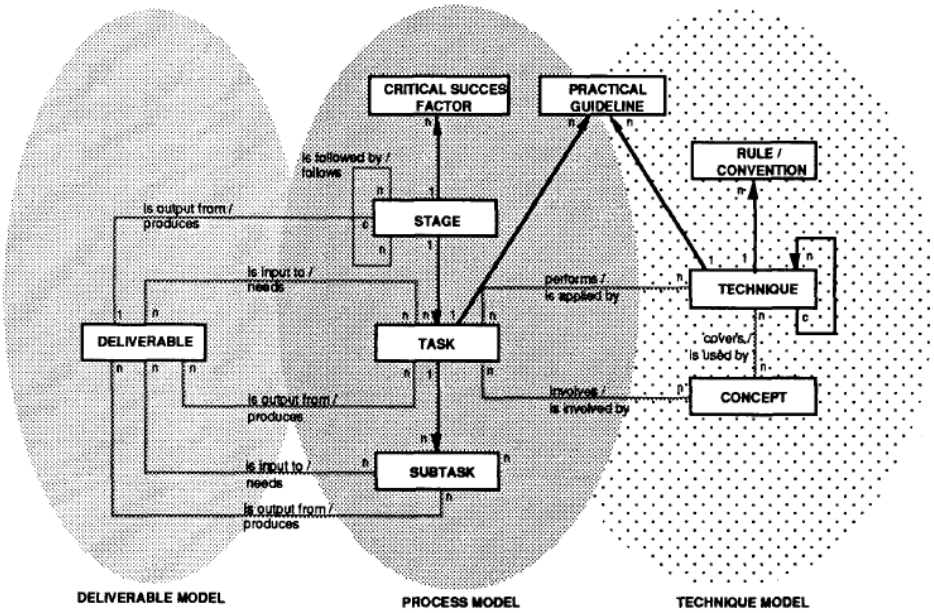


Figure 2-1: Information Engineering Methodology Description Model (Heym and Österle, 1992)

After the construction phase, we need to verify the validity of the constructed method. Because design is inherently an iterative and incremental activity, the evaluation phase provides essential feedback to the construction phase as to the quality of the design process and the design product under development (Hevner et al., 2004).

We discuss the methodology its effectiveness and efficiency by the evaluation framework from Pedersen et al (2000). The framework covers both the theoretical and the empirical dimension. The

theoretical dimension looks at the validity of the constructs used, and the logical structure in which the constructs are put together. Theoretical validity is further researched by means of an expert review to test the general belief in its usefulness. Two interoperability experts use their tacit knowledge to have a critical look at the methodology, and to point out its strengths and weaknesses.

Empirical validity is tested by applying the methodology in practice. This activity involves comparing the objectives of a solution to actual observed results from use of the artifact in demonstration (Peffers et al, 2006). As our goal is to create a methodology that can be used in real-world integration projects, we demonstrate its use by performing a case study at an existing integration project in the Netherlands. The project was chosen as it is exactly what the methodology is intended for, and because of the public availability of the system's documentation. In the case study we test how well the developed methodology responds to real-world situations and how it satisfies the requirements as defined earlier in this chapter.

The ultimate goal of the artifact is to make the interoperability project more efficient. Since we are in a situation where it would be infeasible to represent all means, ends, and laws, we are searching for a satisfactory solution, rather than the optimal solution. Hence, we construct a methodology that improves the process, not necessarily optimizes. After the design phase the method is tested in one single case study. Although testing the method in several different cases would provide better validation, due to time constraints we are performing only one iteration of the design process as illustrated in Figure 2-2.

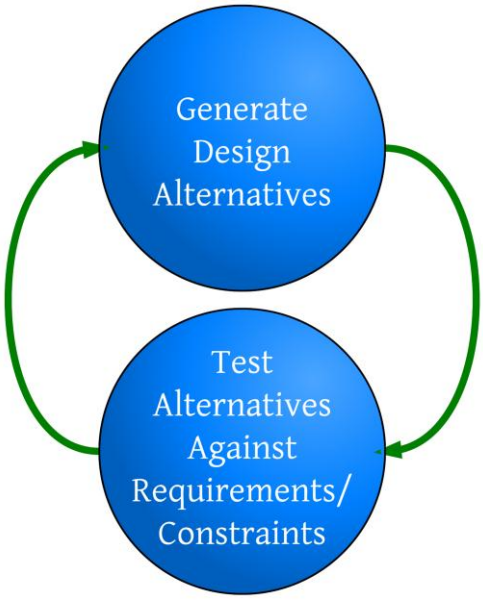
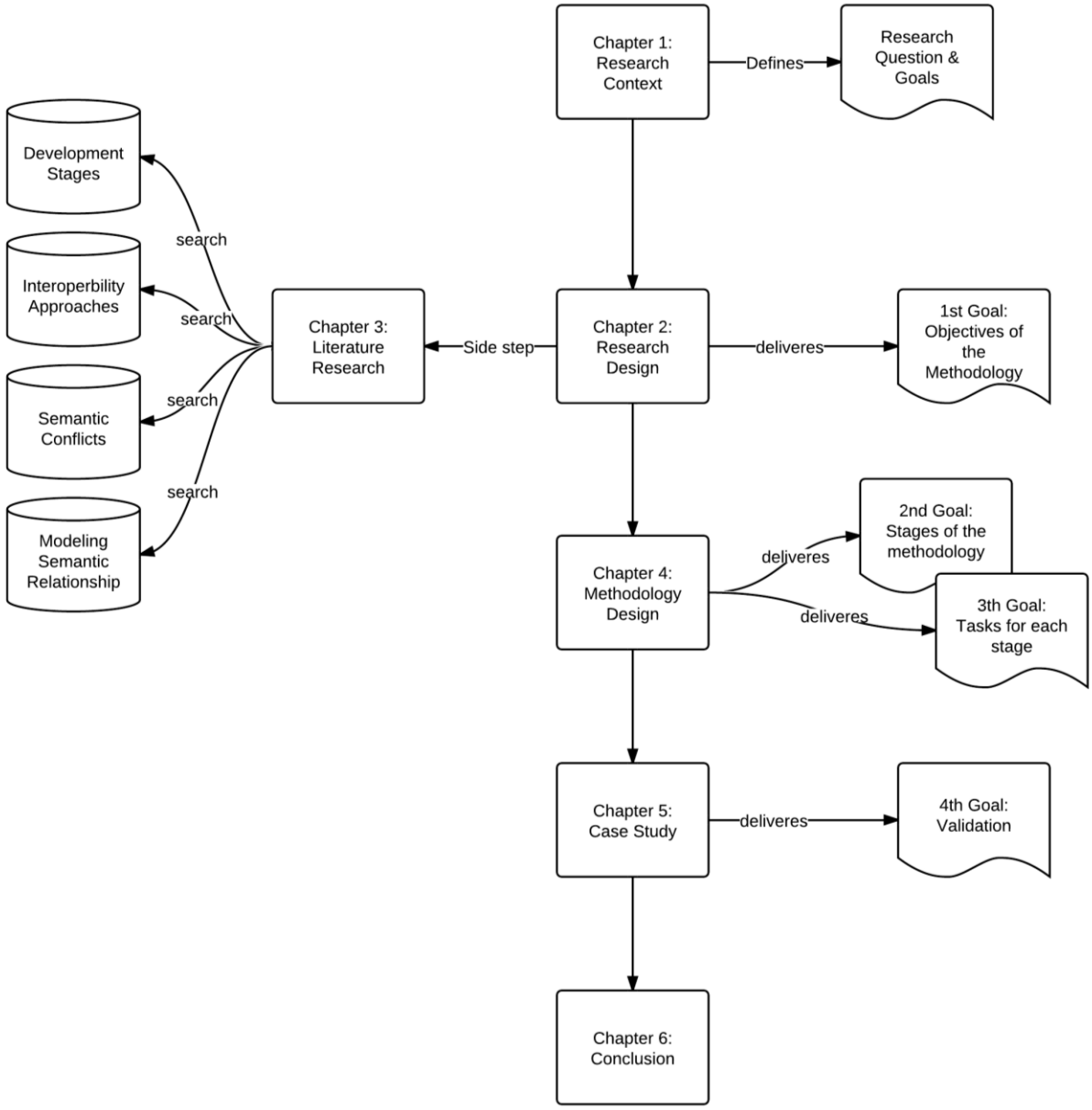


Figure 2-2: Design Process

Structure of this paper



3. Theoretical Background

This chapter summarizes the literature found by researching the current body of knowledge on interoperability. We categorize the used literature according to the four subject areas we searched for. We then summarize the findings, grouped by category, and describe the implications for our research.

Before we discuss the four subject areas, we need to define the concept of interoperability. Interoperability has been given many different definitions in existing research, differentiating in the scope and complexity of the term. Konstantas et al. (2006) use a broad definition to describe interoperability:

"The ability for a system or a product to work with other systems or products without special effort of the part of the consumer."

Their definition includes both systems and products in the description, where in the case of products one can think of a bolt and a mating nut. Because the nut is specially designed to fit onto the bolt, the products work with each other without special effort from the user. Although the definition provides a good understanding of the basic meaning of interoperability, the scope of this research will be limited to systems. Rothenberg et al. (2007) focus on the systems:

"The ability of distinct systems to communicate and share semantically compatible information, perform compatible transactions, and interact in ways that support compatible business processes to enable their users to perform desired tasks."

The definition clearly defines the purpose of interoperability. However, this research will not focus so much on the purpose and opportunities of interoperability, but will be more targeted at the problem space (semantic incompatibility) when trying to achieve interoperability. We therefore prefer the definition by Naudet et al. (2010) :

"An interoperability problem appears when two or more incompatible systems are put in relation. Interoperability per se is the paradigm where an interoperability problem occurs."

This definition fits perfectly with our research goal. When two independently developed systems exchange information and a semantic conflict occurs, we are thereby having an interoperability problem.

Design of Literature Study

The literature study follows the principles of a good literature study as defined by Webster and Watson (2002). Following their guidelines the literature search consists of three phases: (1) scan the top journals, (2) go backward, and (3) go forward. However, as they acknowledge, the top journals should be seen as a good starting point and one "should also examine selected conference proceedings". Since interoperability is a relatively new field in information systems research, there is not a lot of publication in the top journals yet. We are therefore not limiting our database search to the top journals.

We start by searching for literature that helps us define the various development stages of the methodology. We both look at the development stages for the total interoperability project, and to

the stages for a data integration project. Findings are compared and the results are used when building our methodology in chapter four.

Next we look at literature covering interoperability- and schema integration approaches. This information creates an understanding for the role our methodology plays within the interoperability project. We describe the three main approaches so that we learn how the selection of a specific approach changes the role of our methodology. This information is then used in chapter four where we describe how this changes the use of the methodology.

The third subject area covers the various categories of semantic conflicts. Here we learn what kind of semantic conflicts we can expect, so that we know what our methodology needs to identify.

The last subject area is about modeling semantic relationships. It describes how we can compare the semantics of two independently developed systems, and provides suggestions for the notation that can be used to formalize the comparison.

	Development Stages	Interoperability- and Schema Integration Approaches	Semantic Conflicts	Modeling Semantic Relationship
Batini & Lenzerini	X		X	
Chen et al.		X		
Daclin et al.	X	X		
El-Khatib et al.			X	
Fagin et al.		X		
Gagnon		X	X	
Goh et al.		X		
Haslhofer & Klas	X			X
Jamadhvaja & Senivongse		X		
Kim et al.			X	
Madnick		X		
Madnick & Zhu		X	X	
Naiman & Ouksel			X	X
Ouksel & Ahmed		X		
Park & Ram		X	X	
Ralyte et al.	X			
Ram & Park		X	X	
Ram & Ramesh	X	X		
Shahri et al.		X		
Sheth & Kashyap			X	X
Shvaiko & Euzenat		X		X

Table 3-1: Literature Classification

Development Stages

Daclin et al. (2008) take a macro view at the interoperability problem. They define a structured approach (Figure 3-1) for the total interoperability project that utilizes solutions from the different subject areas of the problem. The structured approach is divided into four stages:

1. Definition of objectives and needs
   Define the performance level targeted. Involves project planning, such as defining costs of the project, and evaluating the feasibility of the project. Requires the user to make a choice for one of the three interoperability approaches (*integrated*, *unified*, and *federated*).
2. Analysis of existing systems
   Identify actors, applications, and systems that are involved. Define the 'as-is' situation, then compare with the 'to-be' situation. Define the interoperability barriers to get from the as-is to the to-be situation.
3. Select and combine solutions
   Search and select available interoperability solutions for the barriers defined in step two.
4. Implementation and test
   Test the solutions selected in the previous stage, then implement, and evaluate the results. Compare the results with the performance level targeted.

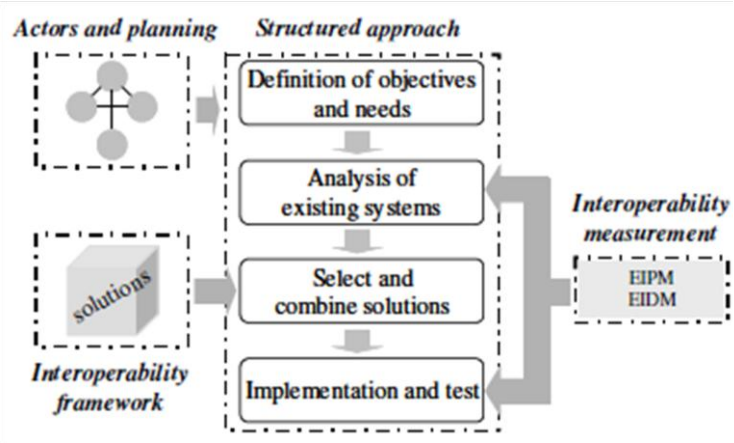


Figure 3-1: Structured approach to interoperability

The same basic structure the above methodology follows, can be applied to the various solutions that are selected during step three. This is demonstrated by Battini and Lenzerini (1984) who focus their research on the data integration aspect. They make a comparative analysis of methodologies for database schema integration. They conclude that "any methodology eventually can be considered to be a mixture of the following activities":

1. Preintegration
   The first stage involves an analysis of the different schemas subject to the integration project. The goal of this stage is to choose an integration strategy. What schemas will be integrated? Will integrating only portions of the different schemas satisfy the demands of the project? The stage also involves collecting assertions and/or constraints.
2. Comparison of the Schemas
   The next stage governs a comparison of the schemas involved. Goal of this stage is to identify possible data conflicts. Interschema properties may be discover while comparing schemas.
3. Conforming the Schemas
   Once the conflicts have been identified, it is time to resolve these problems, so that merging of the information stored is possible.
4. Merging and Restructuring

After solving all conflicts, it is time to give rise to some intermediate integrated schema(s). The intermediate result can be tested against:

a) Completeness and correctness: must represent the union of the application domain.
b) Minimality: concepts represented in more than one component schema must be represented only once in the integration schema.
c) Understandability: not only for the designer, but also for its end user.

Ralyte et al. (2008) defines eight stages of the ICT-development process. It basically covers the same concepts as the previously described approaches, but is divided into more different stages:

1. Feasibility Evaluation
2. Requirements Engineering
3. Analysis
4. Design
5. Development
6. Test
7. Deployment
8. Maintenance

Although all three approaches are slightly different, they all basically cover the same concepts and follow the same structure (Table 3-2).

<table>
<tr><th>Daclin et al. (2008)</th><th>Battini & Lenzerini (1984)</th><th>Ralyte et al. (2008)</th></tr>
<tr><td rowspan="2">Definition of objectives and needs</td><td rowspan="2">Preintegration</td><td>Feasibility Evaluation</td></tr>
<tr><td>Requirements Engineering</td></tr>
<tr><td>Analysis of existing systems</td><td>Comparison of the schemas</td><td>Analysis</td></tr>
<tr><td rowspan="2">Select and combine solutions</td><td>Conforming the schemas</td><td>Design</td></tr>
<tr><td rowspan="3">Merging and restructuring</td><td>Development</td></tr>
<tr><td rowspan="2">Implementation and test</td><td>Test</td></tr>
<tr><td></td><td>Deployment</td></tr>
<tr><td></td><td></td><td>Maintenance</td></tr>
</table>

Table 3-2: Development stages

Additionally we discuss the metadata mapping cycle by Haslhofer & Klas (2010). The cycle (Figure 3-2) starts with *mapping discovery* which is concerned with finding semantic and structural relationships between elements. The *mapping representation* phase is concerned with the formal declaration of the mapping relationships between the two schemas. The next phase, *mapping execution*, represents the execution of mapping specifications at run-time. The last phase in the cycle is concerned with the documentation that must provide information about the mappings made in the previous phases. This documentation makes it easier for future adjustments, necessary when one of the systems changes (i.e. versioning). The mapping maintenance also is the key for discovering new mappings from existing ones. If, for instance, schema A and schema B are connected, as well as schema B and C, we can make also create a mapping between schema A and C.

The steps in the mapping cycle are quite similar to the schema integration methodology by Ram & Ramesh (1999). Their methodology (Figure 3-3) starts with *schema translation*. During this phase each database subject to the integration project is translated into schemas using a common model. This could be an entity-relationship model or a class diagram. The objective of the next phase, *interschema relationship identification*, is to identify objects in the underlying schemas that may be related (i.e. entities, attributes, and relationships). Related objects should be classified according to their semantic relationship. After confirming the relationships by a designer/expert, an integrated schema is generated in the next step. The integrated schema represents the concepts found in both systems. Finally, the *schema mapping generation* takes place, where concepts from the involved systems are mapped to the integrated schema.

These last two steps can be performed by several different integration approaches, which will be discussed in the next section.

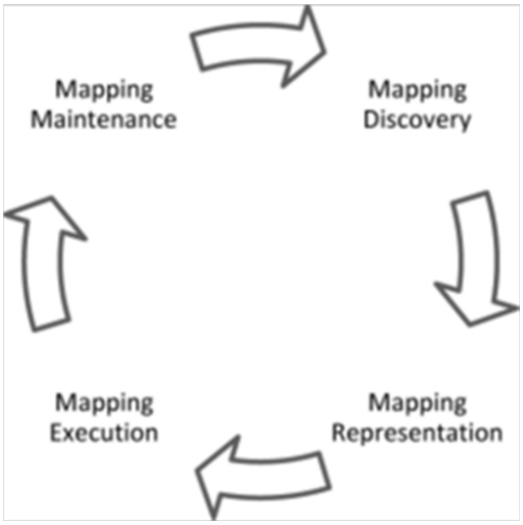


Figure 3-2: Metadata Mapping Cycle (Haslhofer & Klas, 2010)

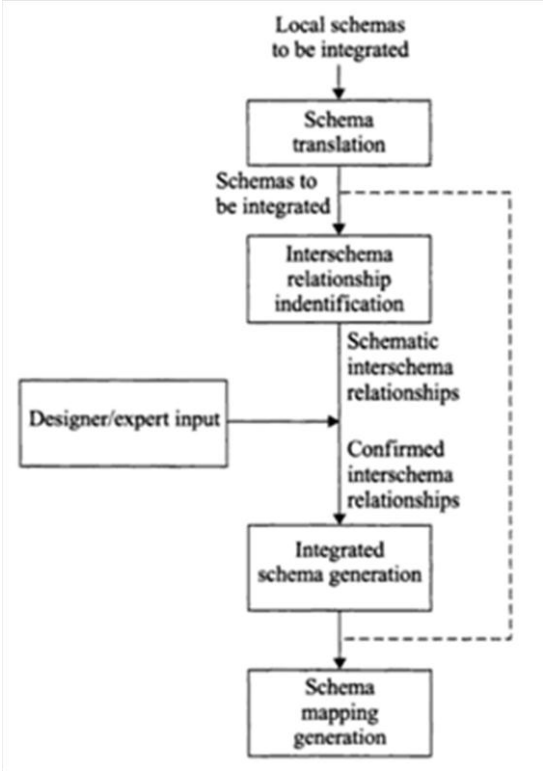


Figure 3-3:Schema Integration Methodology (Ram & Ramesh, 1999)

Interoperability & Schema Integration Approaches

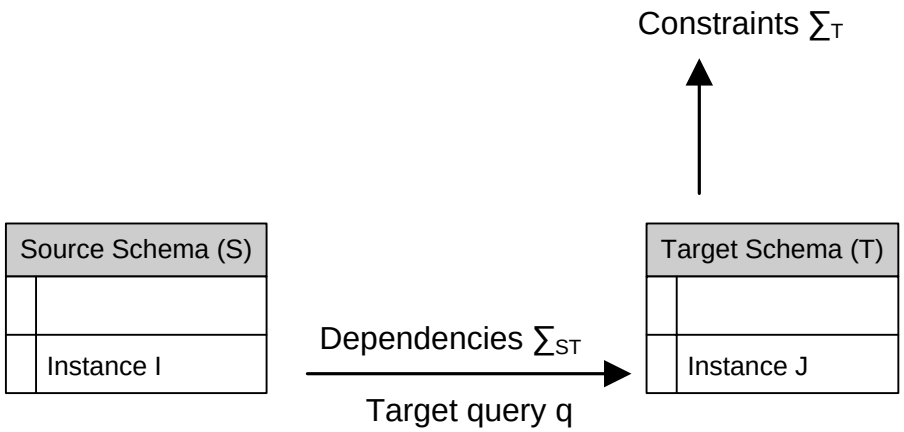
The difficulty of finding correspondences between schemas originates from the fact that the conceptual models used for data representation, do not capture the semantics of the data with enough precision (Shahri et al., 2008). One organization may use the concept 'Author' to describe the creator of a story, where another organization may use the term 'Writer' to indicate the same person. While both refer to the same real world entity, the information is stored with different labels in their databases, resulting in integration problems when systems are connected. The problems are solved by creating an interoperable environment. The Framework for Interoperability presents three approaches to accomplish semantic interoperability: federated, unified, and integrated. Each of these approaches will now be discussed.

Federated approach

The federated approach does not attempt to integrate data schemas, but facilitates information exchange. Each system remains independent, there is no common format. Using the federated

approach implies that no partner imposes their models, languages and methods of work. In the federated approach, each local database provides an export schema. Local database administrators can then use these schemas to define an import schema (Ram & Ramesh, 1999).

The federated approach can be modeled as having a source schema **S** and target schema **T**, assumed to be disjoint (Fagin et al., 2005). Constraints resulting from the independently designed schema **T** are represented by $\sum_T$ and a set of source-to-target dependencies as $\sum_{ST}$. Now if we have an instance *I* over schema **S**, we need an instance *J* over target **T** to satisfy $\sum_T$ while *I* and *J* together must satisfy $\sum_{ST}$.

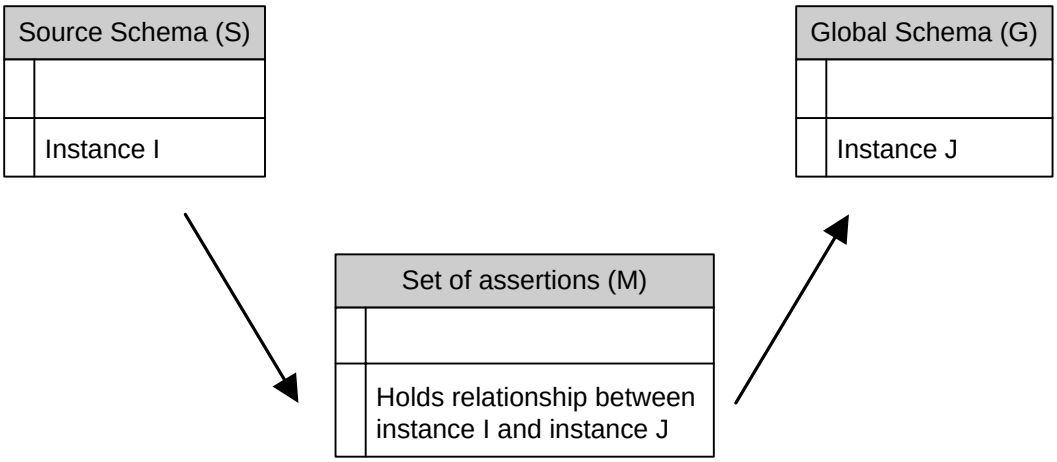


During the 1990s, this 'loose-coupling' approach gained a lot of attention and provides a lot of flexibility in adding and removing systems. However, this approach also "requires users to have intimate knowledge of the semantic conflicts between the sources and the conflict resolution procedures" (Madnick, 1999). This limits the scalability of loosely-coupled systems, as the knowledge grows and changes when more sources join the system.

Unified approach

The unified approach attempts to create mappings between semantically equal data sources. This approach respects the differences in ontologies used by different organizations. It aims at providing ways to correctly connect the data from different systems, while not enforcing organizations to make changes in their daily operations. This is often performed by creating a federated schema and mapping the various data sources to the right component in the schema. This way a common format is developed, but only on the meta-level. The meta-model is not an executable entity, but provides a mean for semantic equivalence to allow mapping between models.

The unified approach is modeled by a source schema S, a global schema G, and a set of assertions relating elements of the global schema to elements of the source schema. One could see the resemblance with the federated approach, where we can define M as $\sum_{ST}$ and G as a combination of T and $\sum_T$.



An important difference between the federated- and unified approach is that target schema T in the federated approach is developed independently and comes with its own set of constraints, while global schema G in the unified approach is "commonly assumed to be a reconciled, virtual view of a heterogeneous collection of sources and, as such, it is often assumed to have no constraints" (Fagin et al., 2005). This implies that the unified approach is not designed to be independent of particular schemas and applications (Ram & Ramesh, 1999).

The unified approach requires conflicts to be identified and reconciled a priori. Although this approach provides good support for data access, its solution does not scale-up efficiently given the complexity involved in constructing and maintaining a shared schema for a large number of, possibly independently managed and evolving, sources (Madnick, 1999).

Integrated approach

The integrated approach requires a common format for all models. This is normally achieved by developing a common ontology that forms the basis for each involved party to build systems and elaborate models.

Ontology development tries to define the specification of concepts more accurately. It represents the real world concepts and how they are connected to each other. Since business needs differ per organization, a decision of what the ontology does, and does not, represent has to be made. Second, the concepts have to be named and relationships between concepts have to be drawn.

Where information systems are connected for data sharing, the ontologies have to be merged into one common model. Every organization involved in the project will have to make changes in their organization in order to use the newly created ontology. When merging the ontologies, there is potential for inconsistencies and the ontology designer needs to make complex decisions in various steps of the process (Shahri et al., 2008). Hence, the process can only be semi-automated and no algorithmic solution exists (Noy & Musen, 2003).

Other approaches

As indicated in the description of the three approaches, each has its own advantages and drawbacks. Several attempts have been made to develop a framework that combines the advantages of each approach, and minimizes the drawbacks. Two of these frameworks are now shortly described.

CREAM Framework

The Conflict Resolution Environment for Autonomous Mediation (CREAM) framework as developed by Park and Ram (2004) presents “a generic approach to achieving semantic interoperability at both the data and the schema levels”. The framework makes use of a federated schema, but to a large extend automates the way mappings are made with the use of Semantic Conflict Resolution Ontology (SCROL) (Figure 3-4). SCROL exists of concepts based on commonly found semantic conflicts. Each of these concepts, for example ‘Area’, has several instances as a child. In this case it could be ‘Square meter’ and ‘Acre’. With the CREAM framework, involved organizations can map their concepts to the federated schema and indicate the concept’s instance used in their organization. The SCROL mechanisms will make sure that when two systems are communicating, they are essentially speaking in the same language.

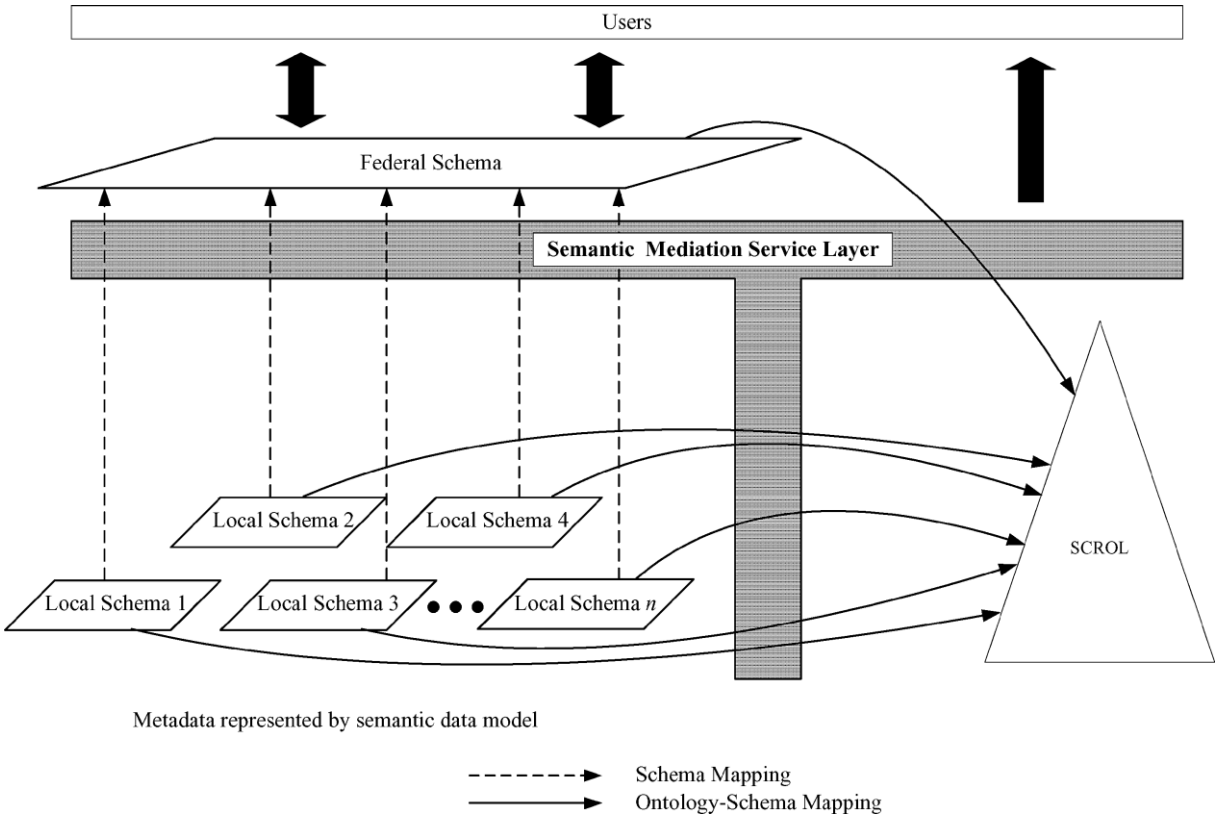


Figure 3-4: Semantic Conflict Resolution Ontology (Park and Ram, 2004)

Context Interchange Framework

Context Interchange (COIN) is a mediator-based approach for achieving semantic interoperability among heterogeneous sources and receivers (Goh et al., 1999). It basically combines the federated- and unified approach. The COIN Framework (Figure 3-5) consists of three components: (1) domain model, (2) elevation axioms, and (3) context axioms. The domain model defines the semantics, covering the application domain of the systems to be connected. The elevation axioms defines the semantic objects used in each source schema, and the context axioms defines the interpretations of the semantic objects for each source schema.

When a query is submitted to the system, it is intercepted by the *Context Mediator*. This mediator uses the elevation- and context axioms to transform the query into an optimized query plan, taking into account the differences between the systems. “The provision of such a mediation service

requires only that the user furnish a logical (declarative) specification of *how data are interpreted* in sources and receivers, and *how conflicts, when detected, should be resolved*, but *not what conflicts exists a priori* between any two systems." (Goh et al., 1999).

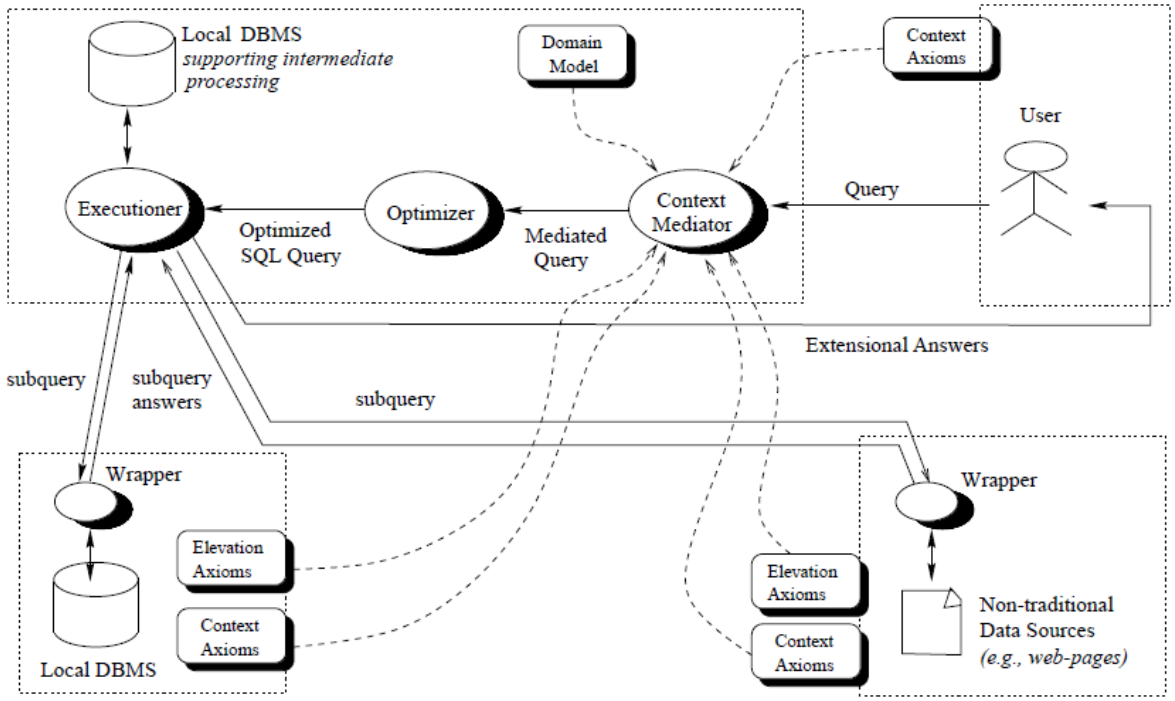


Figure 3-5: Context Interchange Framework (Goh et al., 1999)

Semantic Conflicts

This part of the literature study aims to identify all of the semantic conflict types. We study various research paper that categorize semantic conflicts, so we learn which conflicts our methodology should be able to identify and resolve.

Park and Ram

Park and Ram (2004) identify two different levels at which semantic conflicts can occur (Figure 3-6). Data-level conflicts occur because of multiple representations and interpretations of similar data. This level is then further divided into data-value conflicts, data representation conflicts, data-unit conflicts, and data precision conflicts. The second level of semantic conflicts is the schema-level. Schema-level conflicts are characterized by differences in logical structures and/or inconsistencies in metadata (i.e., schemas) of the same application domain. This level is further divided into naming conflicts, entity-identifier conflicts, schema-isomorphism conflicts, generalization conflicts, aggregation conflicts, and schematic discrepancies. Each of the data- and schema level conflicts will now be explained by the following example.

Organization A is a shoe manufacturer and the supplier of wholesale organization B. Both organizations agree to increase the efficiency of the supply chain by sharing data about stock levels, shipping information, and sales. As A will have a live view of the sales at B, A can instantly adjust production levels to make sure they won't run out of stock. Simultaneously, B will be able to request shipping information from A, so they get better insight into the delivery date and thus they can

better inform their customers. When connecting the data from the two systems, several problems appear:

The first problem arises from the difference in understanding of the sales price. The wholesaler B stores the sales price with the 19% VAT included, while manufacturer A stores the sales price excluding VAT. So while both organizations use the same concept, the value of the concept differs. This is defined as being a *data-value conflict*.

The wholesaler wants to use the shipping information from its manufacturer to estimate the delivery date. However, in the wholesaler's system dates are stored as '01012010' while the manufacturer saves dates as 01-Jan-2010. So while both use the same concept and the same value, there still is a mismatch between the two systems. This is referred to as being a *data representation conflict*. In the shipping information there also is data stored about the size of the shoes. But as organization B is located in Europe and uses European sizes, A originates from the U.S. and uses American sizes. Connecting the systems leads to a mismatch, or a *data-unit conflict*.

In the situation where the two companies agree to both be using European sizes, we might create a new problem, being a *data precision conflict*. This arises when one organization uses a half-point scale to store size information (such as 42.5) while the other uses a 1-point scale (so this becomes 42 or 43).

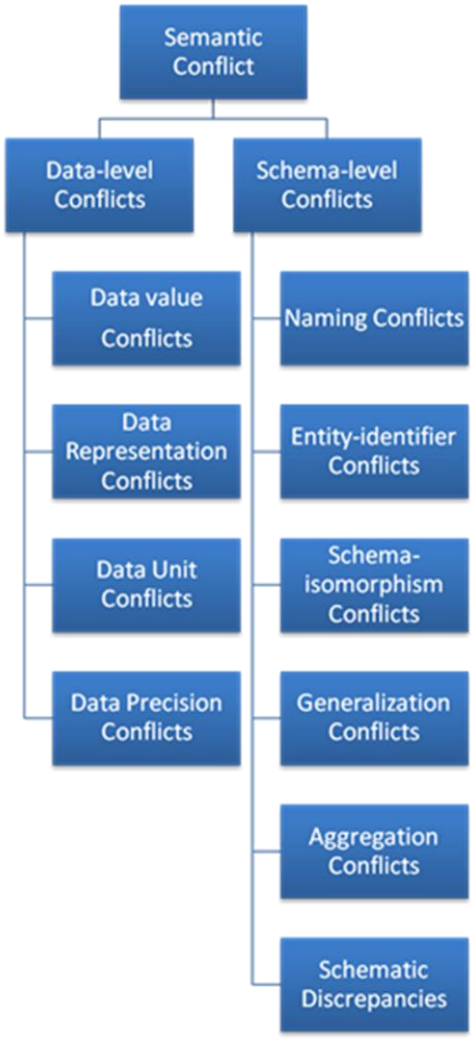


Figure 3-6: Semantic Conflict Categorization (Park and Ram, 2004)

On the schema-level we can see a bunch of other semantic conflicts. While both organizations store the same product-number data in their system, company A calls this the Product ID, while B calls it the Item Number. Hence, we are having a *naming conflict.*

We can see an *entity-identifier conflict* when the sales data is exchanged. Organization B assigns a primary key to every sale stored within their system. When this sales data is copied to the information system of A, a different primary key gets assigned to the same sale. As a result, it becomes hard to match the data at a later stage since the unique identifier does not match. Another problem is the dissimilar set of attributes, by which each sale is described. Company B might save information about the payment method being used for the transaction, while A does not use this attribute to describe a sale. This conflict is being described as a *schema-isomorphism conflict*. Also, while B saves the first name and surname of the customer separately, A chooses to aggregate these into the one attribute 'name'. This is called an *aggregation conflict*.

When different design choices are made for modeling related entity classes, we are having a *generalization conflict*. In our example this is the case when the manufacturer produces shoes for every of the four seasons, and their information system categorizes each shoe accordingly. The wholesaler however is located in the southern part of Europe and, because of its mild climate, only defines two seasons. When the season category information is exchanged, A will have to define which instances of season two to assign to fall, and which to winter.

The last semantic conflict on the schema-level is called a *schematic discrepancy*. This is defined as "the situation where the logical structure of a set of attributes and their values belonging to an entity class in one database are organized to form a different structure in another database" (Park and Ram, 2004). In the running example this would be the case when one company would define the customer name as an attribute of sales, while the other defines it as a different entity that is linked to from the sales data.

Madnick and Zhu

Madnick and Zhu (2005) take a different approach when categorizing semantic heterogeneities. They divide semantic conflicts in the following categories:

- Representational heterogeneity

The same concept can have different representations in different sources and receivers. The representational conflicts are similar to the data representation-, data unit-, and data precision conflicts categories from Park and Ram (2004).

- Temporal representational heterogeneity

A concept referred to by one source can have different representations over time. For example, the introduction of the Euro as legal tender in the Netherlands changed the representation of the concept 'Price' within the same source from Gulden to Euro. If not addressed properly, this would mean that when one makes a database query to get company profits over the last twenty years, he would get an incorrect comparison between the periods 1991-2001 and 2001-2011.

- Ontological heterogeneity

Different choices can be made for the representation of the same concept. This problem is often experienced in the field of accounting. Financial concepts such as the P/E (stock price / earnings) ratio is defined in various different meanings. Some would use earnings for the last 12 months, some would take earnings for the last calendar year. These different choices lead to a different value for the P/E ratio, while the concept is the same. This category is similar to the 'data-value conflict' category from Park and Ram (2004).

- Temporal ontological heterogeneity

In the same source, the same concept can have different meaning over time. This is often the result of the context in which it is being used. The term Gross Revenue may have different meaning for tax authorities then for internal usage (including/excluding taxes).

- Aggregational ontological heterogeneity

There may be a mismatch of what is included in a certain concept. For instance, the marketing department may separate profits from product A and product B, while the accounting department uses profits from product segment 1 (including product A & B).

El-Khatib et al.

El-Khatib et al. (2000) developed a framework for classifying the different types of heterogeneities. They identify six main categories (Figure 3-7) with a number of subcategories.

1. Naming Heterogeneity

This is the same category as the 'Naming Conflicts' in Park and Ram (2004). Interesting is the distinction in (1) naming synonyms and (2) naming homonyms. Both of them can refer to attributes (same field in database) or relations (table names in database). In the case of naming homonyms we can also have an attribute-relation homonym, which means that a certain attribute from Table A in database 1 has the same name as some table in database 2.

2. Relational Structure Heterogeneity

When two tables in two different databases have the same name, but are described with a different number of attributes, we classify this as a relational structure heterogeneity. This could be compared to the category schema-isomorphism conflict from Park and Ram (2004).

3. Value Heterogeneity

This category includes the problems seen as Data-level conflicts in Park and Ram (2004). Interesting is the distinction in numeric-numeric, string-string and numeric string. Each of these require a different form of conversion when exchanging data. Furthermore, the fourth subcategory labeled as Structures could be compared to the aggregation conflict from Park and Ram (2004). The fifth subcategory Incomplete Information refers to the definition of NULL, which could have different meanings such as unknown, not applicable, or unavailable.

4. Semantic Heterogeneity

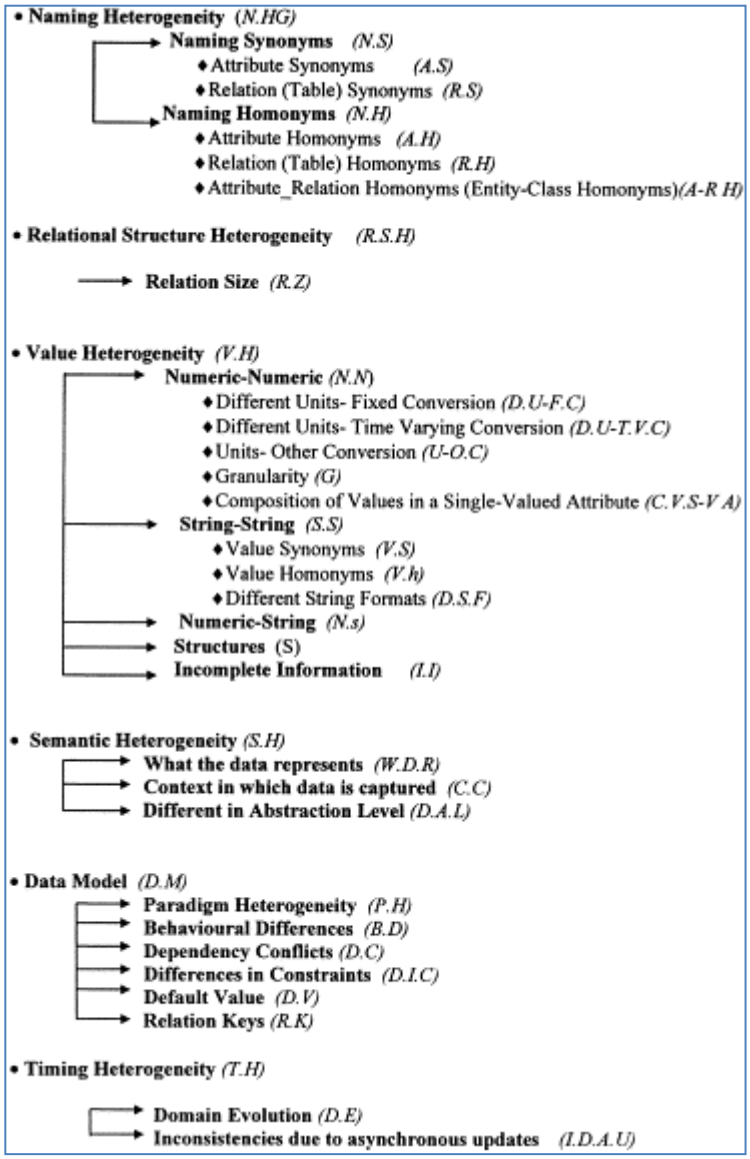


Figure 3-7: Framework for classifying heterogeneities (El-Khatib et al., 2000)

Semantic heterogeneity comes from misunderstanding of what the data represents. There can be confusion of (1) what the data represents (phone number could be mobile number, or home number), (2) context in which data is captured (in case of a blood pressure value one would like to know how this has been measured to determine the error range), (3) difference in abstraction level (comparable to a generalization conflict).

5. Data model heterogeneity

Subdivided into (1) paradigm heterogeneity (a database may be relational, hierarchical, object-oriented), (2) behavioral differences (different insertion/deletion policies), (3) dependency conflicts (1:1 relation in one database, 1:n in other), (4) differences in constraints (a birthday field could have a different date range amongst databases), (5) default value (current date or NULL), and (6) relation keys (conflicts in primary keys).

6. Timing heterogeneity

This refers to changes applied to data over time, often the result of a change in data semantics. A good example is when a country changes its national currency, influencing the meaning of the database field 'Price' over time.

Naiman and Ouksel

Naiman and Ouksel (1995) use a two dimensional set to classify semantic conflicts in heterogeneous database systems. The two dimensions are:

- *Naming*: The naming dimension refers to the names of the objects, attributes, or instances. This dimension can be either (1) synonyms, (2) homonyms, or (3) unrelated. The relationship is said to be commutative, so for instance, "*x* is a *synonym* of *y*".
- *Abstraction*: Refers to the abstraction relationship between the two schematic elements. This dimension can have the following values: (1) class, (2) generalization, (3) aggregation, or (4) computed function. The relationship is directed from *x* to *y*, so for instance, *x* is a *generalization* of *y*. The value *class* is assigned "when each term refers to some class of schematic elements, and those classes map to either the same set of instances or to disjoint sets of elements". A *generalization* refers to the case where one element has a lower abstraction level then the other (one is a superset of the other). The value *aggregation* refers to a situation in which an object in one database refers to a group of objects in another database. The last value, *computed function*, refers to an incompatibility "where there is no direct path from one object (or attribute) to another, but the object can be derived or approximated using some function or expert knowledge".

Combining the two dimensions results in twelve categories of semantic conflicts (Table 3-3).

		Abstraction			
		class	*generalization*	*aggregation*	*Computed function*
Naming	*Synonym*	1	2	3	4
	Homonym	5	6	7	8
	Unrelated	9	10	11	12

Table 3-3: Semantic Conflicts Dimensions (Naiman and Ouksel, 1995)

Sheth and Kashyap

Sheth and Kashyap (1992) classify structural incompatibilities due to heterogeneity in five main categories (Table 3-4). Each main category is further subdivided in several subcategories. Next to each subcategory, the typical semantic proximity is written in italic.

Domain incompatibilities arise when two objects have differing definitions of semantically similar attribute domains. The naming conflict is either seen when two semantically alike attributes have different names (synonyms), or when two semantically unrelated attributes are sharing the same name (homonyms). Data representation conflicts occur when two semantically similar attributes have different data types or representations (such as integer vs. string). Data scaling conflicts result from the use of different units and measures (such as currencies). When the attributes use different precision in their measures/units and there may not be one-one mapping between the values, we are defining this as a data precision conflict. A default value conflict is experiences when the default value for two semantically alike attributes is different. Finally, an attribute integrity constraint conflict is said to be the case when data constraints for both attributes are different.

While the previous category was related to the attributes of two objects, the entity definition incompatibility category relates to the entity level. A database identifier conflict occurs when two semantically related entities are identified by a different index in each database. A naming conflict is similar to that of the domain incompatibility category, but is now relating to the naming of entities. Union compatibility conflicts occur when the set of attributes are semantically unrelated in such a way that a one-one mapping is not possible between the two sets of attributes. We define a schema isomorphism conflict as a situation in which semantically similar entities have different number of attributes. In the case where one entity has a missing attribute, we define this to be a missing data item conflict.

- Domain Incompatibility
  - Naming Conflicts
    - Synonyms
    - Homonyms
  - Data Representation Conflicts
  - Data Scaling Conflicts
  - Data Precision Conflicts
  - Default Value Conflicts
  - Attribute Integrity Constraint Conflicts
- Entity Definition Incompatibility
  - Database Identifier Conflicts
  - Naming Conflicts
    - Synonyms
    - Homonyms
  - Union Compatibility Conflicts
  - Schema Isomorphism Conflicts
  - Missing Data Item Conflicts
- Data Value Incompatibility
  - Known Inconsistency
  - Temporary Inconsistency
  - Acceptable Inconsistency
- Abstraction Level Incompatibility
  - Generalization Conflicts
  - Aggregation Conflicts
- Schematic Discrepancy
  - Data Value Attribute Conflict
  - Attribute Entity Conflict
  - Data Value Entity Conflict

Table 3-4: Structural incompatibilities due to heterogeneity (Sheth and Kashyap, 1992)

The third main category, data value incompatibility, results from the values of data present in different databases. Known inconsistencies are recognized ahead of time. In some cases one database is considered to be more reliable than the other, and thus is set to overrule inconsistent data. Temporary inconsistencies occur when one of the databases stores obsolete information. Since inconsistencies are of a temporarily nature, the objects may be said to be eventually semantically equivalent. When data inconsistencies are within an acceptable range, we classify this as an acceptable inconsistency.

When two entities are represented at different levels of abstraction in two different databases, we classify them as having an abstraction level incompatibility problem. This can be either a generalization conflict or an aggregation conflict.

The final category is the schematic discrepancies problem. This is defined as the situation in which data from one database corresponds to metadata in another. In the case of a data value attribute conflict, the value of an attribute corresponds with an attribute. With an attribute entity conflict, the same entity is modeled in one database as an attribute, and as a relation in the other. Another possibility is a data value entity conflict, where the value of an attribute corresponds to a relation.

Kim et al.

Kim et al. (1993) classify techniques to resolve semantic conflicts. Each of the categories will now be described:

1. Renaming Entities and Attributes
   Used to resolve problem with synonyms and homonyms.
2. Homogenizing Representations
   a. Expressions
      Used to resolve problems when different expressions are used to represent the same data. A static lookup table that defines the isomorphism (mapping) solves the problem.
   b. Units
      An arithmetic expression is used to convert a numeric value in one unit to another. For example, one could convert pounds into kilograms. There may be some loss in accuracy during conversion.
   c. Precision
      A mapping may be defined to resolve problems with different precision of data. This implies a loss of data, as we convert from a more precise code to a less precise one.
3. Homogenizing Attributes
   a. Type Coercion
      Solve problems with different domains (types) for semantically equivalent attributes. For example, from an integer to a float.
   b. Projection of Aggregation Hierarchy
      Used to resolve problems when an attribute in database A refers to a separate entity, while in database B a similar attribute stores a value. The solution is to find an attribute in the separate entity in A that holds similar value to the attribute in database B.
   c. Default Values
      A mediator should replace the default value when communicating between the two systems, so that the receiver gets the appropriate default value. However, the conversion should not be propagated when inserting or updating a value in the original database.
   d. Attribute Concatenation
      Resolves problems when an attribute in database A, of type string, refers to two or more attributes in database B, also of type string. Simply solved by combining attributes when querying database B.

4. Horizontal Join
    a. Union Compatible Join:
        i. For no structural conflicts
        Integrates entities from several databases. In this case there is no structural conflict, so that the entities from different database can simply be united.
        ii. For missing attributes
        In this case there is a difference in the number of attributes amongst the entities in the different databases. There are three ways to solve this problem. First is to coerce nonexistent attributes to NA (not available). Second is to neglect the extra attributes when the database with fewer attributes queries the other. Third is to model the entity with fewer attributes as the superclass of the other, provided that the all entities being integrated are related by the inclusion relationship.
        iii. For missing attribute with implicit value
        Simply setting the missing attribute with the implicit value when querying the database will solve this problem.
    b. Extended Union-Compatible Join:
        i. For entity inclusion
        In this case two entities are related as they have a similar inclusion relationship. For example, in one database entity B is a subclass of entity A, while in the other database entity C is a subclass of entity A. As entity B and C are both subclasses of entity A, they are related. In this case we can integrate both subclasses into the metaschema.
        ii. For attribute inclusion
        Similar to the entity inclusion problem, but at the attribute level.
5. Vertical Join
    a. For Many-to-Many entities
    A concept may be decomposed into a number of entities. When differently decomposed entities are merged into one single entity in the metaschema, we need to perform a many-to-many join.
    b. For entity-versus-attributes
    This is experienced when a data-modeling concept is represented as an entity in one database, and as a set of attributes in another. This can be solved by splitting an entity in multiple parts, or by integrating the attributes into a single entity.
6. Mixed Join
    This is a combination of a horizontal and vertical join. The problem is resolved by combining solutions from both categories.
7. Homogenizing Methods
    This problem occurs when an entity in one database has a method that the other has not. This can sometimes be solved by treating methods as attributes.

Modeling Semantic Conflicts

Haslhofer and Klas (2010) define metadata mapping as “a specification that relates the model elements of two metadata schemes in a way that their schematic structures and semantic interpretation is respected on the metadata model and on the metadata instance level”. In other

words, the model defines the structural and semantic relationships between the elements of the schemas and their content values.

Figure 3-8 represents the main elements of such a metadata mapping. The model includes two independently designed schemas, schema S^s and schema S^t. Each schema consists of a set of elements, $e^s \in S^s$ and $e^t \in S^t$. Relationships between the two schemas are represented by mapping M, where each mapping relationship is represented by a mapping element $m \in M$. The mapping elements include a mapping expression $p \in P$ that describes the semantic relationship. An appropriate function $f \in F$ supports reconciliation on the instance level.

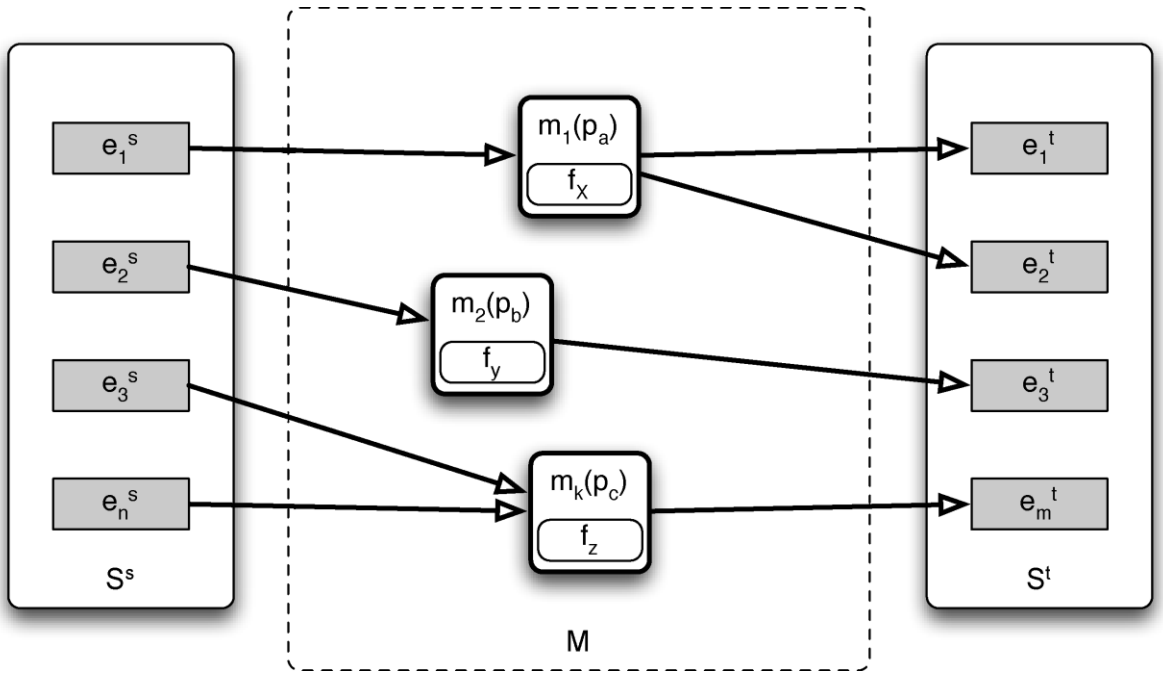


Figure 3-8: Meta data mapping (Haslhofer and Klas, 2010)

Shvaiko and Euzenat (2005) provide a formal notation for the mapping elements, defined as a 5-uple: ‹*id, e, e', n, R*› where

- *id* is a unique identifier given to the mapping element
- *e, e'* is de specific set of entities that are mapped together, one from each schema/ontology. An entity could for instance be a table, XML element, property, or class.
- *n* is the result of some mathematical measure that presents the confidence measure for the correspondence between *e* and *e'*.
- *R* is the relation holding for the specific set of elements. A relation could be *equivalence, more general, disjointness,* or *overlapping.*

The notation is quite similar to the notation by Naiman and Ouksel (1995). In the previous section we introduced the classification of semantic conflicts by these authors. A third dimension is added to the classification to assist the schematic mapping between the two elements. It presents the structural relationship between the two elements, which is a pair of the values *object*, *attribute*, or *instance*.

For an example of the classification, we take the following two subschemas:

Local Subschema:

MONTHLY-SALESLOG(Salesman, Item#, Month, Sales)

Remote Subschema:

ORDER(Date, Qty, Product#, Customer, Salesman)
PRODUCT(Product#, Current-Price)
WIP-INV-ITEM(Item#, Qty-on-Hand,Cost)

With the assumption that MONTHLY-SALESLOG.Item# corresponds to ORDER.Product# we can now write down the following conflict classification:

Assert [MONTHLY-SALESLOG.Item#, ORDER.Product#]
(synonym, generalization, (attribute, attribute))

Sheth and Kashyap (1992) developed a notation to present the semantic proximity between to elements, described as a 4-tuple between two objects O_1 and O_2:

semPro(O_1, O_2) = <Context, Abstraction, (D_1, D_2), (S_1, S_2)>

D_1 and D_2 refer to the domain of respectively O_1 and O_2. S_1 and S_2 refer to the state of the objects. The term context refers to the context in which a particular semantic similarity holds. This should not be confused with the domain. The context is associated with a group of objects, while the domain is associated with the description of an object. Thus, the context is a named collection of the domains of objects.

The context can be determined to be one of the following:

- ALL: the semPro notation is defined with respect to all possible context.
- SAME: the semPro notation is valid for the same context. The context has to be defined explicitly.
- SOME: the semPro notation is defined with respect to more than one context. Each context must be defined individually or collectively in an instance of semPro.
- SUB-CONTEXTS: the semPro can be defined in a previously defined context that is further constrained. The subcontext must be specified in a semPro instance.
- NONE: the objects under consideration do not have any useful semantic similarity under any context.

The abstraction in the semPro notation “refers to a mechanism used to map the domains of the objects to each other or to the domain of a common third object”. A few useful abstractions are:

- ALL (Total 1-1 value mapping): for every value in the domain of one object, there exists a value in the domain of the other object. Also, the value correspond.
- NONE: indicates that there is no mapping defined between two semantically related objects.
- NEG: indicates there is no mapping possible between the objects.
- ANY: indicates that any abstraction (such as the ones mentioned below) can be used to define a mapping between the objects.
- Partial many-one mapping: not every value in one domain is mapped to the other. Also, some values might be mapped with many values in the other domain.

- Generalization: one domain can be generalized/specialized the other, or both can be generalized/specialized to a third domain.
- Aggregation: this can be seen as a partial, 1-1 mapping. So the value in one domain can be seen as a subset of the value in the other domain.
- Functional Dependencies: can be seen as a partial, many-one mapping between the cross-products of the domains of the determining objects and the cross-product of the domains of the determined objects.

The semantic proximity between two objects is classified in table 3-5.

		Abstraction		
		NONE	*ANY*	*ALL*
Context	*NONE*	Semantic Incompatibility		
	SAME/SOME	Semantic Resemblance	Semantic Relevance	
	ALL		Semantic Relationship	Semantic Equivalence

Table 3-5: Semantic Proximity (Sheth and Kashyap, 1992)

We briefly discuss each classification:

Semantic Incompatibility
When there is no semantic similarity between two objects, we speak of semantic incompatibility.

Semantic Resemblance
The weakest measure of semantic proximity is semantic resemblance. In this case two objects cannot be related to each other by any abstraction in any context, but they have the same roles in their respective context(s).

Semantic Relevance
When two objects can be related to each other using some abstraction, but limited to the same context, we speak of semantic relevance. In this case the relation between the objects does not exist in a different context.

Semantic Relationship
If two objects are related to each other in any context using some abstraction, we define this as a semantic relationship. In this case the aggregation or generalization is true in any chosen context.

Semantic Equivalence
When there is a total 1-to-1 value mapping possible between the two classes under any given context, we define this as semantic equivalence.

Key Findings

In this chapter we researched literature in four different subject areas. The first section studies literature describing the various development stages in existing interoperability- and data integration methods. The stages in the different methods largely overlap with each other, and can basically be summarized into (1) objectives setting, (2) analysis of existing systems, (3) comparison of existing

systems, and (4) conforming the systems. We use this knowledge to construct the stages for our own methodology in the next chapter.

The second subject area studies the different interoperability- and schema integration approaches. We discuss the three approaches from the Interoperability Framework and their differences. From this we learn how each approach has a different impact on the systems comparison, so that we can anticipate this when constructing the methodology. We also discussed two approaches that try to combine the advantages, and minimize the disadvantages of the different interoperability approaches.

The third subject area we studied is the types of semantic conflicts that can occur between two or more systems. From the many categorizations we found we will construct our own in the next chapter, so we are confident to have included all common conflicts.

In the fourth and last subject area we studied literature about ways to model and notate semantic relationships. The techniques found will be used to present the findings from our methodology in a graphical way.

4. Methodology Design

The Information Engineering Methodology (IEM) developed by Heym and Österle (1992) provides the framework for the methodology creation process. Our goal is to create a series of stages that help the user to identify, classify, and model the semantic problems that may be encountered in an information integration project. We then explain what these results mean, and how they can be used to improve the speed and quality of the integration project.

Stages of the Methodology

To define the stages of the methodology we synthesize the literature discussed in the previous chapter in the section *Development Stages*.

In the first stage we want to formalize the requirements of the project. Daclin et al. (2008) refers to this stage as the "definition of objectives and needs", Battini & Lenzerini (1984) label it as the "preintegration" phase. Ralyte et al. (2008) further divides this stage into "feasibility evaluation" and "requirements engineering". We consider the feasibility evaluation outside the scope of this methodology, we expect this to be performed before utilizing this methodology for semantic comparison.

We decide to use the definition from Battini & Lenzerini (1984) for the first stage, as this best describes the purpose of the first stage in our methodology, and we use the steps they define as part of the stage. We will now shortly describe the scope of this stage.

1. **Preintegration**
   During the first stage we need to formalize the objectives we want to achieve with the data integration. We write down a list of the real world entities that are stored in system A and that we want to exchange information about with system B. If necessary, we repeat the exercise in the other direction. The list enables us to later search the schemas of both systems for the right entities and attributes. We also select the schemas to investigate, assign priorities, and select an integration strategy.

After we have defined what information we want to share or integrate, we analyze the existing systems. This stage is best defined by Ram & Ramesh (1999) as "schema translation". During this stage, we translate each system into a common model.

2. **Schema Translation**
   In the second stage we analyze each system individually. We scan the system for the data objects subject to the integration project, as defined in the previous stage. We model these objects using a common model, so we can easily compare the systems.

Following the methodology from Ram & Ramesh (1999) the next phase is to identify interschema relationships, broken down by Haslhofer & Klas (2010) in "mapping discovery" and "mapping representation". These activities can be considered the core of our methodology.

3. **Mapping discovery**
   In the third stage we find and categorize relationships between the entities and attributes in the schemas. As will be explained later, we break down this stage in several sub stages following the semantic conflicts found in the literature study.

The relationships that follow from the mapping discovery are formalized by a graphical representation in stage four. The third and fourth stage form an iterative process. After each sub stage of the mapping discovery, the findings are added to the mapping representation. The mappings are based on the representation technique used by Haslhofer and Klas (2010).

4. **Mapping Representation**
   The fourth stage is concerned with the formal declaration of the mappings between the schemas. The model displays the semantic relationships between the schemas, and presents the conversion rules necessary to solve existing conflicts.

The structure of the methodology is represented by Figure 4-1:

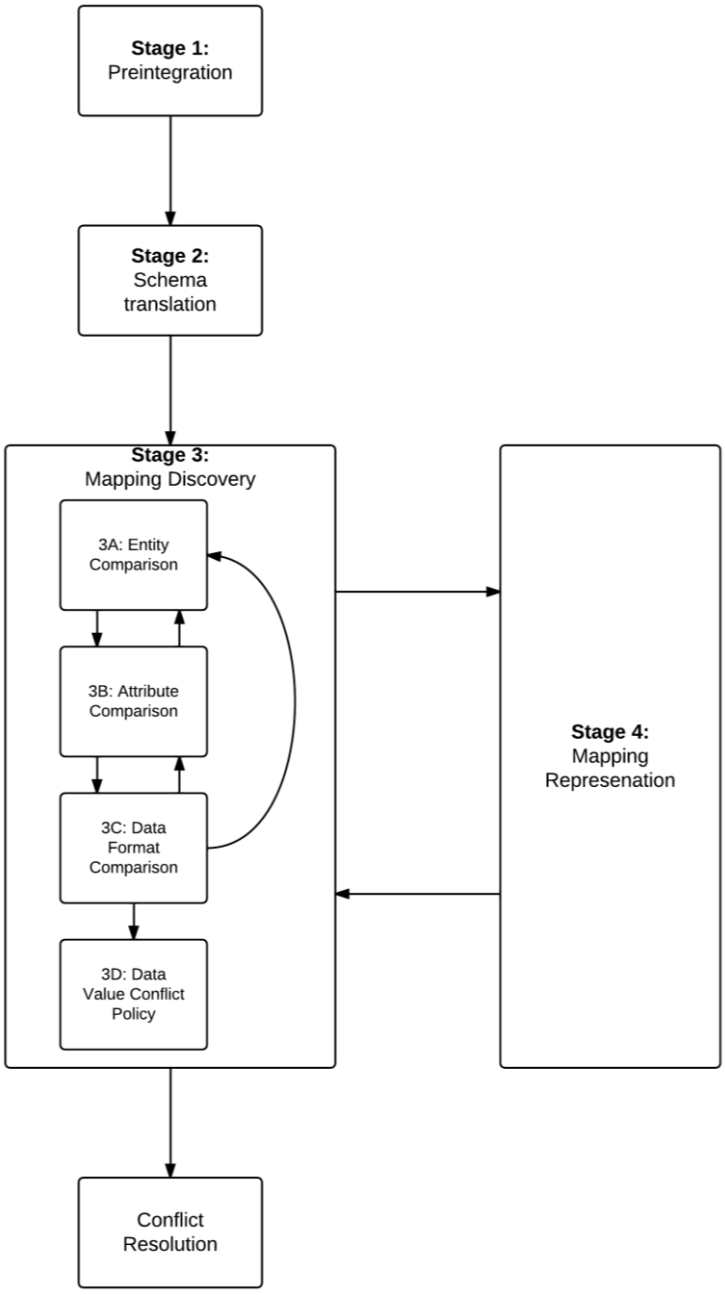


Figure 4-1: Methodology Structure

Stage 1: Preintegration

The preintegration phase is designed to define an integration policy that supports efficient and effective execution of the following stages. Without doing so, the following stages can be misunderstood, thereby creating undesirable outcomes. According to Betini and Lenzerini (1984) this stage should include the following steps:

- A selection of the schemas to be integrated
- The order of integration and a possible assignment of preference to entire schemas or portions of schemas
- Global integration strategies

Before addressing these points, we believe it to be important to clearly write down the objectives of the integration project. This formal description of the purpose of the project serves as a tool to check if all parties involved understand the goals of the project, and agree to the projected outcome of the project. It also provides a great evaluation tool at the end, to check if the end result is consistent with the original intentions.

Define objectives

The description of the objectives should start with the immediate cause of the integration project. Where does the desire to integrate the systems come from? This cause will be obvious to the initiator of the project, but might be less clear to others. By writing it down, we make sure everyone involved understands the problem we try to solve.

We further write down a description of every party involved in the project. Which benefit directly from integration? And which parties are not so much interested in the outcome of the project, but are necessary for successful execution? This helps us in selecting the schemas to be integrated, and provides understanding for differences in priorities across the involved partners. Note that in many information integration processes we can observe an unequal power balance, and projected benefits often differ strongly. As a result, one party may be reluctant to share information, thus frustrating the project. By signaling this problem up front, it is easier to act appropriate.

We placed the political process of conflicting benefits and willingness to share information outside the scope of this research. The methodology presumes that both organizations provide the necessary information to successfully integrate the systems so that the goal of the project can be achieved. Note that incomplete information will not result in failure of the methodology, but inevitably will lead to poorer quality of the total result of the project.

Select schemas to be integrated

In projects that involve medium to large size organizations, the information systems subject to study can be very large. Systems will often be segregated into multiple parts such production-, financial, and logistical information systems. There is no need to scan each and every class in the whole system, if we are only interested in for example integration of the logistic processes. This would significantly reduce the speed and efficiency of the project. Therefore we make a selection of the schemas to be integrated.

List order of integration and assign preference

In projects where we want to integrate multiple parts of the systems, such as financial- and production systems, we need to prioritize one over the other. Exercising the whole integration project at once is often a bad idea, since this would risk all processes at once. By prioritizing one part over the other, we can first integrate that part, before moving on to the next.

Choose global integration strategy

In interoperability projects, the global integration strategy involves a selection of the right interoperability approach. Three approaches have been defined by the CEN/ISO standard 11354-1: the federated-, unified-, and integrated approach. Since each of these approaches affect the project in a different way, we need to choose one of the three possibilities.

With the federated approach one system makes a direct request to the other system, and needs to convert the information it receives into its own format. In this case we need to compare the semantics of every local schema involved, and identify conflicts between each pair. It is not necessary to compare the complete systems if our goal is not to integrate every aspect of it. The objectives as defined in the first step of this stage, are therefore very important when using this approach. First of all we need to know if we want to exchange data in both directions, or that there is only one receiving- and one sending party. For every receiving party we need to know the concepts it wants to receive, which is considered as the target schema(s).

The unified approach uses a federated schema to map each individual system to. With this approach we have to connect the local system to the concepts of the federated schema. The target scheme is therefore predefined; it is the federated schema. In constructing the federated schema, choices have already been made about what parts of the systems to integrate, and what not. Our methodology will then be used to compare the semantics of the federated schema with similar concepts in the local system.

The integrated approach requires existing systems to be transformed into the common ontology that was created earlier in the interoperability project. The current system uses different design choices to store the same real world concepts, thus a discrepancy exists. Before we can convert from the current- to the new system, we have to identify semantic conflicts so we can address them properly. In this case the new ontology is the target schema. We should compare all concepts that are part of the ontology with the existing system.

Stage1: Preintegration	
Goal	Objectives setting, and to define an integration policy that supports efficient and effective execution of the following stages.
Tasks	1. Define objectives 2. Select schemas to be integrated 3. Prioritizing 4. Select an interoperability approach
Technique	Preintegration sheet (Appendix 1)
Deliverable	Document describing the objectives and integration policy (Appendix 1) Diagram of each schema involved in the project

Stage 2: Schema Translation

During the first stage we defined what concepts we want to integrate, and selected the schemas relevant to the project. This provides a good starting point, but the concepts can still be widely defined and the schemas might still consist of many classes that won't be used for the integration process. During this stage we narrow this down to the parts of the system that store the information to be exchanged.

To present the findings of this stage, we need to choose a diagramming language. In selecting the best modeling language to do so, we define three minimum requirements:

- The diagram must visually graph the classes, its attributes, and the relationship between the classes in a system
- The modeling language must be widely accepted, therefore supported by most diagram software packages.
- The language must be naturally readable, thereby easy to learn.

In our view, two languages qualify best based on these requirements: UML (Unified Modeling Language) class diagram, and ER (Entity Relationship) diagram. After experimenting with both of them, we choose to use the ER diagram mainly for two reasons. The first reason being that ER diagrams are better in visually representing the relationships between the classes/entities. The second reason is that this modeling language better fits with the integrated approach, since ontologies are often modeled in ER diagrams.

The task is now to model the relevant parts of the systems using the ER-diagram notation. In the first stage we defined what concepts the target schema wants to receive. We find these concepts in the local schema of each party, and model the entities storing that information. In many projects it's not only the entities themselves that store vital information about the concepts, but it's the relationships between them that's important. In such projects it is important that the relationship is part of the diagram, although this might mean that some entities included in the diagram will not be used directly.

Stage2: Schema translation	
Goal	Locate the relevant parts of each system.
Task	Create an ER-diagram of each information system involved in the integration project. Limit the diagram to the parts of the system that store information about the relevant concepts as defined in the previous stage.
Technique	Use the Entity Relationship diagram technique.
Deliverable	A separate ER-diagram of each system.

Stage 3: Mapping Discovery

In the previous stage we have established a mapping of the relevant parts of each local schema. By laying these schemas next to each other, we are now ready to start the process of semantic comparison.

From the literature study we found many different types of semantic conflicts. During this stage, we search for these conflicts so that we can try to solve them. Before we do so, we first make a

categorization of the conflicts we found during the literature study. We found conflicts at four different levels of comparison: the entity-, attribute-, data format-, and data value level (Table 4-1). There also is a category of problems that is seen across these four levels, referred to as 'schematic discrepancies'. This category refers to conflicts where data in one system is similar to meta-data in another system.

Note that we did not include naming conflicts in the categorization since these can be easily addressed. Remember that we are not importing foreign classes in existing systems here. If that would be the case, we would have a problem when two entities or attributes are given the same name. However, in interoperability projects we are directly targeting the instances that are stored by the other system. When referring to a foreign entity, we therefore proceed the name of the entity by the name of the database. Each entity name therefore is unique, so that naming conflicts can be ignored in this methodology.

Entity Comparison	
Generalization Conflicts	Conflict between two semantically related entities where one is a more generalized concept as the other.
Aggregation Conflicts	Conflict where one entity in a database covers the same concept as two or more entities in a different database.
Database Identifier Conflicts	Situation where two similar entities have semantically unrelated identifiers, thereby complicating a comparison of its entries.
Attribute Comparison	
Generalization Conflicts	Conflict between two semantically related attributes where one is more generalized than the other.
Aggregation Conflicts	Conflict where a single attribute in one system stores similar information as multiple attributes in the other system.
Schema Isomorphism Conflicts	Conflict where attributes of semantically related entities do not exist in each system.
Data Format Comparison	
Data Representation Conflicts	Conflict where two semantically similar attributes use different data types or notations to store data.
Data Scaling Conflicts	Conflict where two semantically similar attributes use different scales and measures to store its value.
Data Precision Conflicts	Conflict where two semantically similar attributes store their value in a different level of precision.
Default Value Conflicts	Conflict where semantically similar attributes have different default values.
Attribute Integrity Constraint Conflicts	Conflict where semantically similar attributes have different value constraints.
Data Value Comparison	
Known Inconsistency	Situation in which one system/database is known to be more reliable than the other.
Temporary Inconsistency	Situation in which differences in values between two systems are time dependent, and will eventually disappear.
Acceptable Inconsistency	Situation in which differences in values between two systems can be neglected.
Schematic Discrepancies (data in one DB refers to meta-data in other DB)	

Data Value Attribute Conflicts	Conflict where the value of an attribute is similar to an attribute itself in the other system.
Attribute Entity Conflicts	Conflict where an attribute in one system is similar to a relation in the other system.
Data Value Entity Conflicts	Conflict where the value of an attribute in one system is similar to a relation in the other system.

Table 4-1: Schematic Conflicts

The conflicts occurring at lower levels require some kind of semantic relation between their "parents". For instance, conflicts found by data value comparison can only occur between two semantically related attributes, otherwise there would be no point in comparing the values. We therefore propose a top-down approach, where we start by finding semantically related entities. We then compare their attributes, followed by the attributes' data format, and finally the data values. We also look out for schematic discrepancies during the whole mapping discovery stage. This top-down approach is supported by Haslhofer and Klas (2010).

Stage 3A: Entity Comparison

At the entity level we identified four types of conflicts. Each conflicts happens between two semantically related entities, so the first step is to map related entities. During stage 2, we searched for the concepts, as defined in stage 1, in each of the local systems and translated this into an ER-diagram. We therefore know what concept each entity describes, and should be able to map the related entities of the local systems.

For each related pair of entities, we map the abstraction level of the relationship which can have two values: a generalization or an aggregation.

Generalization

In cases where one of the entities describes the concept in a more generalized/specialized way than the related entity in the other system. So, entity E_x in system 1 is a subset of entity E_y in system 2:

$$(E_x \in system1) \subseteq (E_y \in system2)$$

Aggregation

In cases where an entity in one system is related to more than one entity in the other system. So, entity E_x in system 1 intersects with n entities E_y in system 2:

$$(E_x \in system1) \bigcap \sum_{y=1}^{n} (E_y \in system2)$$

Besides a classification as a generalization or aggregation, the relationship is defined by either a 1-to-1 relationship, or a 1-to-many relationship.

1-to-1

In a 1-to-1 relationship, instances of the related entities correspond 1-to-1. So, an instance i_j of entity E_x in system 1 is similar to an instance i_k of the related entity E_y in system 2:

$$(i_j \in E_x \in system1) = (i_k \in E_y \in system2)$$

1-to-many

One instance of the entity in one system, is similar to the information captured by more than one instance of the similar entity in the other system. So, an instance i_j of entity E_x in system 1 is equal to n instances i_k of entity E_y in system 2:

$$(i_j \in E_x \in system1) = \sum_{k=1}^{n} (i_k \in E_y \in system2)$$

An example of a 1-to-many relationship is when both systems have an entity 'Revenue', but one system stores daily revenues, while the other stores weekly revenues. In this case we have a 1-to-many relationship, since each instance of the weekly revenue entity is similar to seven instances of the daily revenue entity.

During comparison we write down a list of related entities and their abstraction. The listing is accompanied by a formal description of the relationship. If for instance a generalization is found, we should explain exactly how one entity is a generalization of the other.

We then have a list of semantically related entities. However, if we want to exchange instances of these entities, it is important that we can distinguish and then compare individual entries. It is therefore crucial that the two entities share a common instance identifier. Hence, we compare the identifiers.

Identifier Comparison

For each pair of entities previously defined, we are going to compare their identifiers. This way, we can discover identifier conflicts before happening, and we make sure that the exchanged information relates to the same real world entity in both systems.

At this stage, we are not comparing actual database entries, but are merely looking at the semantics. Two identifiers can hold four types of relationships:

Unrelated

Although the two entities describe the same real world objects, they do not share a population. For instance, let's say we integrate the systems of two schools. Each system includes the class 'Student', storing information about the students studying at that school. Since a student can only be submitted to one school, they could never share an object. In this case, the lack of a common identifier is not a problem, since the entities do not share a set of instances:

$$\{i : i \in E_x \in system1\} \neq \{i : i \in E_y \in system2\}$$

Where:

i = instance

E_x, E_y are the related entities

External domain

Since both entities derive the identifier from the same external domain, they must share a common identifier for the same real world entity. The social security number of a person is a good example of an identifier derived from an external domain.

$$(id \in E_x \in system1) = (id \in E_y \in system2)$$

Where:

id = identifier

E_x, E_y are the related entities

Functional

The identifiers are not derived from a common external domain, but there exists a relationship between the two. In this case, it is possible to write a function that converts one identifier to the other.

$$(f(id) \in E_x \in system1) = (id \in E_y \in system2)$$

Where:

f = function or mapping that relates the two identifiers

id = identifier

E_x, E_y are the related entities

Conflict

A conflict implies that there is no common identifier, so matching of instances is not possible. We can therefore not exchange instances between the entities without risking duplicate entries. Also, we cannot query a direct instance from one system to another.

$$(id \in E_x \in system1) \neq (id \in E_y \in system2)$$

Where:

id = identifier

E_x, E_y are the related entities

To make it easier to find the right classification for the two identifiers, we created a decision tree (Figure 4-2). We now explain the questions in this diagram.

- *Are the entity instances identified by one of the entity attributes?*

In some cases the identifier of an entity is stored in a different entity. In that case the entity subject to comparison holds a relational key to the instance in the entity that stores the identifier. This design choice allows for flexibility and makes it possible to use different identifiers within the same system. If this is the case, we note an attribute-integer conflict, and then swop the entities so that we can compare the actual attributes that hold the unique identifier.

- *Are the identifiers in both entities derived from the same external domain?*

When identifiers are borrowed from the same external domain, we know that every duplicate entry will have the same identifier value, thus making data matching very easy. For example, we can have two entities from two different systems, each storing information about certain individuals, and both use the person's tax ID as their unique identifier. In this case, the identifier is borrowed from the same external domain and thus can be matched instantly. Note that it must be the SAME external

domain, i.e. if one database stores the German tax ID while the other uses a UK tax ID, obviously we cannot match the data.

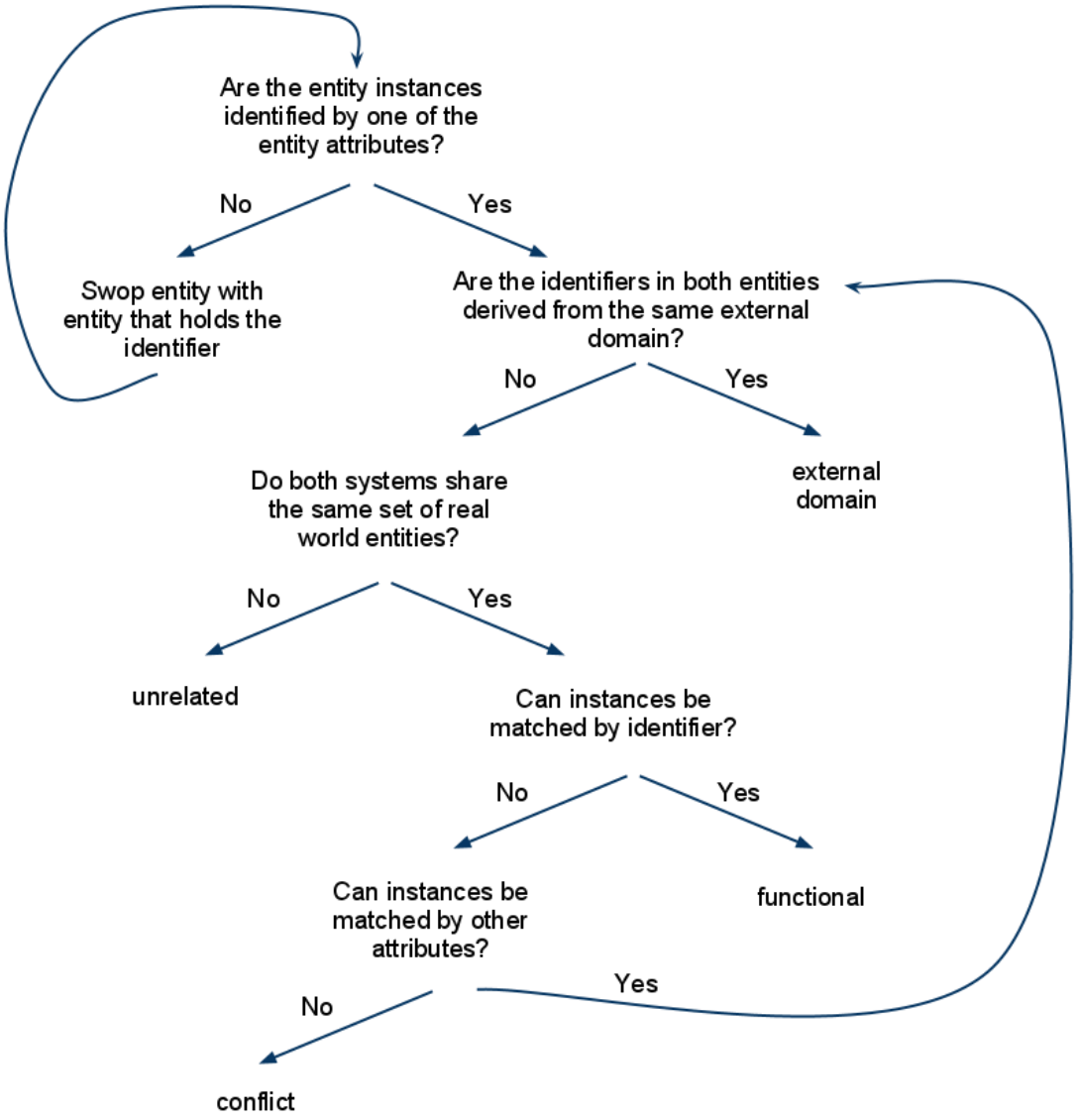


Figure 4-2: Identifier Comparison

- *Do both systems share the same set of real world entities?*

When the identifiers are not borrowed from the same external domain, it is the question whether there is a chance of duplicate entries when comparing both entities. If the databases share no same real world instances, existing data cannot be conflicting with each other, making the integration project easier.

- *Can instances be matched by identifier?*

In case the databases do share similar real world entities, we want to know if we can use some function or expert knowledge to convert the identifier from system 1 into that of system 2. For instance, system 1 may hold a person's tax ID as a unique identifier, while system 2 uses the persons

passport number. With the help of some remote database that stores both the tax ID and passport number of each person, we could easily convert system 1 its identifier into the identifier from system 2.

- *Can instances be matched by other attributes?*

When identifier matching is not possible, we have to see if there are other entity attributes that could help us find duplicate entries. It is noted that this can be a comprehensive study in itself, known in literature as 'candidate keys'. However, in some projects it might be obvious and we can take the new pair of identifiers and follow the decision tree from the start again.

After the relationship between two identifiers is found, we include this in the list of related entities previously created. We have then identified all three semantic problems categorized under the entity level.

Stage 3A: Entity Comparison	
Goal	Identify semantic conflicts at the entity level.
Task	Find related entities and map the abstraction of the relationship. Find and compare the identifiers of related entities.
Technique	To find related entities we use the deliverables from the stages one and two. To compare the identifiers we use the decision tree as presented in this stage.
Deliverable	A list of related entities and their identifiers.

Stage 3B: Attribute Comparison

At the attribute level we make a comparison between attributes of related entities. The goal here is to check whether the abstraction of the relationship defined in stage 3A remains valid, and to identify semantic conflicts at attribute level.

This stage is limited to finding the right relationships between the attributes of both entities. The way the data is stored in each attribute is irrelevant here, this will be the focus of the next stage. For example, if we compare two entities each with an attribute describing 'temperature', this stage will result in the creation of a direct relationship between the two attributes. The fact that one attribute measures its value in Kelvin and the other in Celsius is not of concern at the attribute level.

Table 4-X lists three categories of semantic conflicts at the attribute comparison level: generalization-, abstraction- and schema isomorphism conflicts. The table also includes attribute-entity conflicts, filed under schematic discrepancies, which can also be identified during attribute comparison.

Besides these four, we propose an extra outcome of the attribute comparison process. Not so much of a conflict but nonetheless a type of relationship between two attributes, is a functional dependency:

> *Functional Dependency:* When the value of an attribute in one system can be computed from the value of an attribute in the other system, but the two are not similar.

Each pair as listed in the deliverable from stage 3A is now to be investigated. For each attribute (except identifiers) of the entities in the list, we use a decision tree for classification (Figure 4-3).

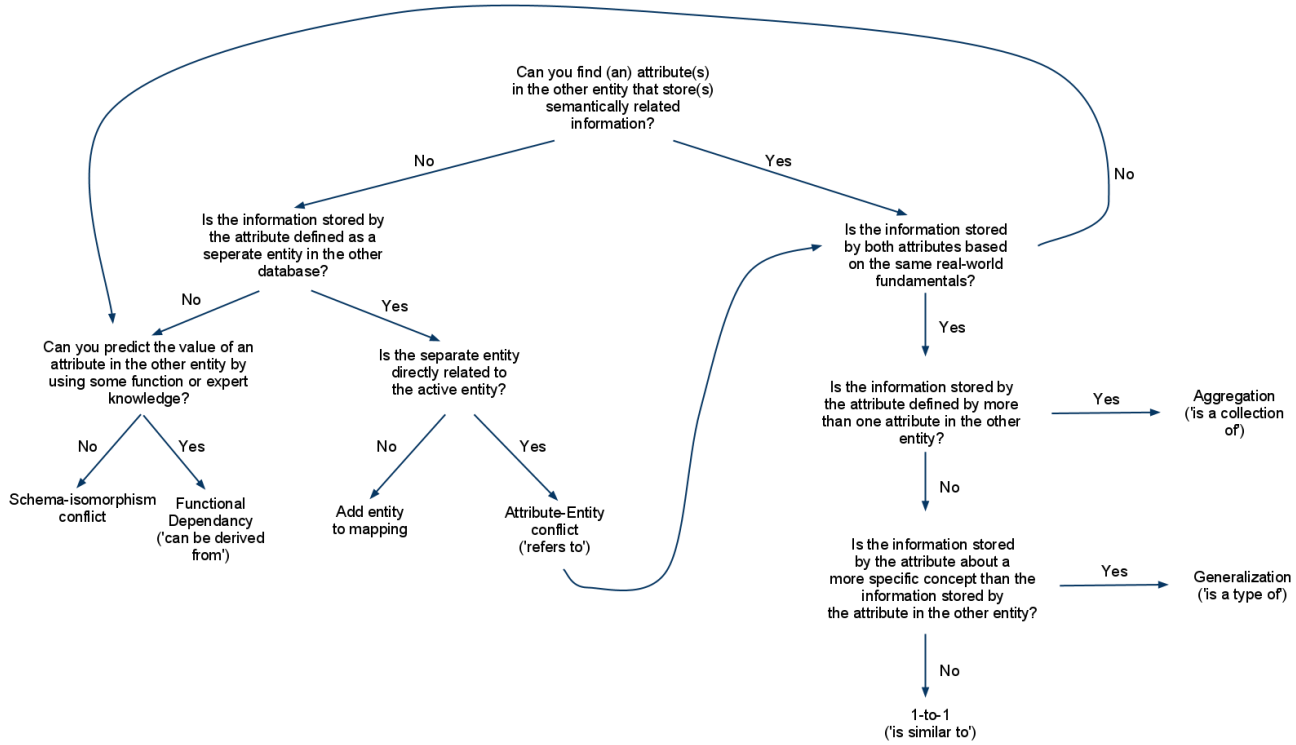


Figure 4-3: Attribute Comparison

We shortly describe each question from the decision tree:

- *Can you find (an) attribute(s) in the other class that store(s) semantically related information?*

We are looking for a similar attribute in the other entity so that we can integrate the information stored in both entities.

- *Is the information stored by the attribute defined as a separate entity in the other database?*

In some cases we might not be able to find a semantically similar attribute because the information stored by this attribute is defined as an entity in the other database. For example, one system includes the class 'Sales' with attributes *customer name* and *customer address*. The other system also has an entity named 'Sales' but this one does not have attributes for the customer name and address. The system has defined the customer as an entity, and thus stores this information in the entity 'Customer'.

- *Is the separate entity directly related to the active entity?*

If the separate entity is directly related to the entity that was subject to the comparison, we note the attribute-entity conflict, and continue the decision tree with the different entity and its appropriate attribute. If the entity is not directly related, we make the entity part of the entity mapping, meaning that we go back to stage 3A.

- *Can you predict the values of an attribute in the other class by using some function or expert knowledge?*

Although two attributes do not have a semantic relationship, it still might be possible to retrieve the value for one by using the data from the other. As an example we can think of two entities storing information about compact music players for a certain brand. One entity has an attribute storing the players' color, the other has an attribute storing the serial number of the device. Although we cannot find a similar attribute for 'color', by utilizing our expert knowledge we know that every serial number ending with a *B* is colored black.

- *Is the information stored by both attributes based on the same real-world fundamentals?*

Two attributes can be semantically similar, even though they are not based on the same real-world fundamentals. For instance, two financial databases can store an attribute named P/E (price-to-earnings) ratio. Although these attributes are semantically similar (they both describe the price-to-earnings ratio of a certain company) they can be based on different real-world fundamentals. For instance, for earnings we could take earnings over the last 12 months, over the last calendar year, the last fiscal year, or the last three historical quarters and the estimated current quarter (Madnick and Zhu, 2006).

- *Is the information stored by the attribute defined by more than one attribute in the other entity?*

If this is the case, the attribute forms an aggregation of several other attributes in the other class. For example, one entity might have the attribute *Name* while the other entity has attributes *First Name* and *Surname*.

- *Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other entity?*

To elaborate on the previous example, this would be the case when both systems have an attribute *Name* but one only stores the surname, while the other stores the full name (both first- and surname).

Each attribute classification is added to the mappings created in stage 3A. We write down every result from the decision tree as the following:

Mx: [{system.entity.attribute ; system.entity.attribute}, result]

The list is accompanied by a description of the relationship that clarifies the classification. The description provides the end user with more insight in the relationship. For instance, in case of a generalization we would have to describe what specific part of the concept captured by one attribute is captured in the other, and for a functional dependency we would have to describe how one value can be calculated from the other.

Stage 3B: Attribute Comparison	
Goal	Identify semantic conflicts at the attribute level
Task	Find related pairs of entity attributes and classify the relationship.
Technique	Use the decision tree to find the right classification for each pair.
Deliverable	A list of relationships between attributes.

The attribute comparison could provide us with new insights into the entity relationship. Therefore, it might be necessary to update the relationship list from stage 3A. If the result of the decision tree is 'add entity to mapping', we have to update the mapping to include the entity and possibly change the classification.

Stage 3C: Data Format Comparison

Now that we know which attributes are related, and are therefore expected to exchange information, we compare their data formats. In our classification of semantic problems we listed five types of conflicts under the data format level. We now describe the problems involved with each of these.

Data representation

Data values can be stored in either a numeric format, or in a string format (includes date type and Booleans). When the related attributes use a different format, we might have problems exchanging information. For instance, a database field with a numeric format will not accept a string.

Another form of a data representation conflict is a difference in string formats. For example, compare the string '*January 1, 2010*' with the string '*2010-01-01*'.

Data precision

When communicating values from one system to another, we can have a difference in the level of precision. Imagine two systems communicating student grades. Both systems use a 1 to 10 scale for the students' grade, but one is saved in 1 decimal precision, the other one uses integers only. This conflict type addresses the problem when communicating a grade of *7.4* to the other system.

Data unit

Values that represent measurements are always measured by a certain type of units. For example, attributes representing size can be measure in meters, centimeters, feet, and miles (to name a few).

Default value

The default value can be either a set value, or be empty. Correctly identifying a different choice can be important. Imagine system A having a default value of '0' for the attribute 'inventory', while system B has an empty value as default. When the value of '0' is copied from system A to system B, and we request inventory for this product in system B, it will tell us that there are currently no products in stock. This might be false, since the inventory level has never been counted and therefore carries the default value.

We also experience problems in the case of data representation conflicts, where it may be difficult to directly compare defaults. For instance, think of a situation in which we have to compare an attribute with Boolean value (default: *false*) with an attribute with integer value (default *empty* or *0*).

Integrity constraints

Every attribute includes some restriction(s) that bounds the storage of information. String format data types are often restricted by a maximum number of characters, numeric data types often have a minimum and maximum value. The restrictions can be either given by design choices for the information system, or by design choices for the objects stored by the class. Let's say we have a class *Children* and a class *Person*. The class *Children* will have its attribute *Age* bounded by a maximum

value of 12, while the same attribute for the class *Person* can range from anywhere between 0 and 150. In this example the restrictions are given by the objects stored in the system.

In the same example we also have an attribute *Name*. In one system it can have a maximum length of 50 characters, the other system allows 75 characters. In this case the restrictions are a result of different design choices for the system.

To identify each of the conflicts listed above, we created a table (Table 4-2) that the methodology user is required to fill in for each attribute mapping. Goal is to easily check the differences between the data formats of both attributes, by comparing the two columns in the table.

Mapping	(system1) Attribute name	(system2) Attribute name	Conflict
Format			
Characters			
Notation			
Range			
Default value			
Scale			

Table 4-2: Data Format Comparison

The table exists of six characteristics to describe each data format:

- Format: A format can either be numeric, or a string. If the attribute only accepts numbers as valid values, the type is numeric. If the attribute accepts alphanumeric values, the type is string. Differences in format indicates a *data representation* conflict between the attributes.
- Characters: The number of characters that is allowed. A difference between the number of characters for each attribute could indicate a *data precision-*, a *data representation* conflict, and/or an *attribute integrity constraint* conflict.
- Notation: Describes the structure of the notation. For instance, a date can have many different notations such as 01012011, 01-01-2011, or January 1, 2011. Differences in the notation that two attributes use indicates a *data representation* conflict.
- Range: The range indicates the allowed values for the attributes. Differences in the range of two attributes indicates an a*ttribute Integrity constraint conflicts.*
- Default value: Obviously, this describes the default value of the attribute. Differences in default values indicates a *default values* conflict.
- Scale: In case the attribute describes a measurement, this describes the scale that was being used. Differences in the scale used by the two attributes indicates a *data unit* conflict.

The result of the data format comparison is a list of all data format conflicts between related attributes. The list is added to the deliverables from the previous two stages.

Stage 3C: Data Format Comparison	
Goal	Identify semantic conflicts at the data format level
Task	Fill in the comparison table for each attribute pair, and compare the values of the two columns.
Technique	Use the table provided.
Deliverable	A list of semantic conflicts between the data formats of related attributes.

Stage 3D: Data Value Comparison

In the data value comparison stage we identify the entities where data value conflicts can occur, so we can later set a policy for how to handle these conflicts. Value conflicts can only occur between entities that share the same set of instances, so we only need to investigate these pairs. For each pair, we look at three possible value inconsistencies that can take place between their instances.

Known inconsistencies

Are the data values for the entities known to be more reliable in one system than the other? In some projects we might know that the values in one system are older, and therefore may be outdated. In another project we might know that the values in one system are recorded with better measurement instruments than in the other, therefore to be more reliable.

Temporary inconsistencies

In projects where the information stored in databases is time dependent, we might have conflicting values because the change in information in one database has not yet propagated to the other databases. Imagine two systems having databases storing the current cost price of a product. One of the systems updates the value every Monday, the other on Fridays. If values are exchanged between these systems on a Wednesday, we might find a conflicting value.

Acceptable inconsistencies

Certain differences in data values might be regarded as acceptable, and therefore do not have to be corrected. We can think of two logistical systems storing the distance between cities. If for example, we compare the data values of the recorded distance between Amsterdam and New York, we can accept conflicts within a certain range.

As a deliverable for this stage, we create a list of each pair of entities sharing the same set of instances, and formalize their known inconsistencies, the risk of temporary inconsistencies, and (if applicable) the acceptable range in value inconsistencies.

Stage 3D: Data Value Comparison	
Goal	Identify the type of data value conflicts that might occur between related entities.
Task	Identify related entities that share the same set of instances. Research their known-, temporary-, and acceptable data value inconsistencies.
Deliverable	Document describing the known-, temporary-, and acceptable data value inconsistencies between related entities.

Stage 4: Mapping Representation

The diagrams we create in stage 4 should provide the programmer with an overview of the relationships between the systems. This should improve the efficiency of the programming, and eliminate data problems resulting from semantic conflicts. The diagrams should be created as an addition to the lists created during stage three, never as a replacement. The documents resulting from stage three provide detailed insights into the relationships, while the diagrams only display the relationship. For example, while the diagrams only show a functional dependency between two attributes, the documents from stage three explain the exact relationship between two values of the same object.

As explained earlier in chapter four, stages three and four are executed in a parallel process. After each sub-stage of stage three, the findings are documented with the representation techniques that we will now explain.

The mappings are based on the representation technique used by Haslhofer and Klas (2010). First, a diagram is created with two swim lanes, each describing one system. Each swim lane consist of the entities used in the integration project. We then draw relationships between the entities of the systems as defined in stage 3A. Between the relationship, we draw a mapping element M with a unique number for that relationship. Besides the unique number, we write down the classification of the relationship. An example of the representation technique is illustrated in Figure 4-4.

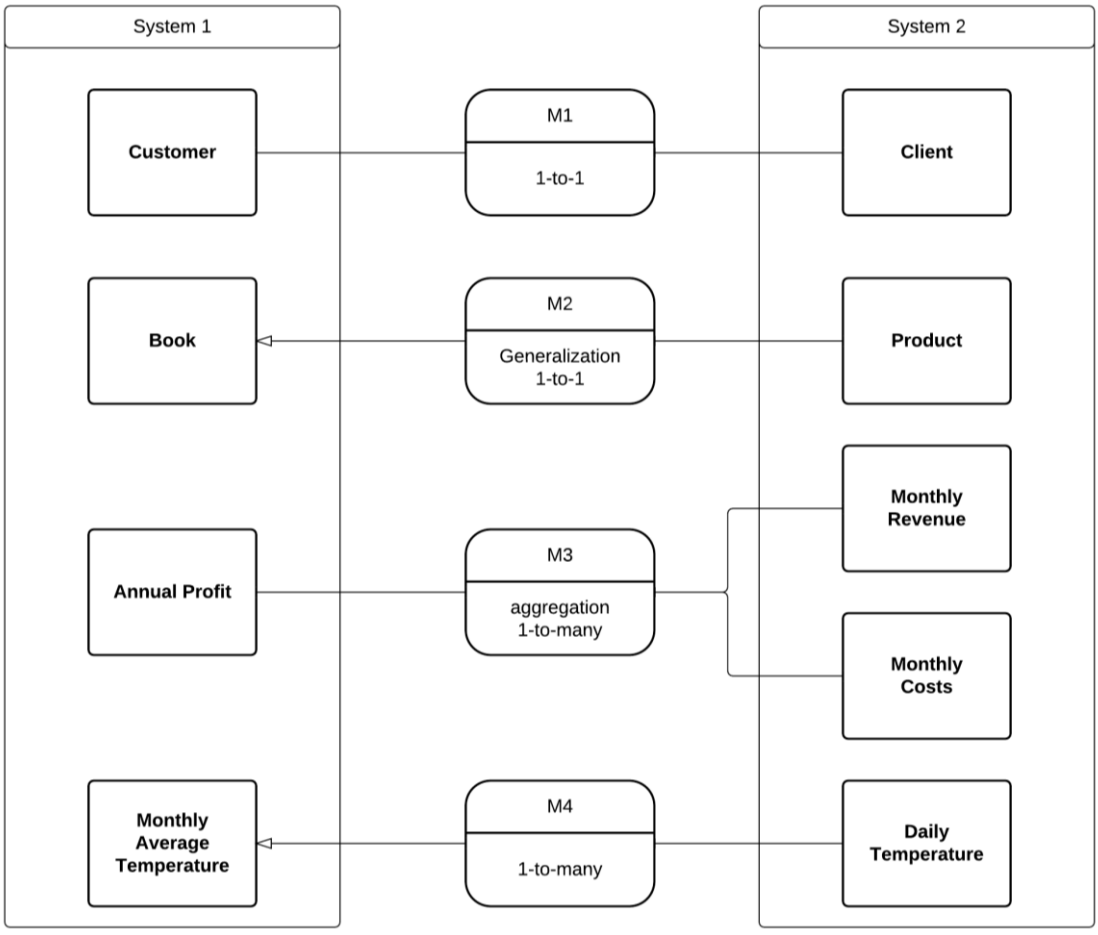


Figure 4-4: Entity Comparison Notation Example

Next, we draw a separate diagram for each mapping relationship. So in the example above, we would create four different diagrams (M1 – M4). Each diagram consists of the related entities and their identifiers and attributes.

A line between each pair of attributes is drawn to indicate the classification of the relationship. Attributes that do not have a counterpart in the other entity are modeled in the 'schema-isomorphism' section at the bottom of the diagram.

Figure 4-5 illustrates what the mapping should look like:

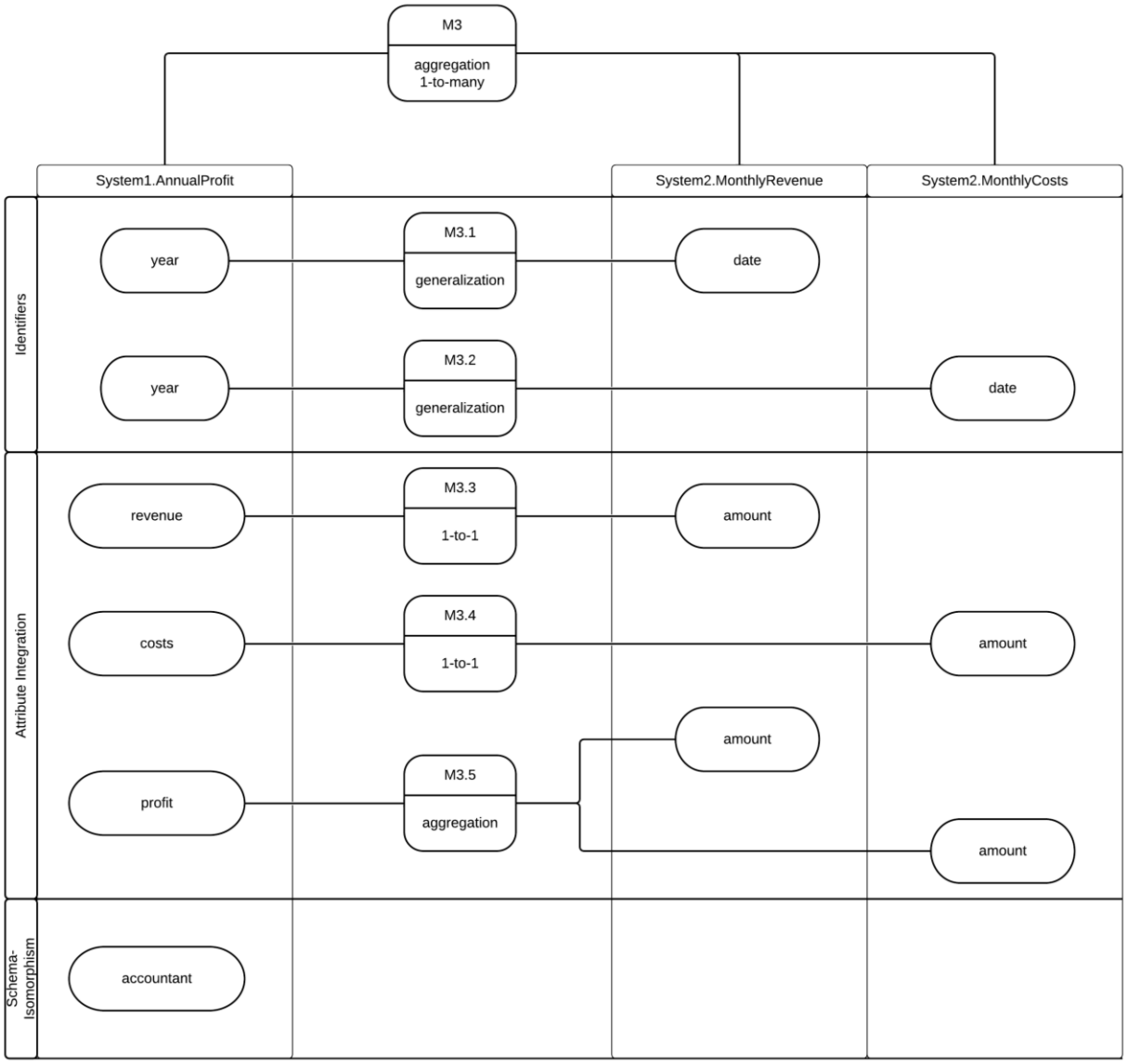


Figure 4-5: Attribute Comparison Notation Example

After all semantic conflicts have been identified and documented, the problems have to be resolved. Although solutions to these problems are often very context dependent, in the following section we provide several guidelines on how to address them.

Stage 4: Mapping Representation	
Goal	To create an overview of all relationships and conflict types found with the methodology.
Task	Create a diagram with an overview of the related entities. For each pair of related entities, create a diagram of the attribute mapping.
Technique	Use the diagramming technique proposed in this stage.
Deliverable	A diagram of each mapping.

Conflict Resolution

The classifications made for related entities and attributes in the third stage of the methodology help us in solving the semantic conflicts. In this section we provide guidelines about how to solve each conflict, based on its classification. Since each conflict is unique as it depends on its context, we can only provide general guidelines that assist the user in solving the problem. Custom functions must be written however, so that queries from one system are translated to the semantics of the other system.

Entity conflicts

We first address the classifications at the entity level.

Aggregation

An aggregation relationship indicates that a single entity in system 1, has to extract/send data from/to several entities in system 2.

The conflict can be solved by creating a virtual entity in system 2 that combines all of the attributes it needs to communicate with system 1 (Figure 4-6). This way, the virtual entity can reroute all queries in both directions. The great benefit of using such a virtual entity is that when system 2 makes any changes to the underlying entities, it can update the virtual entity themselves, without system 1 having to rewrite its queries.

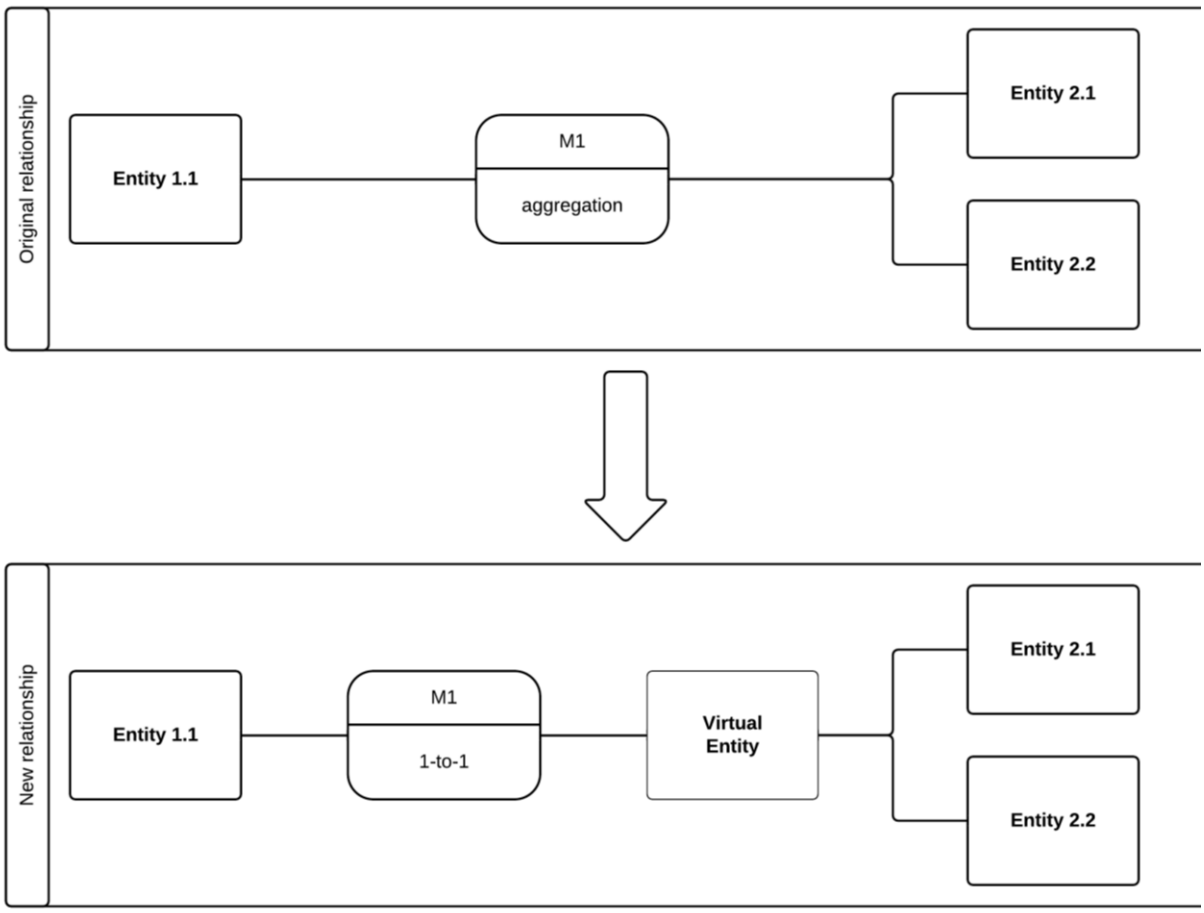


Figure 4-6: Virtual Entity for aggregation conflict

Generalization

A generalization at the entity level indicates that one entity stores instances that are a subset of the instances stored by the other.

$$\{i : i \in E \in system1\} \subseteq \{i : i \in E \in system2\}$$

Where:

E = entity
i = instance

This will only create problems in one direction. When the superset queries the subset, there is no problem as it can target every instance. The other way around is more problematic. The subset can only query a targeted set of instances in the superset. For instance, if we have a generalization between the entity 'Book' in system 1, and the entity 'Product' in system 2, then a query from 2 to 1 must invoke that only instances about books are returned.

The solution here is to write a function that filters the set of returned instances (Figure 4-7). The challenge therefore is to find an attribute that makes it possible to identify the right set of instances.

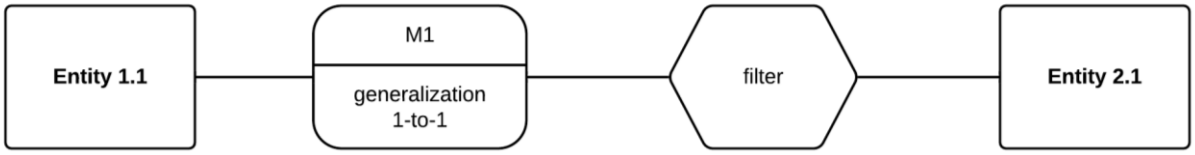


Figure 4-7: Filter for generalization conflict

1-to-many

As the relationship '1-to-many' implies, queries from the system indicating '1' must always query several instances from the system indicating 'many' at the same time. A function must be written that implies which, and how many, instances must be targeted at the 'many'-side.

Unfortunately, communication in the other direction, from 'many' to '1', is problematic. It is impossible to decompose the grouped instance into the individual instances in the other system.

1-to-1

A 1-to-1 relationship does not need any configuration.

Identifier conflicts

Next, we look at the conflict resolution procedures for identifiers. Each of the classifications is discussed.

Unrelated

If two identifiers are classified as 'unrelated', there are no shared instances between the systems, so they can freely exchange instances without the need for a common identifier. Therefore, no mapping between the identifiers needs to be made.

External domain

The relationship 'external domain' between two identifiers indicates that shared instances must already have the same identifier in their respective systems. No configuration is therefore required.

Functional

A 'functional' relationship between two identifiers implies a conversion rule. When exchanging instances between the two entities, the identifier must be translated to the typology used in the other system. So we look for a function *f* such that identifier id_j from entity E_y is equal to identifier id_k from entity E_z:

$$f\left(id_j\right) : id_j \in E_y = id_k \in E_z$$

Conflict

If the outcome of an identifier comparison is 'conflict', we have two options. We can either accept duplicate entries, or we do not integrate the entities at all. Choosing the latter implies that we alter the relationship between the two entities as created in stage 3A.

Attribute conflicts

After solving the problems at the entity level, we start working on the attribute level.

Generalization

When two attributes share a generalization relationship we know that the information captured by one of the attributes is only a subset of the information stored by the other. Therefore, we must write a function that takes into account the difference in scope of the information. So, if we have an attribute a_j in entity E_x that generalizes attribute a_k in entity E_y, we have to write a function f such that the scope of attribute a_j equals that of attribute a_k:

$$f\left(a_j\right) : a_j \in E_x = a_k \in E_y$$

Aggregation

In the case of an aggregation at the attribute level, we have to write a function that combines the values from several attributes in one system, to the single attribute in the other system:

$$f\left(a_1 ... a_n\right) : a_j \in E_x = \sum_{k=1}^{n} a_k \in E_y$$

Where:

f = the function
n = number of attributes in the aggregation
a = attribute
E_x, E_y are the entities in the relationship

Schema-isomorphism

A schema-isomorphism indicates the attribute does not exist in the other system, so we cannot do much about it. It might be wise to think about the default value to give to the attribute when receiving instances from the other system. A special default value indicates why no value for the attribute exists.

Attribute-Entity

An attribute-entity classification implies that the related entity in system 2 stores a relational key to the entity that holds the attribute that is similar to the one in system 1. When writing queries, we can use the relational key to target the right instance in the extra entity.

Functional

In the case of a functional relationship between two attributes, we have to define the conversion rule between two values. This can either be a conversion table, or a special written function, similar to the one listed under generalization.

1-to-1

A 1-to-1 relationship implies that we directly share values between the two attributes.

Data format conflicts

Data representation conflict

Data representation conflicts can be either a difference in data type, or in representation format. With data type differences we coerce the string to be an integer, or the other way around. With representation differences we write a function that converts the value to the proper representation.

Data precision conflict

To overcome data precision conflicts we have to create a conversion table that maps the lower and upper bound of the more precise to the less precise value. For example, consider one database who stores a student's grade as {A+,A,B,C,D,E,F} and the other with {1 – 100}. We then create the following table:

Lower bound	Upper bound	Grade
0	20	F
21	40	D
41	60	C
61	80	B
81	99	A
100	100	A+

Alternatively, for certain measurement values we can just round the more precise number to the less precise one: round(8.6) = 9. Care must be taken though, since in some context a difference between 8.6 and 9 can be catastrophic.

Data unit conflict

When the attributes of comparison use different units to store values, we need to create a conversion expression that maps one value to the other. For instance, one database uses Euro values to store revenue, the other uses Dollar values. We then define a constant *c* to multiply or divide the value with from one database to the other. In this example the constant might be derived from an external domain so that it uses a real-time conversion rate.

Default values conflict

When experiencing different default values for semantically alike attributes we need to convert the value to the default of the local system. However, this value change must then not be propagated to the system with the original data.

Integrity constraints conflict

A difference in data value constraints could mean that the two attributes are no longer related. If the entity 'Person' has a range value for the attribute 'age' of 18-30, and a semantically related entity has a range value of 40-65, the two entities are no longer related so we need to alter that in the deliverable from the entity comparison stage. If there are shared values in the range of both attributes, we have to decide what to do with the instances outside of the overlapped values. These instances can either be filtered out in the query, or we can change the value to be inside the range of the local system.

Data values

After all the semantic conflicts have been resolved, the data integration can take place. In the case where the systems might share duplicate entries, we have to decide what to do when the values for similar attributes on shared instances conflict with each other. We create a policy to handle these problems. Note that we cannot identify the conflicts before they happen, but the policy allows us to implement rules in the system about which system has the authority to overwrite the other. The policy has to cover three categories: known-, temporary-, and acceptable inconsistencies.

Known inconsistencies

In this case we choose one system to be more reliable than the other. When conflicts occur, the more reliable system may therefore overwrite the data in the other.

Temporary inconsistencies

Since conflicts in this category are temporary, we have the chance to neglect them until they are in sync again. However, we might want to define the range of value differences that are acceptable during that period. We also need to define the maximum period the values may be different.

Acceptable inconsistencies

For each attribute pair, we can define acceptable inconsistencies. The wider the range we choose to be acceptable, the less conflicts, but also less accurate data we get.

Key Findings

The methodology we developed consists of four stages. In the first stage the problem holder formalizes the objectives of the interoperability project and defines the concepts to be exchanged. In the second step these concepts are isolated in each participating information system and expressed in an Entity Relationship diagram. In the third step the concepts in the different systems are compared at four different levels: the entity-, attribute-, data format-, and data value level. At each level we indicate the potential semantic conflicts and provide tools to identify them. In the fourth stage the user creates a visual overview of all discovered conflicts. Finally, we propose conflict resolution techniques for each conflict identified by the methodology.

5. Case Study

In the following we are going to demonstrate the methodology as performed for a case study at an integration project between the information systems of the *Dienst Uitvoering Onderwijs* and the *SUWI Gegevensregister* in the Netherlands.

Preintegration

The *SUWI Gegegevensregister (SGR)* is the standard used for data exchange between the parties within the SUWI domain. The SUWI domain is a partnership between various public services in the Netherlands, with the main goal to reintegrate unemployed people faster into the labour market.

To make it easier to find the right new job for unemployed people, the system stores information about the education history of each person. This information can be used to compare the person's skills with vacancies to get a better fit. However, not every person is honest about his or her education history. Some people explicitly don't mention the study they've followed as they would rather work in a different field of work. Since the parties involved in the SUWI domain are only concerned with getting people back in the labor market as quickly as possible, they find the missing information unsatisfying and are looking for ways to retrieve this data.

The education history of each person that followed a study at one of the Dutch educational institutions is stored in the information system of the *Dienst Uitvoering Onderwijs (DUO)*. This public organization works for the Dutch ministry of education, culture, and science. The SGR would now like to integrate this information into their own system, so that they always have a complete picture of the person's education history.

This integration project can be solved by using the federated approach. This project is characterized by a one way data exchange. There is no need to send data from SGR to DUO, and there are no system processes that have to be integrated. So in this case we only need to define the entities that SGR wants to receive information about, search for similar entities in the DUO system, and then identify and solve semantic conflicts between the two systems.

This integration project is characterized by two real world entities (Figure 5-1): (1) human beings, and (2) education programs. The point of interest is the direct relationship between the two, being the participation of a human being in an education program. So in the next stage we will have to search each system for these two entities, and the relationship in between.

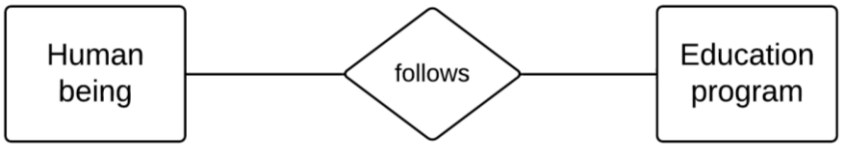


Figure 5-1: Integration concepts

Before we look exactly at how these concepts are defined in the two systems, we first search the SGR system for the part of the system that captures information about these concepts. In the systems' documentation we find one schema that models a human being (Appendix A.2) and a schema that models involvement in education program (Appendix A.3). For the DUO system we find the schema

that involves both of these concepts (Appendix A.4). Prioritizing one schema over the other is not of concern in this project, since the scope of the project is really small.

The findings from this stage are formalized in the deliverable in Appendix A.1.

Schema Translation

In the SGR system a human being is defined as the entity *'Persoon'* (Person). Each instance of this entity is identified by a *'Burgerservicenummer'* (social security number) and has 17 other attributes.

Existing education programs are defined by the entity *'Opleidingsnaam gecodeerd'* (Education name coded), which is based on a standard table that defines a code for each education program. Since not every partner connected with the SGR system support this code standard, there is an alternative entity labeled *'Opleidingsnaam ongecodeerd'* (Education name uncoded).

Next, we look for the linkage between the entities person and education name (un)coded. Researching the database class diagram, we find that the relationship between the two is captured by the entity labeled *'Opleiding'* (education), which consists of nine attributes describing the relationship. We then find the shortest path linking the different entities, so that we can construct the entity relationship diagram (Figure 5-2). Note that every instance of *'Opleiding'* is related to either the coded or the uncoded entity, so this is an OR-relationship.

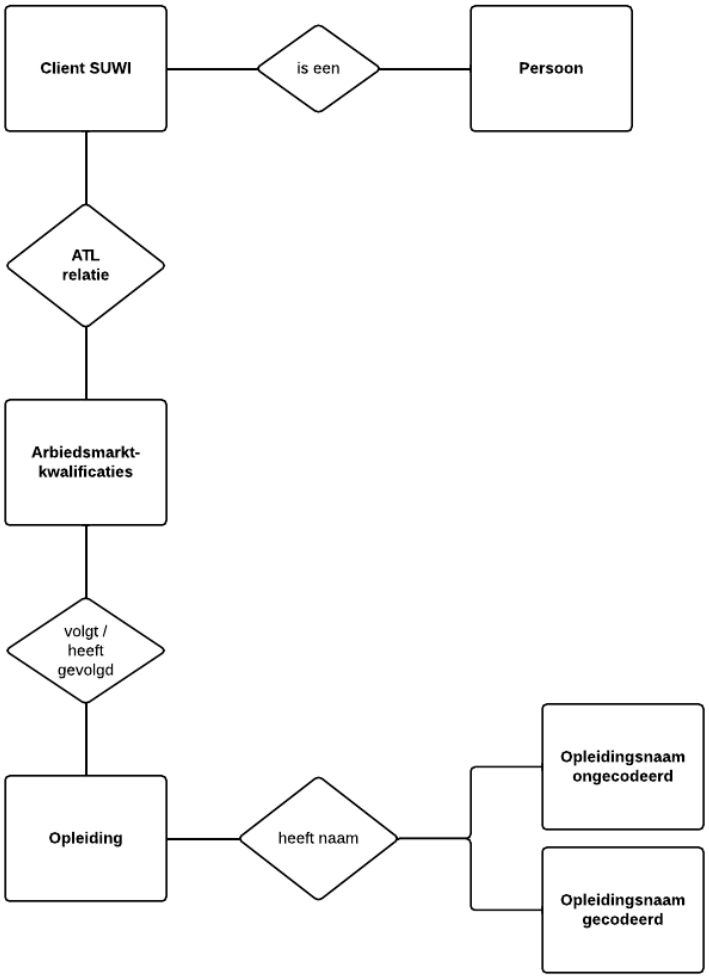


Figure 5-2: SGR ER Diagram (full diagram in Appendix B.2)

Note the following relationships in the diagram:

- *Client SUWI* is a generalization of *'Persoon'* (Person), thus each *Client SUWI* is a *Persoon*. The relationship is unique in both ways; each *Persoon* can be only one *Client SUWI*, and each *Client SUWI* can be only one *Persoon*.
- A Client SUWI can have multiple *'ATL Relatie'* (ATL Relations), but each *ATL Relatie* is linked to only one *Client SUWI*. Each *ATL Relatie* has a start- and enddate, making them unique for the specific Client.
- *'Arbeidsmarktkwalificaties* ' (Labor Market Qualification) has a 1-to-1 relationship with *ATL Relatie*.
- *Arbeidsmarktkwalificaties* can have multiple connections with *'Opleiding'* (Education), but each instance of *Opleiding* is linked to only one *Arbeidsmarktkwalificaties*.
- *Opleiding* is linked to either *'Opleidingsnaam ongecodeerd'* (Education Name Uncoded) OR '*Opleidingsnaam gecodeerd'* (Eduction Name Coded). The Coded class is the preferred relationship, the Uncoded class is providing backwards compatibility for systems that have not adopted the Coded class yet.

Next we look at the DUO system. In this system a human being is defined as the entity *'Mens'* (Human). Instances of this entity only consist of relationships to other entities, it does not have any attributes by itself.

Existing education programs are captured by the entity *'Opleiding'* (Education). There is no direct relationship between *Opleiding* and *Mens*. However, from the class diagram, we find a class that captures the participation of a person in education programs: *'Onderwijsdeelname'* (Education Participation). We then find the shortest path across these three entities, and construct the ER diagram (Figure 5-3). This path also contains the entities *'Opleidingaanbod'* (Education offerings), *'OpleidingPerInstelling'* (Education per Institution), and *'Opleiding'* (Education).

Note the following relationships in the diagram:

- An instance of *Onderwijsdeelname* can only be related to only one *Mens*, but one *Mens* can be related to multiple *Onderwijsdeelname*.
- An instance of *Onderwijsdeelname* can be linked to only one instance of *Opleidingaanbod*, but many relationships are possible in the other direction.
- *Onderwijsdeelname* holds relational keys to both *Mens* and *Opleidingaanbod*.
- *Opleidingsaanbod* and *OpleidingPerInstelling* have an aggregation relationship, where each instance of *Opleidingaanbod* is related to one (and only one) instance of *OpleidingPerInstelling*.

Figure 5-3: DUO ER Diagram

- *Opleidingaanbod* holds a relation key to *OpleidingPerInstelling*.
- An instance of *OpleidingPerInstelling* can be linked to only one instance of *Opleiding*, but many relationships are possible in the other direction.
- *OpleidingPerInstelling* contains a relational key to *Opleiding*.

Mapping Discovery & Representation

We now compare the two ER diagrams from the previous stage, starting with the entity comparison.

Entity Comparison

In the first stage of the methodology we defined three concepts of concern: human being, education program, and the relationship between. In the second stage we found these three concepts in both the systems, and constructed an ER diagram from that. Finding relationships between the entities of the two systems is thus pretty straightforward.

We defined both the entities SGR.Persoon and DUO.Mens to describe a human being, and can therefore list a relationship between the two entities. Since we do not find a generalization or aggregation relation between the two, and one instance of SGR.Persoon is similar to exactly one instance of DUO.Mens, we define the relationship to be 1-to-1:

M1: Entities[{SGR.Persoon ; DUO.Mens}, 1-to-1]

The concept 'education program' was found as two entities in the SGR system, being SGR.OpleidingsnaamOngecodeerd and SGR.OpleidingsnaamGecodeerd. As explained, the first one (the uncoded version) is only used for partners that do not support the coded version. In the DUO system we found just one entity to describe the concept, being DUO.Opleiding. Since every instance of DUO.Opleiding can be related to an instance of the coded version (SGR.OpleidingsnaamGecodeerd), we draw a relationship between the two:

M2: Entities[{SGR.OpleidingsnaamGecodeerd ; DUO.Opleiding}, 1-to-1]

The relationship between the two entities, being the participation of a human in an education program, was captured in both systems by one entity. Also, each instance in the DUO system is similar to exactly one instance in the SGR system:

M3: Entities[{SGR.Opleiding ; DUO.Onderwijsdeelname}, 1-to-1]

Now that we have a list of related entities, we compare their identifiers to see if we can exchange instances. For each mapping, we utilized the decision tree to compare the identifier semantics.

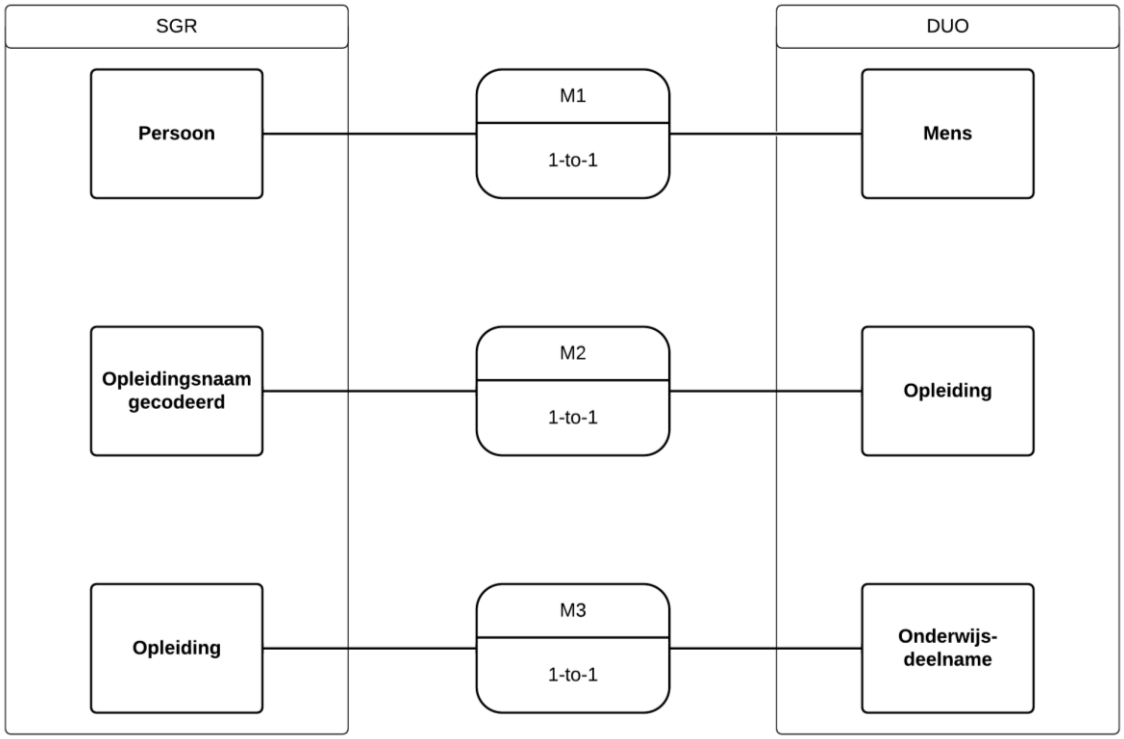


Figure 5-4: Entity Comparison diagram SGR-DUO

For M1 we found (Appendix C) that entity DUO.Mens does not have an identifier of itself, but that this is captured in another entity DUO.identificatieMens. Both SGR.Persoon and DUO.IdentificatieMens have an attribute 'burgerservicenummer' (social security number) that belongs to an external domain. Thus, we list the relationship between the identifiers of M1 as an attribute-entity conflict, and relate the identifiers of SGR.Persoon and DUO.IdentificatieMens to the external domain:

M1.1: Identifiers[{SGR.Persoon.Burgerservicenr ; DUO.Mens.identificatieMens}, attribute-entity]

Identifiers[{SGR.Persoon.Burgerservicenr ; DUO.IdentifiecatieMens.burgerservicenummer}, external domain]

For M2 we have to map the identifier of SGR.OpleidingsnaamGecodeerd to the identifier of DUO.Opleiding. We find that the identifier from SGR.OpleidingsnaamGecodeerd refers to a table of values in the CWI domain, conflicting with the identifier from DUO.Opleiding which refers to a table of values in the OCW domain. Since we can create a conversion table between the values, we define the relationship to be functional as it requires a mapping between the tables they refer to:

M2.1: Identifiers[{SGR.OpleidingsnaamGecodeerd.CdOpleidingsnaam ; DUO.Opleiding.Opleidingscode}, functional]

For M3 we found that they do share the same real world instances, but that there is no common identifier between the entities.

M3.1: Identifiers[{SGR.Opleiding.id ; DUO.Onderwijsdeelname.id}, conflict]

This creates a big problem when integrating the information from the DUO system into the SGR system. Since there is no common identifier, instances of these entities cannot automatically be matched with each other. We now have two options. Either we do not integrate the two, in which case we would have to remove the 1-to-1 mapping, or we accept duplicate entries in the system. For illustrative purposes we decide to accept duplicates in this case study.

Attribute Comparison

Since the project is about one-way communication, we only need to check each attribute of the entities from SGR, to see if there is a similar attribute in the DUO system. As we are only interested in integrating the information about the participation in education programs, we do not need to look at the attributes from the entities describing a human being (M1) as we can already match instances by their identifiers.

We start by looking at the attributes from M2. Entity SGR.OpleidingsnaamGecodeerd has four attributes, of which CdOpleidingsnaam was already compared in the identifier comparison. The remaining three attributes are described in Table 5-1.

SGR.OpleidingsnaamGecodeerd	
Attribute	**Description**
OmsOpleidingsnaam	The name of an education program, derived from an external domain
CdSrtOpleidingsnaam	Indicates whether the name of de education program is the name used by the *CWI* for matching and mediation. Two values are possible: (4) Reference by CWI, (5) Synonym
IndOpleidingsnaamActief	Indicates whether the name of the education program belongs to the active list of the external domain. It can have two values: (1) Yes, (2) No

Table 5-1: Attributes of SGR.OpleidingsnaamGecodeerd

We then search for similar attributes in the DUO system and utilize the decision tree to find the right classification (Appendix D). We found that from the three attributes, only OmsOpleidingsnaam could be related to an attribute in the DUO system. Since the possible values for both attributes in the relationship are defined by a different external table, we find the relationship to be functional as it requires a conversion table that links the values in the DUO system (derived from the OCW domain) to the appropriate values in the SGR system (derived from the CWI domain):

M2.2: Attributes[{SGR.OpleidingsnaamGecodeerd.OmsOpleidingsnaam ; DUO.Opleiding.NaamOpleidingLang}, functional]

Next, we look at the attributes from M3. A description of the nine attributes is listed in Table 5-X.

SGR.Opleiding	
Attribute	**Description**
DatBVolgenOpleiding	The date of the first day the CLIENT SUWI is enrolled in the education program.
DatEVolgenOpleiding	The data of the last day the CLIENT SUWI is enrolled in the education program.
CdStatusOpleiding	Code that represents the stadium of the education program. It can have one of four values: (0) Unknown, (1) Successfully ended, (2) Cancelled, (3) Running

IndDiploma	Code that represents if the CLIENT SUWI received a diploma or certificate for the education program. It can have one of four values: (0) Unknown, (1) Yes, (2) No, (3) Not applicable
AantJarenSuccesvolAfgerond	When no diploma has been received: the number of years successfully completed
AantUrenOpleiding	Number of hours and minutes the CLIENT SUWI is attending the education program (on average per week)
CdTijdsbeslagOpleiding	Code that represents if the education program, given the time consumption, is of influence for the labor reintegration process. It can have one of six values: (0) Unknown, (1) Daytime Course, (2) Evening Course, (3) Day + Evening Course, (4) By letter, (5) Other
IndDeeltijdopleiding	Code that represents if the course if followed part-time. It can have two values: (1) Yes, (2) No
ToelOpleiding	A characterization of the course. This is only used when there is no link to Education Name Coded available.

Table 5-2: Attributes of SGR.Opleiding

Again, we utilize the attribute comparison decision tree to find and classify related attributes in the DUO system (Appendix D). This resulted in the following relations.

SGR.Opleiding.DatBVolgenOpleiding is similar to attribute DUO.Onderwijsdeelname.datum-Inschrijving.

M3.2: Attributes[{SGR.Opleiding.DatBVolgenOpleiding ; DUO.Onderwijsdeelname.datumInscrijving}, 1-to-1]

SGR.Opleiding.DatEVolgenOpleiding is similar to attribute DUO.Onderwijsdeelname.datum-Uitschriving.

M3.3: Attributes[{SGR.Opleiding.DatEVolgenOpleiding ; DUO.Onderwijsdeelname.datumUitschrijving}, 1-to-1]

The third attribute, SGR.Opleiding.CdStatusOpleiding, is an interesting case. There is no similar attribute in the DUO system, but its value could be calculated from combining several. However, its value can also be calculated by combining data from other attributes of its own entity, which raises the question why the attribute exists. For this reason, it is unnecessary to list the attribute here, internal functions can be written to get its value.

SGR.Opleiding.IndDiploma does not have a similar attribute in DUO.Onderwijsdeelname, but is related to the separate entity DUO.Examenuitslag. Since that entity does not have a direct relationship with DUO.Onderwijsdeelname, we cannot qualify the relationship as an attribute-entity conflict. Instead, we add DUO.Examenuitslag to the third mapping, thereby changing M3 to an aggregation conflict.

The value of DUO.Examenuitslag.codeUitslag describes the result of the exam taken, while SGR.Opleiding.IndDiploma only describes whether a diploma has been received. We can therefore classify this as a generalization conflict:

M3.4: Attributes[{SGR.Opleiding.IndDiploma ; DUO.Examenuitslag.codeUitslag}, generalization]

The following attribute in the list is SGR.Opleiding.AantalJarenSuccesvolAfgerond. The attribute is found to have a relationship with the attribute leerjaar from the entity DUO.LeerjaarDeelname. Since DUO.Onderwijsdeelname has a direct relationship with this entity, we qualify the relationship as an attribute-entity conflict:

M3.5: Attributes[{SGR.Opleiding.AantalJarenSuccesvolAfgerond ; DUO.Onderwijsdeelname.leerjaarDeelname}, attribute-entity]

Attributes[{SGR.Opleiding.AantalJarenSuccesvolAfgerond ; DUO.LeerjaarDeelname.leerjaar}, 1-to-1]

SGR.Opleiding.AantUrenOpleiding does not have a similar attribute in the DUO system, thus is added to the schema-isomophisms.

SGR.Opleiding.CdTijdsbeslagOpleiding was found to have a similar attribute in the entity DUO.Onderwijsvorm. Since no direct relationship between DUO.Onderwijsdeelname and DUO.Onderwijsvorm. exists, the latter is added to M3. We then compare the similar attributes and find a 1-to-1 relationship:

M3.6: Attributes[{SGR.Opleiding.CdTijdsbeslagOpleiding ; DUO.Onderwijsvorm.Onderwijsvormcode}, 1-to-1]

SGR.Opleiding.IndDeeltijdopleiding is also similar to an attribute of DUO.Onderwijsvorm. Since IndDeeltijdopleiding only stores whether the education is a part-time, while Onderwijsvormcode also describes what form of education it is when it is not part-time, we classify the relationship as a generalization:

M3.7: Attributes[{SGR.Opleiding.IndDeeltijdopleiding ; DUO.Onderwijsvorm.Onderwijsvormcode}, generalization]

The last attribute in the list is SGR.Opleiding.ToelOpleiding. The information captured by this attribute is similar to the information in DUO.Opleiding.studieinhoud. As DUO.Onderwijsdeelname and DUO.Opleiding are not directly related, we add the latter to M3. We further find the relationship between the attributes to be 1-to-1:

M3.8: Attributes[{SGR.Opleiding.ToelOpleiding ; DUO.Opleiding.studieinhoud}, 1-to-1]

After the attribute comparison of M3, we have made several changes to the entity relationship, changing it to an aggregation relationship. The updated entity relationship diagram is displayed in figure 5-5.

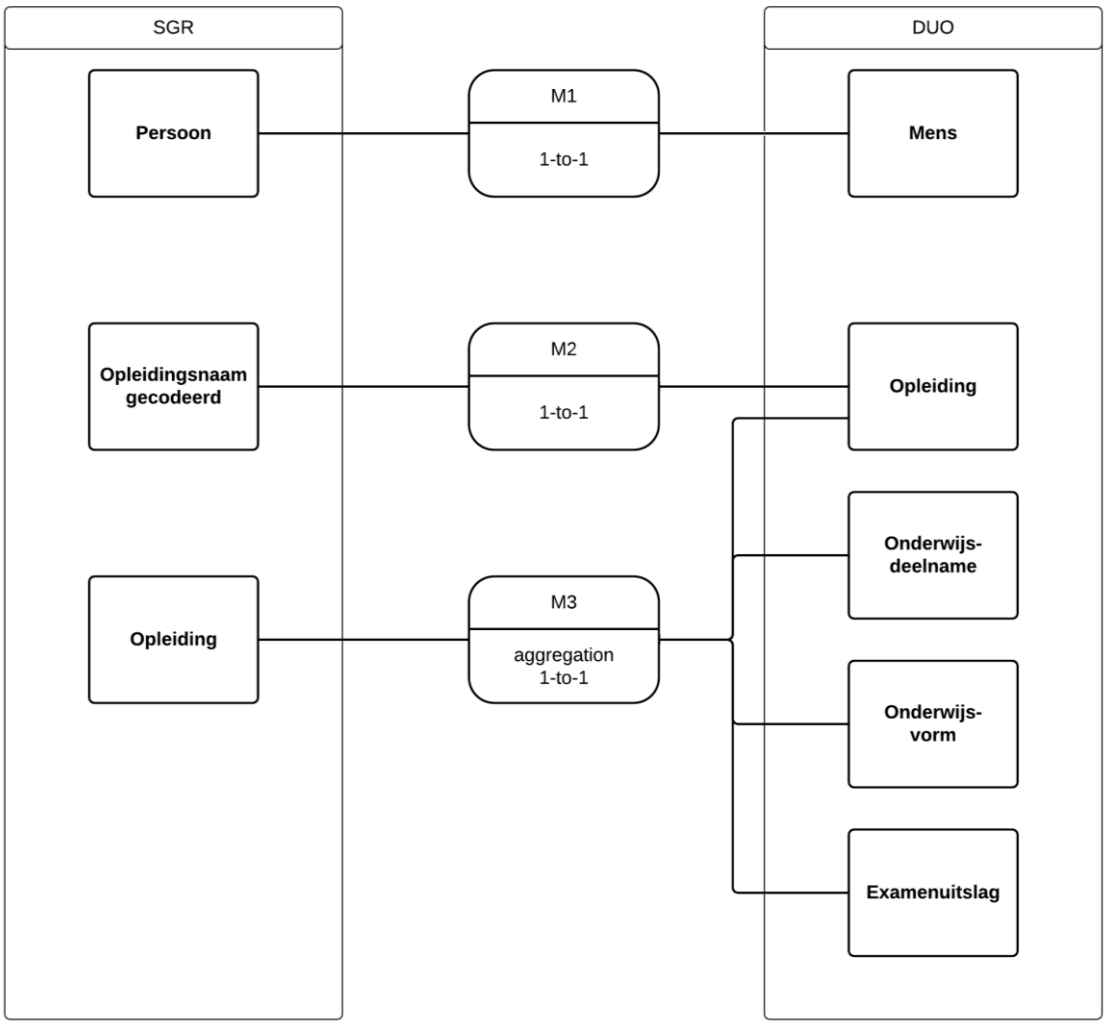


Figure 5-5: Updated Entity Comparison diagram SGR-DUO

Data Format Comparison

For each attribute mapping, we compare their data formats to check for potential conflicts. This is performed (Appendix E) by filling in the table with the characteristics of the data format used by each attribute, as proposed by the methodology. We now discuss the results of the comparison.

For M1.1 we check de data formats of SGR.Persoon.Burgerservicenr and DUO.Identificatie-Mens.burgerservicenummer. Since both attributes are derived from an external domain we expect the data formats to have no conflicts. Further investigations shows that both are 9 characters long, and have the same control measures to check for validity. However, SGR.Persoon.Burgerservicenr is defined as a numerical value, while DUO.IdentificatieMens.burgerservicenummer is defined as a string. We thereby have a *data representation* conflict, as a string cannot be inserted into an attribute that requires an integer.

M1.1: Data representation conflict

M2.1 has a functional relationship, so we have to convert the values anyway. Therefore, the differences in data format and the number of characters used does not add any new problems to the matter.

M2.1: Data representation conflict.
Integrity constraints conflict

The conflicts found in M2.2 are similar to M2.1, although there is no data representation conflict in this one, they both use a string to store their values.

M2.2: Integrity constraints conflict

Since M3.1 is mapped as a conflict and thus we cannot exchange values, there is no use in comparing its data format. We therefore skip this attribute pair.

Both M3.2 and M3.3 have a data representation conflict, as the system designers have chosen a different way to notate the date.

M3.2: Data representation conflict

M3.3: Data representation conflict

M3.4 shows a difference in both the data format, and the range of values the attribute can have. Since the relationship between the attributes is a generalization, we can integrate the data format conversion in writing the function for the generalization.

M3.4: Data representation conflict
Integrity constraint conflict

M3.5 involves a data format comparison between SGR.Opleiding.AantJarenSuccesvolAfgerond and the other end of the attribute-entity relationship, DUO.LeerjaarDeelname.leerjaar. The comparison results in a data representation-, and integrity constraints conflict. They use a different data format to store the value, and a different range the value can have. The SGR system has a maximum value of 99, thus allowing two numbers. The DUO system only allows one number, and thus has a maximum value of 9.

M3.5: Data representation conflict
Integrity constraint conflict

Comparing the data formats of M3.6 results in a data representation-, and integrity constraints conflict. The conflicts are a result from the difference in codes the attribute can have. The DUO system uses letter combinations as a code, contrary to the numeric code from the SGR system. Also, the DUO system has more options available than the SGR system. As the relationship now requires a mapping, we change its relationship to functional.

M3.6: Data representation conflict
Integrity constraint conflict

The data comparison results for M3.7 are similar to that of M3.6. We also have a data representation-, and integrity constraint conflict, as a result of the difference in the code being used to represent the value. Since the relationship was already defined as a generalization, we can address the data format conflicts when writing the generalization function.

M3.7: Data representation conflict
Integrity constraint conflict

Only one conflict was found in M3.8, being an integrity constraint. There was a difference in the maximum number of characters

M3.8: Integrity constraint conflict

Data Value Conflict Policy

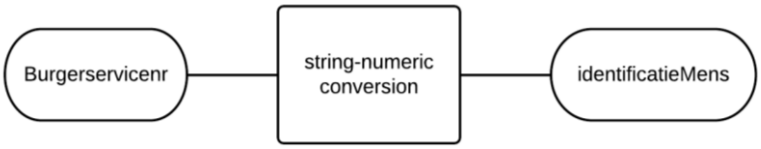
In this case study, we have identified a conflict between the two instances that will be exchanging information: M3. Since the identifiers cannot be matched, we have to set a policy where duplicate entries are allowed. For this reason, there is no possibility of conflicting data values.

Conflict Resolution

For each of the three mappings, we will try and solve the conflicts that were identified.

Mapping 1

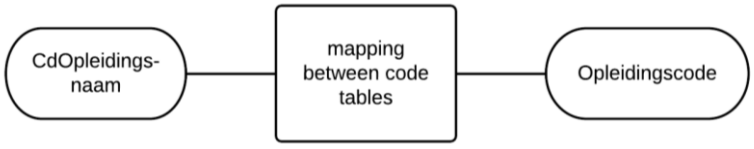
For M1 we identified one conflict at the entity level, and one at the data format level. At the entity level we found an attribute-entity conflict for the identifiers. This means that for all future queries targeting instances at the DUO.Mens entity, we have to extend that query to also query DUO.IdentificatieMens with the relational key value that DUO.Mens holds. The value in the response then needs to be converted from a string to an integer.



Mapping 2

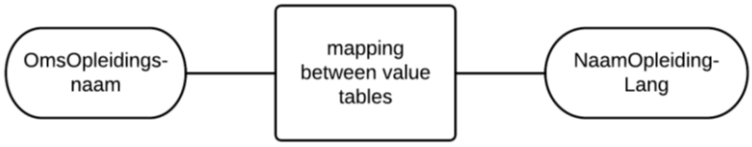
M2.1

In M2.1 we have a functional relationship, where a mapping needs to be made to convert one value into another. Since the mapping implies that we convert the data format from one attribute to the other, the conflicts found at the data format level are instantly solved.



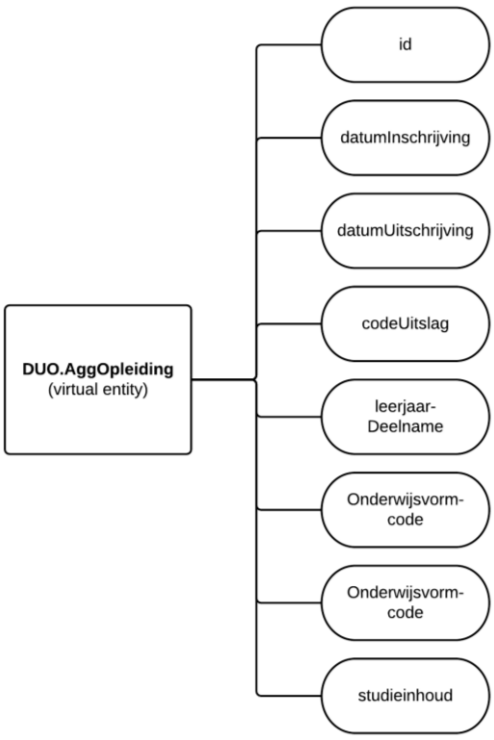
M2.2

The problem in M2.2 is similar to that in M2.1, so we construct a conversion table to map the instance values from *OmsOpleidingnaam* to *NaamOpleidingLang*.



Mapping 3

As M3 has an aggregation relationship between the entities, we need to create a virtual entity for the DUO system, that combined the entities in the relationship. The new virtual entity is displayed below:

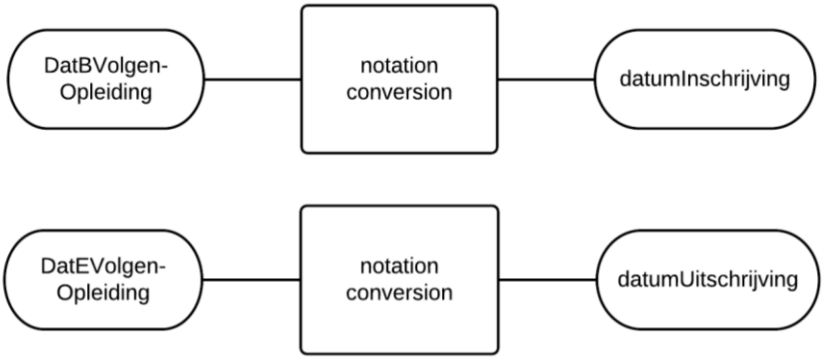


M3.1

As stated before, the identifiers are totally unrelated to each other, so there is no conversion possible.

M3.2 and M3.3

Since M3.2 and M3.3 are very similar, we discuss these at once. They are both defined as a 1-to-1 relationship, so we can directly exchange values. However, since their data formats are a bit different, we need to convert the value from one notation to the other (YYYMMDD to YYYY-MM-DD).



M3.4

We defined *IndDiploma* to be a generalization of *codeUitslag*, thus requiring a function or mapping that converts the instance values from *IndDiploma* to *codeUitslag*. In this case a mapping is more suitable, as both attributes use a very specific range of possible values. Table 5-3 provides a

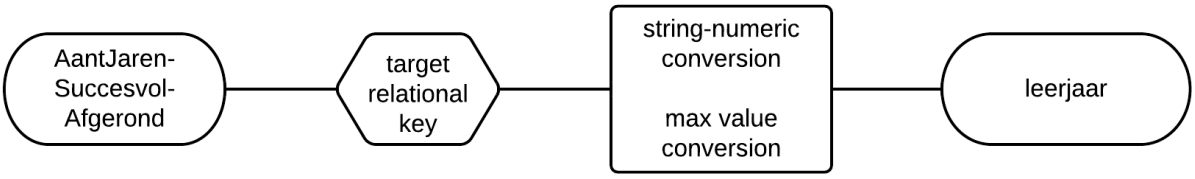
suggestion for the value mapping, but this would have to be verified by a domain expert of both organizations.

codeUitslag	*IndDiploma*
G	1
A, T, E	2
D	8

Table 5-3: Possible attribute mapping M3.4

M3.5

M3.5 includes an attribute-entity conflict, therefore we need to use the relational key in DUO.Onderwijsdeelname to find the right instance in DUO.LeerjaarDeelname. If we then look at the relationship between the two related attributes, we found a data representation and integrity constraint conflict. At first, we need to make a conversion from a string to a numeric format. We then also need to decide what to do with values above 9, as this is the maximum value for *leerjaar*, resulting from a difference in the maximum number of characters.



M3.6

The mapping between *CdTijdsbeslagOpleiding* and *Onderwijsvormcode* was changed into a functional relationship following the data format comparison, as the exchange of instances requires a mapping between the values. Table 5-4 proposes a mapping between the two, but this would have to be verified by a domain expert of each system.

Onderwijsvormcode	CdTijdsbeslagOpleiding
VT, DU	1
DT	2
SC	4
AB, LWOO, LWT, BOL, BBL, VAVO	0

Table 5-4: Possible attribute mapping M3.6

M3.7

We defined *IndDeeltijdOpleiding* to be a subset of *Onderwijsvormcode*. As such, we need to define a function that transforms the exchanged information to the scope of the receiver. The function has to be written with a domain expert of each system. An example of such a function is shown below.

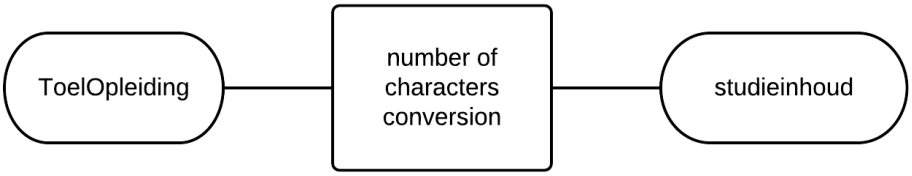
```
If (IndDeeltijdOpleiding == 1)
    Onderwijsvormcode = DT
else if (IndDeeltijdOpleiding == 2)
    Onderwijsvormcode = VT
```

else

Onderwijsvormcode = *default*

M3.8

The two attributes in M3.8 are defined to have a 1-to-1 relationship. The only conflict between the two is the maximum number of characters (180 vs. 150). We therefore need to create a rule about what to do with values longer than 150 characters.



Key Findings

In this chapter we applied our methodology to a data integration project of *Dienst Uitvoering Onderwijs (DUO)* and the *SUWI Gegevensregister (SGR)* in the Netherlands. In this case study we compared the semantics of the DUO system to those from the SGR system, and identified all conflicts when sending the required information from DUO to SGR. The methodology was found easy to apply on the project and a number of semantic conflicts are presented. From the guidelines that the methodology offers we were able to provide solutions for each of these conflicts. In the next chapter we will validate the results with those found by the problem holder.

6. Research Validation

Research validation is a process of building confidence in its usefulness with respect to a purpose. We associate usefulness of a design method with whether the method provides design solutions 'correctly' (effectiveness), and whether it provides 'correct' design solutions (efficiency) (Pedersen et al., 2000).

The Validation Square (Figure 6-1) addresses the effectiveness and the efficiency of the proposed method. Effectiveness covers structural validity, efficiency covers performance validity. Both have to address the theoretical-, as well as the empirical dimension. We will discuss the validity of the proposed method for semantic conflict identification at each of the four constructs of the Validation Square.

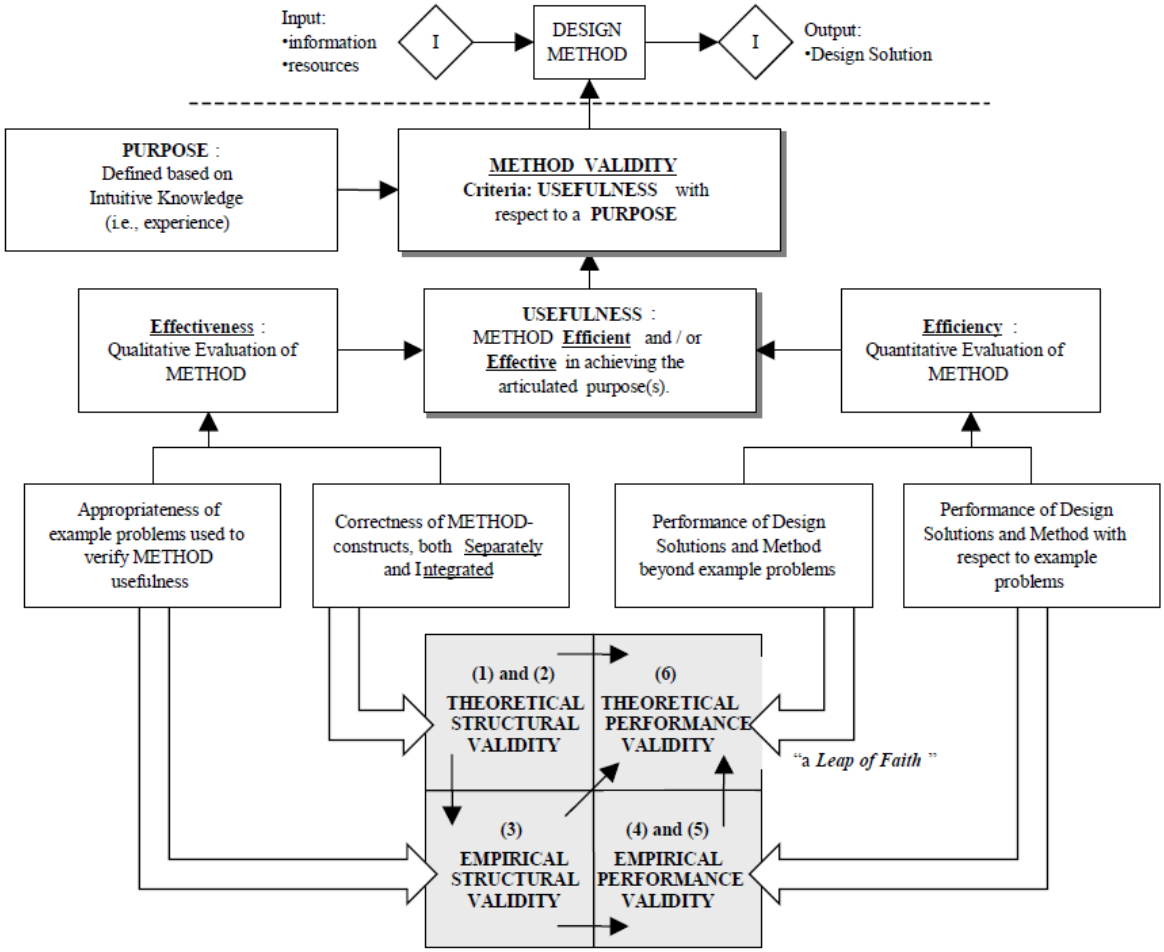


Figure 6-1: Validation Square (Pedersen et al, 2000)

Theoretical Structural Validity

The correctness of the method constructs is evaluated along two dimensions: (1) accepting the construct's validity, and (2) accepting method consistency. The first dimension covers the credibility of the research literature used to build the methodology, the second dimension challenges the confidence in the way the constructs are put together in the method. We now discuss both dimensions.

Accepting the construct's validity

Much of the constructs used in the methodology, are based on the findings from the literature study. We now discuss the constructs for:

1. The methodology stages
2. The modeling language used to model the systems
3. The identification and categorization of semantic conflicts
4. The mapping technique used to present the findings

The methodology stages

The development stages (steps) in the methodology are mostly constructed from Batini and Lenzerini (1984) and Haslhofer and Klas (2010). The first paper has been published in the top journal *ACM Computing Surveys*, and received many citations (388). The second paper was also published in a top journal, with few citations because of its young age. We regard both of these as generally accepted papers, and thus provide a solid foundation for the construction of the development stages of our methodology. Additionally, literature from Daclin et al. (2008), Ralyte et al. (2008), and Ram and Ramesh (1999), provides similar results. Their papers are less credible, but as we state, they only provide additional support for the constructed development stages.

Constructs	Published in	Citations
Batini and Lenzerini (1984)	*ACM Computing Surveys (24)*	388
Ralyte et al. (2008)	*Information Systems (-)*	6
Daclin et al. (2008)	*17th IFAC proceedings (-)*	0
Ram and Ramesh (1999)	*Management of Heterogeneous and Autonomous Database Systems*	12
Haslhofer and Klas (2010)	*ACM Computing Surveys (24)*	5

Table 6-1: Constructs for methodology stages

The modeling language used to model the systems

In the second stage of the methodology, 'schema translation', we use the Entity Relationship modeling language to model the concepts present in each system. The ER model was presented by Chen (1976) and is since one of the accepted languages to model information system structures.

The identification and categorization of semantic conflicts

In the third stage, 'mapping discovery', we identify the semantic conflicts between the systems. The basis of this stage is the categorization of semantic conflicts as presented in chapter four. The categories are constructed from an aggregation of research papers by Park and Ram (2004), Madnick and Zhu (2005), El-Khatib et al. (2000), Naiman and Ouksel (1995), Sheth and Kashyap (1992), and Kim et al. (1993). The papers vary in credibility, but we found none of the papers to be conflicting with each other. By combining the knowledge from these research papers, we believe to have constructed a valid categorization of semantic conflicts.

Constructs	Published in	Citations
Park and Ram (2004)	*ACM Transactions on Information Systems (13)*	67
Madnick and Zhu (2005)	*Data and Knowledge Engineering (-)*	9
El-Khatib et al. (2000)	*Information and Software*	5

	Technology (-)	
Naiman and Ouksel (1995)	*Journal of Organizational Computing (31)*	47
Sheth and Kashyap (1992)	*IFIP Transactions A: Computer Science and Technology (-)*	9
Kim et al. (1993)	*Distributed and Parallel Databases (-)*	46

Table 6-2: Constructs for Semantic Conflict categorization

The mapping technique used to present the findings

The mapping representation technique is based on the one used by Haslhofer and Klas (2010). This paper is published in the widely respected journal *ACM Computing Surveys*. Since it was only published in 2010, there are not a lot of citations yet. But based on the journal, we regard this as a valid construct for the mapping representation.

Constructs	Published in	Citations
Haslhofer and Klas (2010)	*ACM Computing Surveys (24)*	5

Table 6-3: Constructs for mapping technique

Furthermore, the theoretical framework from which the three interoperability approaches have been derived, the Framework for Interoperability, is defined as an ISO standard (11354). Being certified as such, the framework is recognized as credible by the industry.

Accepting method consistency

In order to build confidence in the method's internal consistency, we constructed the methodology following the Information Engineering Methodology Description Model by Heym and Österle (1992). This model divides the method in several clearly distinguishable steps, each with a clearly defined input and output. Figure 6-2 represents a flowchart with the information flow across the various steps of the methodology.

Preintegration

As we explained in the construction chapter, the method starts with the assumption that each organization involved in the project provides the necessary information to successfully integrate the systems. This includes the database schemas and accompanying description of entities and attributes. Without this information it is impossible to search for related entities and attributes, thereby making the methodology unusable. The collection process of this information is represented by the 'Preparation' object in Figure 6-2.

Schema Translation

In the schema translation stage we search the database schemas for the parts that store the concepts defined in the objectives & integration policy document. This means that the quality of the output from the previous stage directly influences the quality of the findings in this stage. Obviously, the database schemas must include the sections necessary for the integration project. The objectives must be clearly defined, i.e. what concepts will be integrated and how these relate to each other.

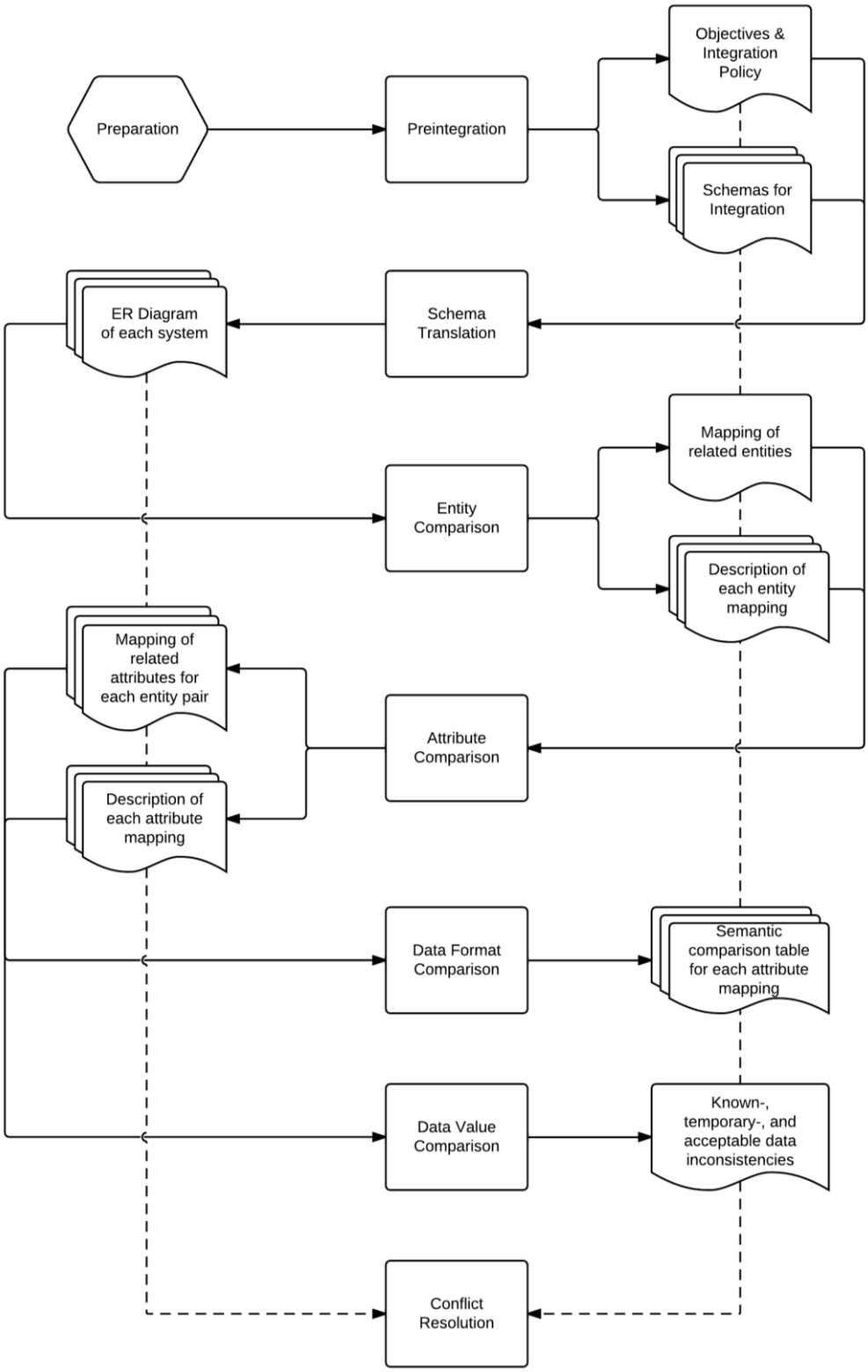


Figure 6-2: Methodology information flow

Entity Comparison

If the ER-diagrams are well constructed, and the entities assumed to represent the concepts in the objectives are valid, we can easily draw the relationships between the entities in the different systems. More complex is the characterization of the relationship. A generalization or aggregation

might not always be identified, thus creating the wrong output. However, these errors should be identified in the next stages after which we iterate back to this stage.

Attribute Comparison
Besides identifying related attributes between the systems, the attribute comparison stage also provides validation for the output from the previous stage. By comparing their attributes, we could find two entities previously assumed to be related, to be not. From their respective attributes, we could also find an entity to be a generalization of the other, which we hadn't identified during the entity comparison. Also, when we find one entity having attributes that are defined in a separate entity in the other system, we change the entity relationship into an aggregation.

Data Format Comparison
In this stage we compare the data format for each pair of related attributes documented in the output from the previous stage. The quality of the results from that output is not of direct influence on the usefulness of this stage. Wrong assumptions about the relationship between attributes, does not hinder us to compare their data formats. At the same time, by comparing their data formats, we could discover that the attribute values cannot be related to each other, or only by a mapping or function. We then may need to remove the assumed relationship, or modify it to a functional relationship. In that way the data format comparison provide validation for the results in the previous stage.

Data Value Comparison
The data value comparison does not validate results from previous stage, but it does use the output from the entity comparison as a starting point. For each attribute pair listed we think about potential data inconsistencies, and if these are considered to be a problem.

Conflict Resolution
All deliverables from previous stages are used for the conflict resolution process. By utilizing the guideliness provides by this methodology, we can resolve the identified semantic conflicts.

Empirical Structural Validity

Empirical structural validity is claimed in three steps. First, we explain that the case study problem is similar to the problems for which the constructs are developed. Next, we explain that the problem represents what the methodology was originally intended to solve. Finally, we argue that the findings from the case study can support a conclusion.

The development stages provided by Batini and Lenzerini (1984) are a generalization for the steps they found in the twelve database integration methodologies studied. Although we are not merging the two database as in the methodologies they studied, the problems to overcome in the first stages of the project are the same as in our case study. Haslhofer and Klas (2010) focus on metdata interoperability, as a "prerequisite for uniform access to media object in multiple autonomous and heterogeneous information systems". This is exactly the focus of the case study. The two involved systems have to remain autonomous but need to access each other's objects. We can therefore say that the case study problem is the same as these constructs are designed for.

Semantic conflicts as defined in the studied literature, is defined as "differences in implicit meanings, perspectives, and assumptions" (Park and Ram, 2004) that arise between independently designed

information systems. In our case study, the DUO system is independently developed from the SGR system, thus being the situations for which the construct was intended.

Haslhofer and Klas (2010) provide the following definition for the scope of the mapping technique proposed:

> "Given two metadata schemes, both settled in the same domain of discourse and expressed in the same schema definition language, we define metadata mapping as a specification that relates their model elements in a way that their schematic structures and semantic interpretation is respected on the metadata model and on the metadata instance level."

Since we constructed an ER diagram for each systems subject to the case study, they are expressed in the same schema definition language, thus making the representation technique applicable. Also, the schematic structure and semantic interpretation of each system is respected as we do not alter their autonomy.

The original goal of this research was to develop a structured approach to identify semantic conflicts between heterogeneous information systems, so that we can address these problems before occurring in an interoperability project. Clearly, our case study represents the problem that our methodology tries to solve. In the case study we have two independently designed information systems, and one wants to get connected to the other, thus creating a good possibility of having semantic conflicts. By utilizing our methodology in the case study, we discover how the method behaves when applied to real problems. By comparing the results to the semantic conflicts found by the case holder, we also learn more about its performance.

Empirical Performance Validity

Empirical performance needs to be validated along two dimensions. The first dimension describes how useful the method was for its intended purpose. The second dimension compares the results with those found by utilizing another methodology, so we can check if the usefulness of the methodology is linked to its application.

If we compare the results found in the case study with those found by the project team using the methodology from the Essence project, we notice several differences. Some of these differences come from a different approach to the project. Since we identified in the objectives setting that this project was only concerned with an information flow from the DUO system to the SGR system, we did not pay attention to potential semantic conflicts in the other direction. In the Essence project, the semantics of each concept is first defined in its own context, and then compared to each other. This approaches thus leads to finding semantic relationships in both directions. Let us illustrate this with the different result found for M2:

We defined a 1-to-1 relationship between SGR.OpleidingsnaamGecodeerd and DUO.Opleiding. That is, we found that every instance from DUO could be related to an instance in SGR. However, if we would compare these entities in the other direction, we see that not every instance of SGR can be related to an instance of DUO. Thereby, these results would lead to a generalization relationship, as identified by the Essence team.

With our methodology, identifying a generalization relationship between two entities means we would have to create a filter between the two, as proposed in the methodology design (section 'conflict resolution'). But since every instance from DUO that is transferred to SGR would pass that filter, and no exchange takes place in the other direction, constructing such would be an inefficient activity for this project. We therefore accept the different label for this relationship as such, neither of them being wrong.

If we look at the differences in findings for M3, we learn that M3.2 and M3.3 are falsely identified as being the same. As the Essence methodology discovered, the date of subscription in the DUO system plays a fundamentally different role in its context than that of the date of participation in the program as defined in the SGR system. In the context of DUO, the system captures the date that a person is formally subscribed to an education program. In the context of SGR, the date represents the day that someone physically participates in the program, thus being unavailable for the labor market. If a person is subscribed to a program, but starts participating at a later date, he or she would falsely be denied from a receiving a social security payment in-between.

The reason that we defined the relationship to be 1-to-1, was the assumption that the date from DUO could be used in the SGR system without problems. The methodology misses the step where this assumption has to be verified by a domain expert of each organization, as that would have led to removal of the relationship. In some cases, the data format comparison can identify false assumption at the attribute level, but not in cases where the conflict is context related.

For M3.4 we found a generalization. Although the Essence team identified the two attributes as being related, they don't classify this as a generalization. The reason for this is similar to what we explained for M2. As SGR uses a much broader scope for what an education is (also includes workshops, courses etc.), the instance set for IndDiploma is also much broader than that of codeUitslag. Again, if we only study the one-way direction from DUO to SGR, we can match every instance, thereby it is possible to label the relationship as a generalization.

Theoretical Performance Validity

The theoretical performance validity looks at the usefulness of the method beyond the example problems. To test this "general belief" in its usefulness, we asked two expert in this field of work to take a critical look at the proposed methodology. We now discuss their criticism.

The main criticism expressed is that the methodology does not provide clear instructions of how to determine the semantic equivalence of two concepts. In the entity and attribute comparison, we take a concept and search for a semantically related concept in the other system. For some concepts this may be clear, but exactly how do we know if they do are the same? With our methodology, the correctness of the relationships found depends on the quality of each system's documentation. If the description of the concepts is a bit vague, we could make the wrong assumptions, and thus falsely define two concepts to be the same.

Another point of criticism expressed by both experts is the swapping of identifiers during the entity comparison process. The methodology suggests that we can use the identifier of a related entity (C) when comparing two entities (A and B). In the case study, this is performed in M1, where we use the attribute *burgerservicenummer* from the related entity DUO.IdentificatieMens to take the role of

identifier for DUO.Mens. The methodology does not provide conditions under which this is a legitimate assumption. Is the swap also allowed if C is indirectly related to B? What do we do if an instance of B is related to two instances of C? What if the identifier in A is related to an aggregation of attributes from entities C and D?

A third point addressed by one of the experts, is if the Entity Relationship modeling language is best suitable for the methodology. It is proposed that ORM (Object Role Modeling) is a much richer and more precise modeling language. As we argued in the method construction chapter, our selection is based on the wide acceptance of ER modeling, thus being understood by almost every potential user of our methodology. It is possible that other modeling languages provide better results, but it should be researched if this would be significant enough to compensate the extra effort required by the user.

Key Findings

In this chapter we validated the methodology by using the validation square by Pedersen et al. (2000). The framework consist of four dimensions: theoretical- and empirical structural validity, and theoretical- and empirical performance validity.

We conclude the method to be theoretical structural valid. The literature used to construct the methodology is widely accepted in its field of research, so we are confident that these are valid constructs. We also claim the method to be structured in a logical manner.

Secondly, we conclude that the situation in the case study is similar to the problems the constructs are designed for, and that the case study can thus be considered as a suitable validation instrument. We also argue that the problem in the case study is similar to what our methodology is intended to solve, thus it to be the right case to test the performance of the method.

Thirdly, from the case study we conclude that the methodology is useful to identify and resolve semantic conflicts between two independently developed systems. Not only was is easy to use and did it identify several conflicts, compared to the findings from the problem holder we conclude that the method did a decent job in finding the right conflicts.

Finally, from the expert review we raise three points of concern for the validity of the methodology beyond the case study. The first one is the method's ability to find related concepts. The second point of concern is the construction of the identifier comparison. And the final point of concern is whether the Entity Relationship diagramming language is the best one for the job.

In the next chapter we present our conclusions based on these key findings about the validity of the method constructed.

7. Conclusion

The aim of this research was to design a methodology that identifies and resolves semantic conflicts when connecting two independently designed information systems. The methodology was created with constructs found in the literature study, and was validated by means of a case study and an expert review. The case study was conducted at an information system integration project involving *Dienst Uitvoering Onderwijs* and the *SUWI Gegevensregister* in the Netherlands, providing insight into the applicability of its instruments in practice. We also conducted an expert review, so that we learned more about the usability of the methodology beyond the case study. We now present our conclusions for this research.

Key findings

In chapter two, following a literature study of methodology engineering, we defined five requirements for our methodology. We conclude that each of these requirements is met (Table 7-1).

	Methodology Requirement	Satisfied by
1.	*One output of the methodology should represent the user's requirements in formal terms.*	The first step in the methodology delivers a formal document representing a problem description and the project's goal, thus providing the ability to check the user's requirements for any inaccuracies, inconsistencies, or incompleteness.
2.	*The methodology needs to define several identifiable, logical stages, where the output of each stage is clearly defined.*	Using the evaluation framework by Pedersen et al. (2000) we claim the methodology to be logically structured and to be based on widely accepted constructs found in literature. The deliverables are clearly defined in the table at the end of each stage in chapter four.
3.	*The output from one stage preferably forms the input of the next stage.*	Figure 6-2 clearly shows that the output of one stage forms the input of the next.
4.	*The methodology must be useful as a method fragment in situational method engineering.*	The methodology can be used as a stand-alone technique to identify semantic conflicts. This is supported by the case study, where the results were found independently from the methods used for the whole interoperability project.
5.	*The method should be easy to use, understand and learn.*	The case study provides support for this, as the steps defined in the methodology were easy to apply on the data integration project subject to the study.

Table 7-1: Evaluation of the methodology requirements

In the case study, following the instructions from the methodology, we come to a list of semantic conflicts found between the two systems. After comparing the results with those found by the project holder, we conclude that the results were mostly correct, with some variations coming from a different design choice in the project's approach. Our methodology requires the user in the first stage to specify an interoperability approach and the target schemas. Since the case study is about one way data exchange, we improve the efficiency of the project as the methodology only searches for semantic conflicts when sending information in this one direction, opposite to the method used by the project holder where semantic relationships are defined when looking at both directions.

The case study further exposed that an important requirement for the usability of the method is the availability of a thoroughly documented description of the concepts stored in each system. And that

even when this is the case, the methodology needs additional control measures to check the validity of any assumption made during the process. We therefore suggest the implementation of a validity check after each stage, that will force a domain expert of each system to validate the relationships between the concepts and the assumptions made. Doing so would have corrected the mistakes we made in the case study.

Based on the results from this case study, we claim the methodology to be useful for at least the integration projects characterized by a federated approach with one-way data exchange, provided that a well formalized description of each system is available.

Practical Implications

When looking at the practical implications of this research, we believe that it offers a practical and structured approach to target the problem of differences in semantics when connecting two or more systems. Such a methodology was not yet available to organizations, thereby requiring them to construct their own methods and tools. This research thus contributes to a more efficient way to achieve interoperability, thereby increasing the competitiveness of the enterprise. Even if one would decide not to use the whole methodology, it still offers a set of practical tools that help to identify and resolve semantic conflicts at four different levels. Additionally, at each of these four levels we provide guidelines for how to resolve the conflicts.

Research Implications

This research contributes to the field of interoperability by making a first attempt to develop a standard approach for the identification of semantic conflicts in interoperability projects. By doing this, we provide tools to remove the conceptual barriers at the service level in the framework for interoperability (Figure 1-1). More research is necessary if we want to claim generality however. In this research, we have only taken one cycle in the design process (Figure 2-2). If we want to find the set of conditions for which the method works, and for which it does not, we have to conduct more case studies with different project characteristics.

A second contribution to the field of interoperability, is the semantic conflict categorization at the different conceptual levels commonly found in information systems diagramming languages. This is different from the categorizations found in literature, where segregations are made on the basis of the characteristics of the conflict. By distinguishing conflicts at the entity-, attribute-, data format-, and data value level, we make these easier to apply on existing interoperability- and data integration literature.

Limitations

A point of concern raised in the expert review is the methodology's ability to discover related concepts. In its current form, the method instructs the user to search for related concepts between the two systems, but does not provide instructions of how to find these, and when the user is allowed to claim a relationship between the two. The method thus depends much on the ability of the user to find valid relationships. Resolving this problem is a daunting task however. If we are to create a general applicable framework for the discovery of two related concepts, we will end up in a highly conceptual and philosophical field of work. This is contrary to our intention to offer a structural approach for semantic conflict detection that was easy to use and learn.

Further Research

To solve the problem listed under limitations, we suggest adding existing methods for finding related concepts as a method fragment (Brinkkemper, 1996), that can be used in cases where the users don't have the required domain knowledge to find the relationships themselves. These method fragments can be derived from literature in the field of ontology mapping, such as Wong et al (2005), and Jamadhvaja and Senivongse (2005).

Another interesting area for future research is the development of instrument guidelines and tools that support the user during the various stages of the methodology. For instance, we can think of a tool that supports easy documentation and representation of the findings from each stage. After each stage, the tool requires the user to create the required drawing, and provide the necessary additional documentation. The result of the tool is an easy to navigate overview of all the mappings. We image an overview of the mappings at the entity level, where a user can click each mapping to read the formalized description of the relationship. A double click brings the user to the attribute mapping of this entity pair, where he again can click a mapping to read more information about the relationship. Another double click brings the user to the data format- and data value conflicts found. The tool can also help the user solving each semantic conflict by providing suggestions based on the conflict type and concept characteristics.

Finally, it would be interesting to research the best diagramming language to model the systems in the first stage. As proposed in the expert review, other languages such as ORM (Object Role Modeling) provide much richer information, and thus could potentially improve the efficiency in the next stages. It would have to be researched if the extra effort that is required to learn and use ORM (or any other modeling language) is compensated by the reduced effort in the next stages.

8. References

Archer, L.B. (1984), *Systematic method for designers*, Developments in Design Methodology, pp. 57-82.

Avison, D., G. Fitzgerald, A. Wood-Harper (1988), *Information systems development: a tool kit is not enough*, The Computer Journal, Vol. 31 (4), pp. 379-380

Bantleman, J.P., *A Feature Analysis of the LBMS System Development Method*, Structured Methods State of the Art Report, Vol. 12

Batini, C., M. Lenzerini (1986), A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, Vol. 18 (4), pp. 323-364

Brinkkemper, S. (1996), *Method Engineering: engineering of information systems development methods and tools*, Information and Software Technology, Vol. 38, pp. 275-280

Catchpole, P. (1986), *Requirements for a Successful Methodology in Information-Systems Design*, Data Processing, Vol. 28 (4), pp. 207-210

Chatzoglou, P.D. (1997), *Use of methodologies: an empirical analysis of their impact on the economics of the development process*, European Journal of Information Systems, Vol. 6 (4), pp. 256-270

Chen, D., G. Doumeingts, F. Vernadat (2008), *Architectures for enterprise integration and interoperability: Past, present and future*, Computers in Industry, Vol. 59, pp. 647-659

Curtis, B., H. Krasner, N. Iscoe (1988), *A field-study of the software-design process for large systems*, Communications of the ACM, Vol. 31 (11), pp. 1268-1287

Daclin, N., D. Chen, B. Vallespir (2008), *Methodology for Interoperability*, in Proceedings of the 17th World Congress The international Federation of Automatic Control

Eekels, J., N.F.M. Roozenburg (1991), *A methodological comparison of the structures of scientific research and engineering design: their similarities and differences*, Design Studies, Vol. 12 (4), pp 197-203

El-Khatib, H.T., M.H. Williams, L.M. MacKinnon, D.H. Marwick (2000), *A framework and test-suite for assessing approaches to resolving heterogeneity in distributed databases*, Information and Software Technology, Vol. 42, pp. 505-515

Gagnon, M. (2007), *Ontology-based integration of data sources*, Proceedings of the 10th International Conference on Information Fusion, Vol. 1-4, pp. 896-903

Goh, C.H., S. Bressan, S. Madnick, M. Siegel (1999), *Context Interchange: New Features and Formalisms for the Intelligent Integration of Information*, ACM Transactions on Information Systems, Vol. 17 (3), pp. 270-293

Harmsen, F., S. Brinkkemper, H. Oei (1994), *Situational method engineering for information system project approaches*, Proceedings of the IFIP WG8.1 Working Conference CRIS'94 Maastricht, pp. 169-194

Haslhofer, B., W. Klas (2010), *A Survey of Techniques for Achieving Metadata Interoperability*, ACM Computing Surveys, Vol. 42 (2), Article 7

Hevner, A.R., S.T. March, J. Park, S. Ram (2004), *Design Science Information Systems Research*, MIS Quarterly, Vol. 28 (1), pp. 75-105

Heym, M., H. Österle (1992), *A semantic data model for methodology engineering*, in Proceedings of the 5th International Workshop on Computer-Aided Software Engineering, pp. 142-155

Fagin, R., P.G. Kolaitis, R.J. Miller, L. Popa (2005), *Data Exchange: Semantics and Query Answering*, Theoretical Computer Science, Vol. 336, pp. 89-124

Jamadhvaja, M., Senivongse, T. (2005), *An Integration of Data Sources with UML Class Models based on Ontological Analysis*, Proceedings of the first international workshop on Interoperability of heterogeneous information systems

Jenkins, A., J. Naumann, J. Wetherbe (1984), *Empirical investigation of systems development practices and results*, Information & Management, Vol. 7 (2), pp. 73-82

Kim, W., I. Choi, S. Gala, M. Scheevel (1993), *On Resolving Schematic Heterogeneity in Multidatabase Systems*, Distributed and Parallel Databases, Vol. 1, pp. 251-279

Konstantas, D., J.-P. Bourrières, M. Léonard, N. Boudjlida (2006), *Interoperability of enterprise software & applications*, Springer

Madnick, S.E. (1999), *Metadata Jones and the Tower of Babel: The Challenges of Large-Scale Semantic Heterogeneity*, Proceedings of the 3rd IEEE Meta-Data Conference

Madnick, S., H. Zhu (2006), *Improving Data Quality Through Effective Use of Data Semantics*, Data & Knowledge Engineering, Vol. 59, pp. 460-475

March, S., A. Hevner, S. Ram (2000), *Research commentary: An agenda for information technology research in heterogeneous and distributed environments*, Information Systems Research, Vol. 11 (4), pp. 327-341

Ministry of economic affairs (2007), Nederland in Open Verbinding: Een actieplan voor het gebruik van Open Standaarden en Open Source Software bij de (semi-) publieke sector, November 2007. http://www.ez.nl/dsresource?objectid=153181&type=PDF

Naiman, C.E., A.M. Ouksel (1995), *A Classification of Semantic Conflicts in Heterogeneous Database Systems*, Journal of Organizational Computing, Vol. 5 (2), pp. 167-193

Naudet, Y., T. Latour, W. Guedria, D. Chen (2010), *Towards a systemic formalization of interoperability*, Computers in Industry, Vol. 61, pp. 176-185

Nunamaker, J.F., M. Chen (1991), *Systems Development in Information Systems Research*, Journal of Management Information Systems, Vol. 7 (3), pp 89-106

Ouksel, A.M., I. Ahmed (1999), *Ontologies are not the Panacea in Data Integration: A Flexible Coordinator to Mediate Context Construction*, Distributed and Parallel Databases, Vol. 7, pp. 7-35

Park, J., S. Ram (2004), *Information Systems Interoperability: What Lies Beneath?*, ACM Transactions on Information Systems, Vol. 22 (4), pp. 595-632

Pedersen, K., J. Emblemsvag, R. Bailey, J.K. Allen, F. Mistree (2000), *Validating Design Methods & Research: The Validation Square*, Proceedings of DETC '00 2000 ASME Design Engineering Technical Conferences

Peffers, K., T. Tuunanen, C.E. Gengler, M. Rossi, W. Hui, V. Virtanen, J. Bragge (2006), *The Design Science Research Process: A Model for Producing and Presenting Information Systems Research*, Design Science Research in Information Systems and Technology (DESRIST)

Ralyte, J., M.A. Jeusfeld, P. Backlund, H. Kühn, N. Arni-Block (2008), *A knowledge-based approach to manage information systems interoperability*, Information Systems, Vol. 33, pp. 754-784

Ram, S., J. Park (2004), *Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts*, IEEE Transactions on Knowledge and Data Engineering, Vol. 16 (2), pp. 189-202

Ram, S., V. Ramesh (1999), *Schema integration: Past, current and future*, Management of Heterogeneous and Autonomous Database Systems, pp. 119–155.

Roberts JR, T.L., M.L. Gibson, K.T. Fields, R. Kelly Rainer JR (1998), *Factors that Impact Implementing a System Development Methodology*, IEEE Transactions on Software Engineering, Vol. 24 (8), pp. 640-649

Rossi, M., M.K. Sein (2003), *Design research workshop: a proactive research approach*, in 26th Information Systems Research Seminar in Scandinavia, The IRIS Association

Rothenberg, J., M. Botterman, C. van Oranje-Nassau (2007), *Towards a Dutch Interoperability Framework*, Recommendations to the Forum Standaardisatie, RAND Europe en GNKS Consult

Shahri, H.H., J. Hendler, D. Perlis (2008), *Grounding the foundations of ontology mapping on the neglected interoperability ambition*, AAAI Spring Symposium - Technical Report SS-08-05, pp. 99-104

Sheth, A., V. Kashyap (1993), *So Far (Schematically) Yet So Near (Semantically)*, IFIP Transactions A: Computer Science and Technology (A-25), pp. 283-312

Shvaiko, P., J. Euzenat (2005), *A Survey of Schema-Based Matching Approaches*, Journal of Data Semantics, Vol. 4, pp. 146-171

Takeda, H., P. Veerkamp, T. Tomiyama, H. Yoshikawam (1990), *Modeling Design Processes*, AI Magazine, pp. 37-48.

Tozer, E. E. (1984), *A Review of Current Data Analysis Techniques*, Data Analysis in Practise Conference Proceeding

Ullberg, J., D. Chen, P. Johnson (2009), *Barriers to Enterprise Interoperability*, Lecture Notes in Business Information Processing, Vol. 38, pp. 13

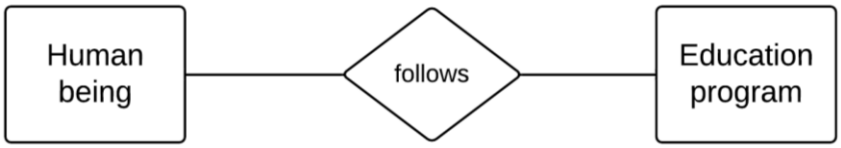
Vernadat, F.B. (1996), *Enterprise Integration: On business process and enterprise activity modeling*, Concurrent Engineering-Research and Applications, Vol. 4 (3), pp. 219-228

Walls, J., G. Widmeyer, O. El Sawy (1992), *Building an Information System Design Theory for Vigilant EIS*, Information Systems Research, Vol. 3 (1), pp 36-59.

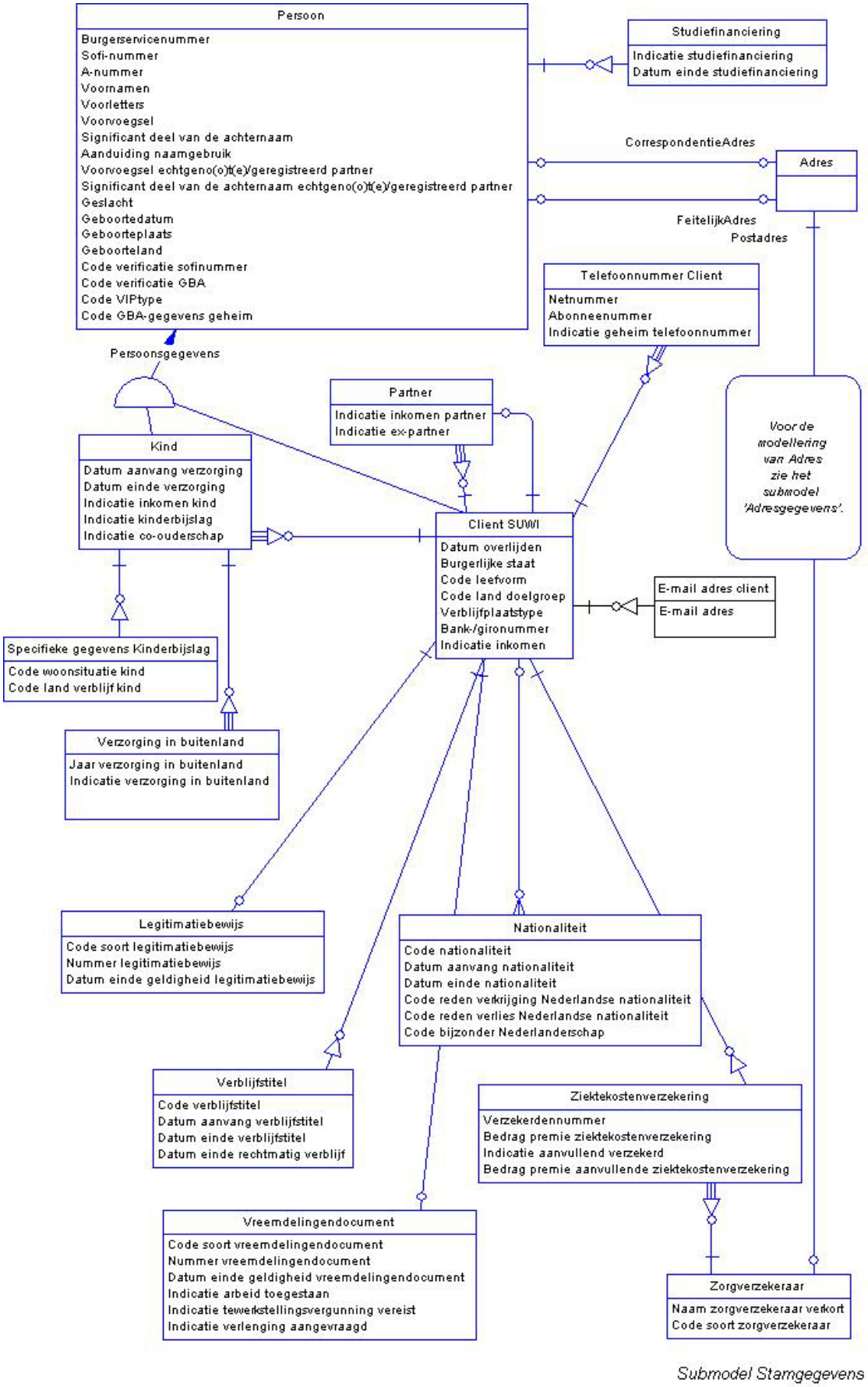
Appendix 1 – Stage 1 Deliverable

Involved Organizations	- Which organizations are involved in the project? - What is the role of each of the participants? - Who will benefit from the integration?
Problem Description	- What is the direct cause of the project? - What is wrong with the current situation?
Goal	- What is the purpose of the project? - When do we label the project as "successful"?
Interoperability Approach & Target Schema(s)	- Which of the three interoperability approaches will be used? - What is the target schema? Create an ER-diagram of the concepts in the target schema.
Priorities	- If the project involves multiple sub-systems, which one has priority? - If more than two organizations are involved, which integration has priority?

Appendix A.1 – Case Study Stage 1 Deliverable

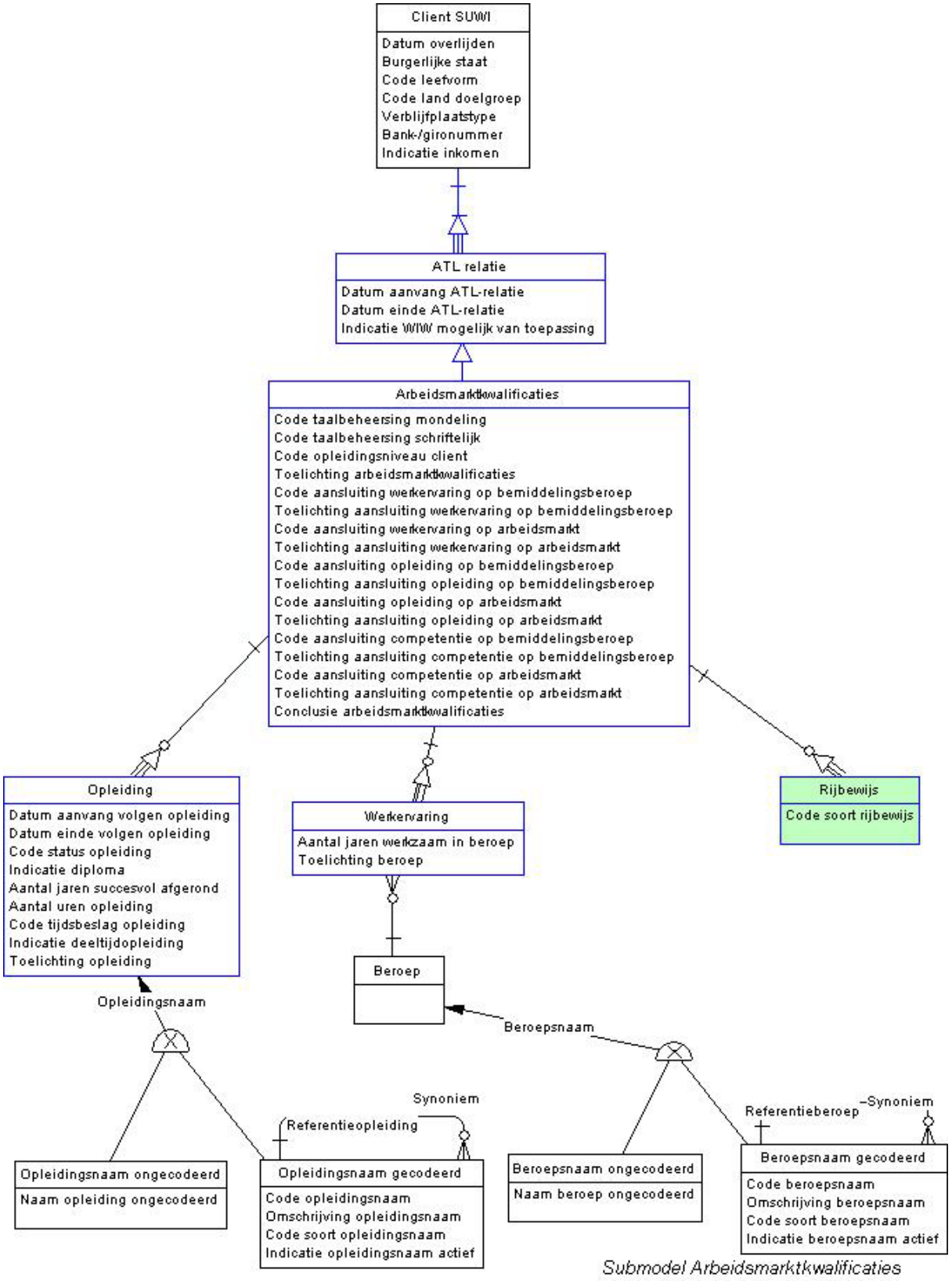
Involved Organizations	The project involves two organizations: DUO (Dienst Uitvoering Onderwijs) and BKWI (Bureau Keteninformatisering Werk & Inkomen). All projected benefits are for BKWI. DUO is necessary for the data integration, but will not benefit from the project directly.
Problem Description	The problem with the current situation is that BKWI does not have access to the information in the DUO system. BKWI wants to know which persons are, or have been, subscribed to education programs so that this information can be used to reintegrate them more effectively into the labor market.
Goal	The goal is to integrate the systems in such a way that BKWI can automatically extract the education history for each of their clients from the DUO system.
Interoperability Approach & Target Schema(s)	We use the federated approach to integrate the systems. The target schema is therefore defined by the concepts the each involved organization wants to receive. In this project only BKWI wants to receive information. The concepts are represented by the following ER-diagram: 
Priorities	Not applicable.

Appendix A.2 – SGR schema (person)



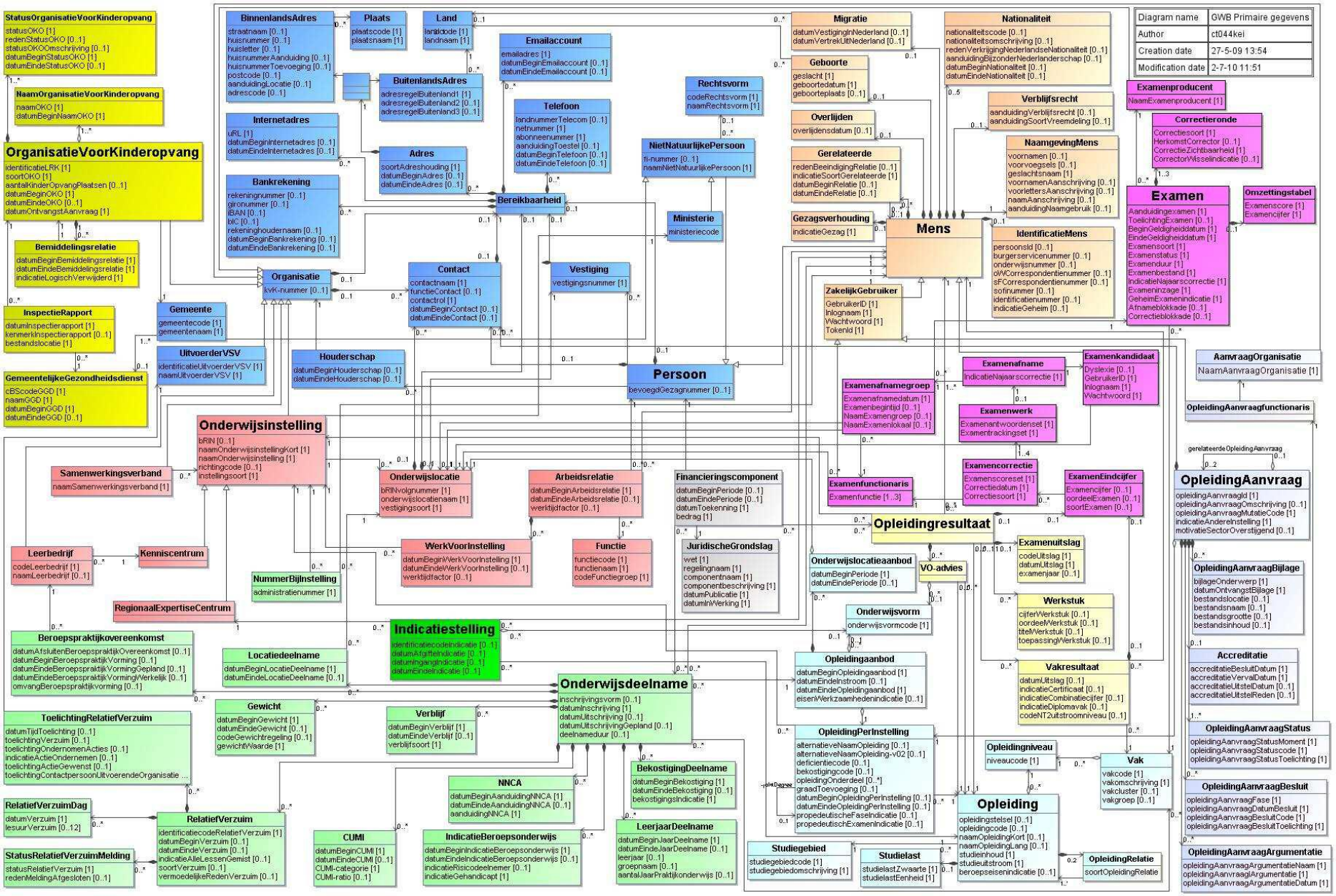
Submodel Stamgegevens

Appendix A.3 – SGR schema (education)

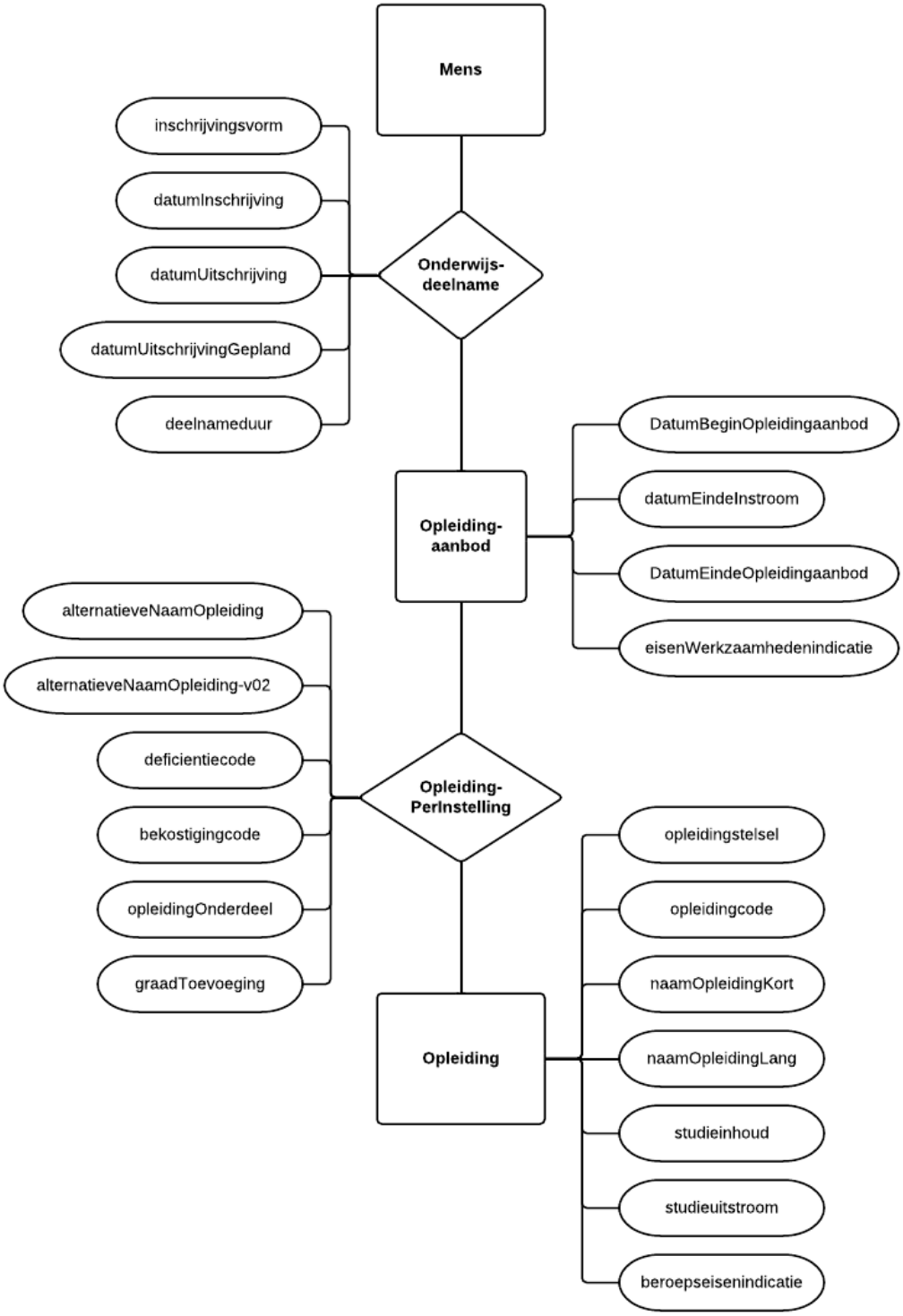


Submodel Arbeidsmarktkwalificaties

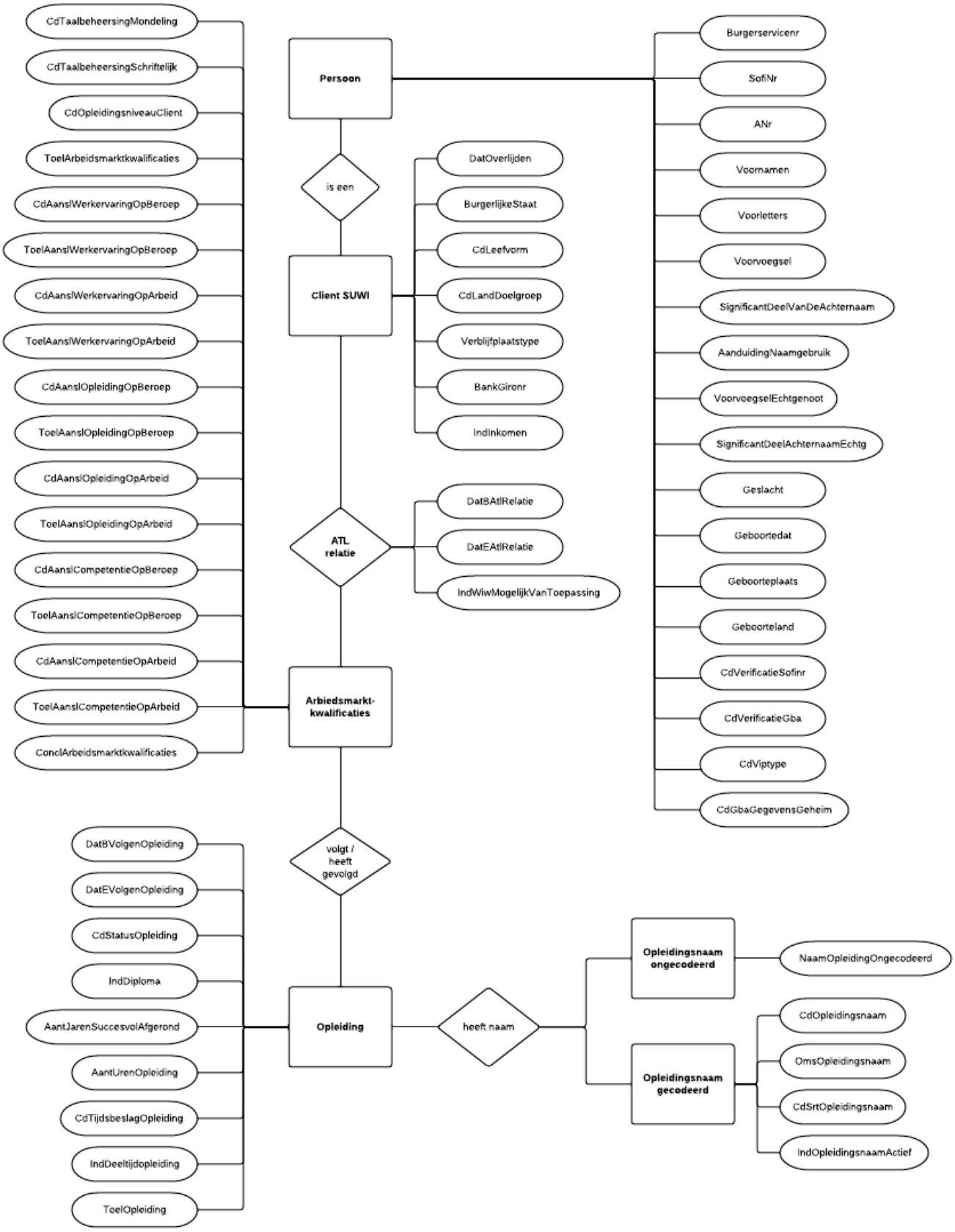
Appendix A.4 – DUO schema



Appendix B.1 – DUO ER diagram (after stage 2a)



Appendix B.2 – SGR ER Diagram



Appendix C – Identifier Comparison

Mapping 1

SGR.Persoon – DUO.Mens

Are the class instances identified by one of the class attributes?

No.

The class DUO.Mens has a relational key to the class DUO.identificatieMens (IdentificationHuman) that stores the identifier for each class instance. We therefore swop the class DUO.Mens with the class DUO.identificatieMens.

SGR.Persoon – DUO.identificatieMens

Are the class instances identified by one of the class attributes?

Yes.

For class SGR.Persoon we have attribute Burgerservicenr as identifier, for class DUO.identificaiteMens we use attribute burgerservicenummer as identifier.

Are the identifiers in both classes derived from the same external domain?

Yes.

Both identifiers use the social security number issued by the government (external domain).

Mapping 2

SGR.OpleidingsnaamGecodeerd – DUO.Opleiding

Are the class instances identified by one of the class attributes?

Yes.

Both classes use their primary keys as the instance identifier.

Are the identifiers in both classes derived from the same external domain?

No.

The primary keys are internally generated.

Do both systems share the same real world entities?

Yes.

And education program offered by one of the Dutch educational institutions can/will exist in both systems.

Can instances be matched by identifier?

No.

The primary keys are unrelated.

Can instances be matched by other attributes?

Yes.

The class SGR.OpleidingsnaamGecodeerd has an attribute CdOpleidingsnaam, and class DUO.Opleiding has an attribute Opleidingscode.

Are the identifiers in both classes derived from the same external domain?
No.
CdOpleidingsnaam is derived from the CWI-domain, while Opleidingscode comes from the OCW-domain.

Do both systems share the same real world entities?
Yes.

Can instances be matched by identifier?
Yes.
If we use a table that links each code in the CWI-domain to the right code in the OCW-domain, we can match the instances of each class.

Mapping 3

SGR.Opleiding – DUO.Onderwijsdeelname

Are the class instances identified by one of the class attributes?
Yes.
Both classes use their primary keys as the instance identifier.

Are the identifiers in both classes derived from the same external domain?
No.
They both use an internally generated unique number of each instance in the class.

Do both systems share the same real world entities?
Yes.
If a person stored in the SGR system mentioned one of the studies he/she followed at one of the Dutch educational institutions, the systems will share the same entities.

Can instances be matched by identifier?
No.
The primary keys are related to nothing and since the two systems are independent from each other, there is now way to find a correlation between the two.

Can instances be matched by other attributes?
No.
There are no attributes present in both classes that would have the same value under any circumstances. It is therefore impossible to match data without human intervention.

Appendix D – Attributes Comparison

Mapping 2

SGR.OpleidingsnaamGecodeerd.OmsOpleidingsnaam

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
Yes.
The attribute NaamOpleidingLang stores similar information.

Is the information stored by both attributes based on the same real-world fundamentals?
No.
The attribute NaamOpleidingLang is based on a table provided by OCW, attribute OmsOpleidingsnaam is based on a table from CWI.

Can you predict the values of an attribute in the other class by using some function or expert knowledge?
Yes.
If we create a conversion table that links each value from the table provided by OCW to the appropriate value in the table by CWI, we can predict the value.

SGR.OpleidingsnaamGecodeerd.CdSrtOpleidingsnaam

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
The attribute describes an internally created aspect of each instance, and can therefore not be found in the other system.

Can you predict the values of an attribute in the other class by using some function or expert knowledge?
No.
The value of this attribute has to be internally generated.

SGR.OpleidingsnaamGecodeerd.IndOpleidingsnaamActief

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
The attribute involves a reference to a table from a foreign domain. Since the other system is not familiar with this foreign domain, we cannot find a related attribute.

Can you predict the values of an attribute in the other class by using some function or expert knowledge?
No.
The value of this attribute has to be internally generated.

Mapping 3

SGR.Opleiding.DatBVolgenOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
Yes.

The attribute datumInschrijving provides similar information.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both attributes store the date at which the person formally starts the education.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
This is a one-to-one relation.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
No.
The attributes describe exactly the same concept.

SGR.Opleiding.DatEVolgenOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
Yes.
The attribute datumUitschrijving provides similar information.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both attributes store the date at which the person formally end the education.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
This is a one-to-one relation.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
No.
The attributes describe exactly the same concept.

SGR.Opleiding.CdStatusOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
No.
The information is not defined in the other system.

Can you predict the values of an attribute in the other class by using some function or expert knowledge?
Yes.
By comparing the internal values for the two attributes 'DatEVolgenOpleiding' and 'IndDiploma', we can calculate the value of this attribute.

SGR.Opleiding.IndDiploma

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.
The class DUO.Examenuitslag stores the same kind of information in its attribute 'codeUitslag'.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both are based on the same real-world event: whether the person graduated for the education.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
Although the class DUO.Examenuitslag consists of three attributes, the concept whether the person received a diploma, is stored by just one attribute.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
Yes.
The class DUO.Examenuitslag stores the result of the exam that was taken, while class SGR.IndDiploma only stores whether a diploma has been received.

SGR.Opleiding.AantJarenSuccesvolAfgerond

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.
The class DUO.LeerjaarDeelname stores the same kind of information in its attribute 'leerjaar'.

Is the information stored by both attributes based on the same real-world fundamentals?

Yes.
Bot attributes are based on the number of years the person has advanced in the study program.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
The concept is described by only one attribute in both systems.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
No.
They describe exactly the same concept.

SGR.Opleiding.AantUrenOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.
The class DUO.Studielast stores the same kind of information.

Is the information stored by both attributes based on the same real-world fundamentals?
No.
DUO.Studielast stores the effort the whole education program requires, while SGR.AantUrenOpleiding indicates the average time a student participates in the program per week.

Can you predict the values of an attribute in the other class by using some function or expert knowledge?
No.
If we want to convert the total time effort of the study program, to the average time per week, we need to know the length of the program. Since we do not have this information, we cannot predict the value of the attribute.

SGR.Opleiding.CdTijdsbeslagOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.

The class DUO.Onderwijsvorm stores the same kind of information in its attribute 'onderwijsvormcode'.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both attributes are based on the form the education program is presented in.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
In both systems this is defined by one single attribute.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
No.
They describe the same concept.

SGR.Opleiding.IndDeeltijdOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.
The class DUO.Onderwijsvorm stores the same kind of information in its attribute 'onderwijsvormcode'.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both attributes are based on the form the education program is presented in.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
In both systems this is defined by one single attribute.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
Yes.
The class IndDeeltijdOpleiding only stores whether the education program is followed part-time, while DUO.Onderwijsvorm stores the form the education is presented in (can be part-time, or any other form).

SGR.Opleiding.ToelOpleiding

Can you find (an) attribute(s) in the other class that store(s) semantically related information?
No.
There is no such attribute in the class DUO.Onderwijsdeelname.

Is the information stored by the attribute defined as a separate entity in the other database?
Yes.
The class DUO.Opleiding stores the same kind of information in its attribute 'studieinhoud'.

Is the information stored by both attributes based on the same real-world fundamentals?
Yes.
Both provide a description of the content of the education.

Is the information stored by the attribute defined by more than one attribute in the other class?
No.
In both systems this is defined by one single attribute.

Is the information stored by the attribute about a more specific concept than the information stored by the attribute in the other class?
No.
Both cover the concept at the same level.

Appendix E – Data Format Comparison

Mapping 1

M1.1: Burgerservicenr - burgerservicenummer

M1.1	(SGR) Burgerservicenr	(DUO) burgerservicenummer	Conflict
Format	Numeric	String	Data representation
Characters	9	9	
Notation	123456789	123456789	
Range	Defined by formula*	Defined by formula*	
Default value	-	-	
Scale	-	-	

* Both must satisfy the following function:

MOD ((9*p1 + 8*p2 + 7*p3 + 6*p4 + 5*p5 + 4*p6 + 3*p7 + 2*p8 – p9) : 11) = 0

Mapping 2

M2.1: CdOpleidingsnaam - Opleidingcode

M2.1	(SGR) CdOpleidingsnaam	(DUO) Opleidingscode	Conflict
Format	Numeric	String	Data representation
Characters	10	8	Integrity constraint
Notation	1234567890	12345678	
Range	Defined by table*	Defined by table**	
Default value	-	-	
Scale	-	-	

* Defined by the CWI-domain

** Defined by the OCW -domain

M2.2: OmsOpleidingsnaam - NaamOpleidingLang

M2.2	(SGR) OmsOpleidingsnaam	(DUO) NaamOpleidingLang	Conflict
Format	String	String	
Characters	120	225	Integrity constraint
Notation	text	text	
Range	-	-	
Default value	-	-	
Scale	-	-	

Mapping 3

M3.2: DatBVolgenOpleiding - datumInschrijving

M3.2	(SGR) DatBVolgenOpleiding	(DUO) datumInschrijving	Conflict
Format	Numeric	String	Data representation

Characters	8	10	Data representation
Notation	YYYYMMDD	YYYY-MM-DD	Data representation
Range	-	-	
Default value	-	-	
Scale	Gregorian calendar	Gregorian calendar	

M3.3: DatEVolgenOpleiding - datumUitschrijving

M3.3	(SGR) DatEVolgenOpleiding	(DUO) datumUitschrijving	Conflict
Format	Numeric	String	Data representation
Characters	8	10	Data representation
Notation	YYYMMDD	YYYY-MM-DD	Data representation
Range	-	-	
Default value	-	-	
Scale	Gregorian calendar	Gregorian calendar	

M3.4: IndDiploma - codeUitslag

M3.4	(SGR) IndDiploma	(DUO) codeUitslag	Conflict
Format	Numeric	String	Data representation
Characters	1	1	
Notation	X	X	
Range	0,1,2,8*	G,A,T,E,D**	Integrity constraint
Default value	-	-	
Scale			

*IndDiploma is bound to four values:
0: Onbekend
1: Ja
2: Nee
8: Niet van toepassing

**codeUitslag is bound to five values:
G: Geslaagd
A: Afgewezen
T: Teruggetrokken
E: Gespreid examen
D: Certificaat

M3.5: AantJarenSuccesvolAfgerond - leerjaar

M3.5	(SGR) AantJarenSuccesvolAfgerond	(DUO) leerjaar	Conflict
Format	Numeric	String	Data representation

Characters	2	1	Integrity constraint
Notation	XX	X	
Range	0-99	0-9	Integrity constraint
Default value	-	-	
Scale	-	-	

M3.6: CdTijdsbeslagOpleiding - Onderwijsvormcode

M3.6	(SGR) CdTijdsbeslagOpleiding	(DUO) Onderwijsvormcode	Conflict
Format	Numeric	String	Data representation
Characters	1	4	Integrity constraint
Notation	X	XXXX	
Range	0-5*	VT,DT,DU,SC,AB,LWOO, LWT,BOL,BBL,VAVO**	Integrity constraint
Default value	-	-	
Scale	-	-	

* *CdTijdsbeslagOpleiding* can be one of six values:

0: Onbekend
1: Dagopleiding
2: Avondopleiding
3: Dag- + avondopleiding
4: Schriftelijk
5: Anders

** *Onderwijsvormcode* can be one of ten values:

VT: Voltijd
DT: Deeltijd
DU: Duaal
SC: Schriftelijk
AB: Ambulante begeleiding
LWOO: Leerwegondersteunend onderwijs
LWT: Leerwerktraject
BOL: Beroepsopleidende leerweg
BBL: Beroepsbegeleidende leerweg
VAVO: Voortgezet Algemeen Volwassenen Onderwijs

M3.7: IndDeeltijdopleiding - Onderwijsvormcode

M3.7	(SGR) IndDeeltijdopleiding	(DUO) Onderwijsvormcode	Conflict
Format	Numeric	String	Data representation
Characters	1	4	Integrity constraint
Notation	X	XXXX	
Range	1-2*	VT,DT,DU,SC,AB,LWOO, LWT,BOL,BBL,VAVO**	Integrity constraint
Default value	-	-	

Scale	-	-

* *IndDeeltijdOpleiding* can be one of two values:

1: Ja

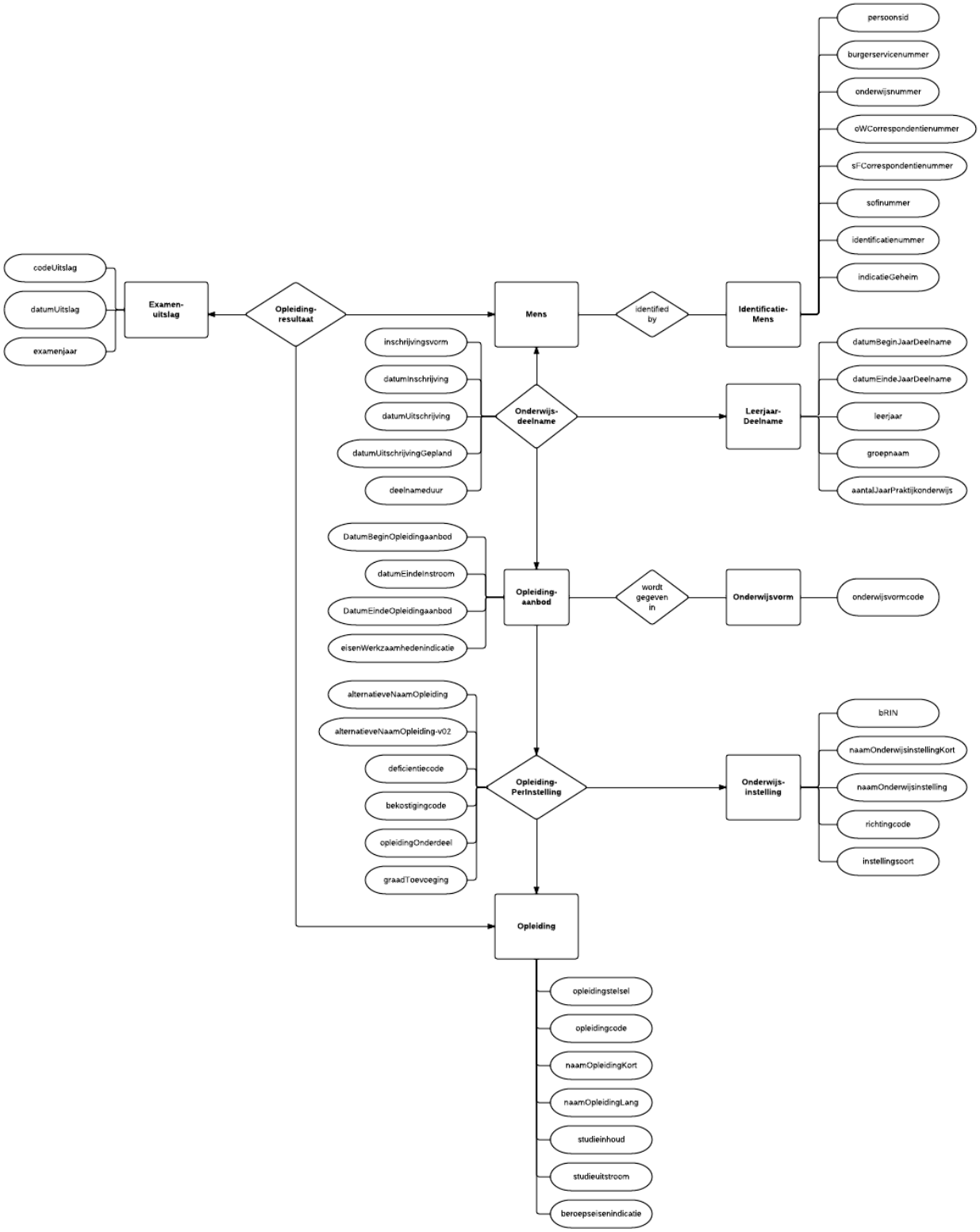
2: Nee

** See table M3.6

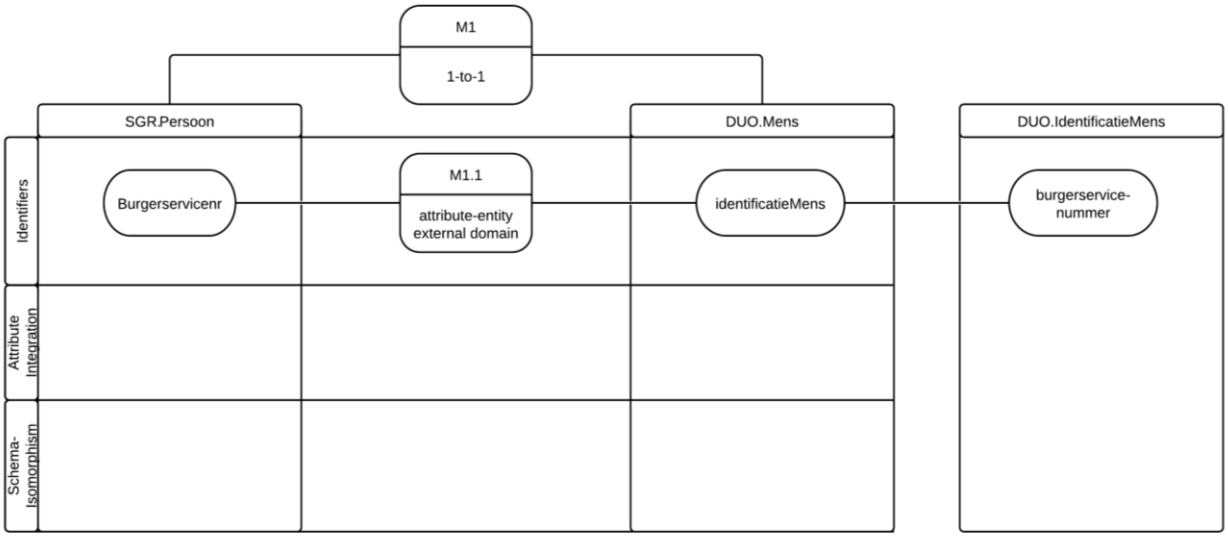
M3.8: ToelOpleiding - studieinhoud

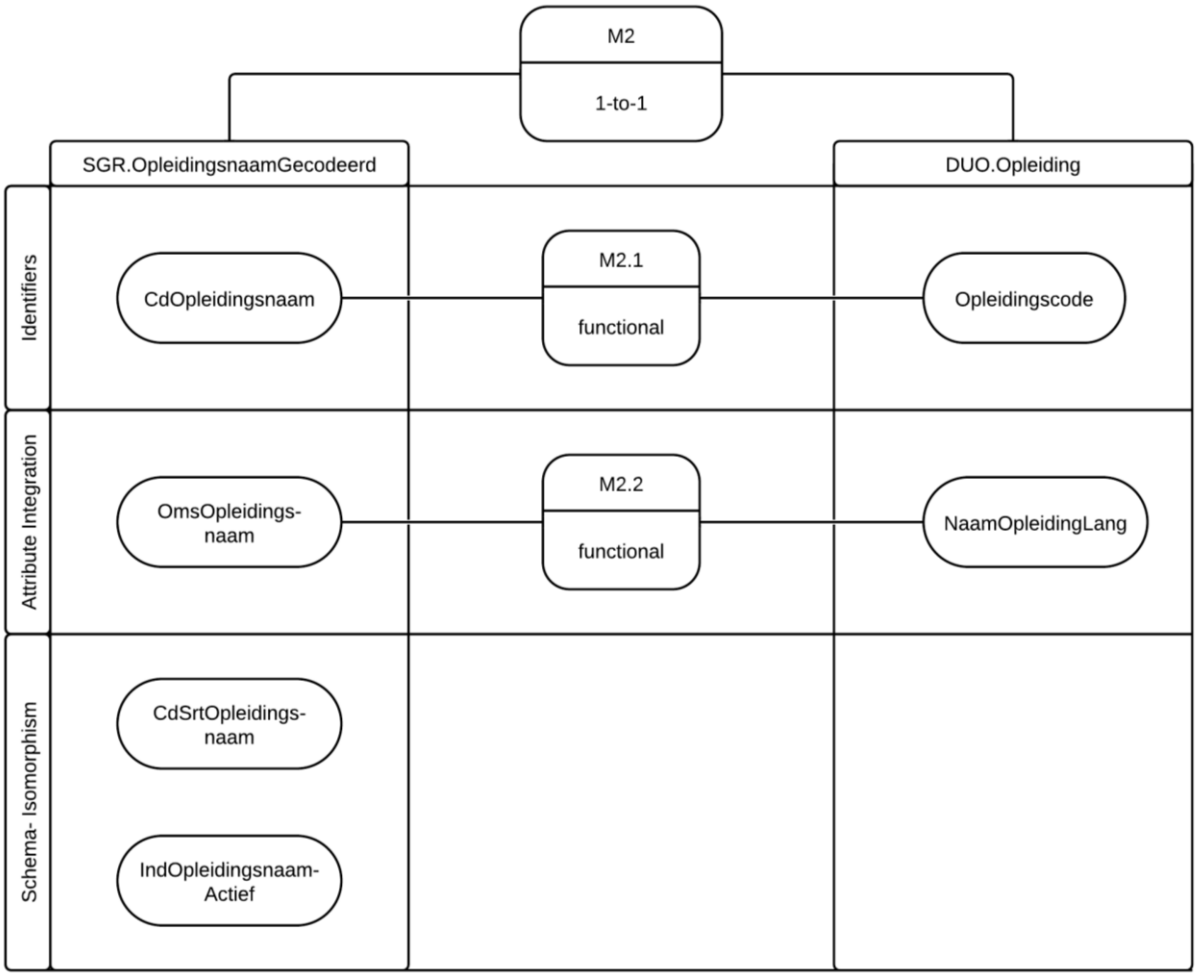
M3.8	(SGR) ToelOpleiding	(DUO) studieinhoud	Conflict
Format	String	String	
Characters	180	150	Integrity constraint
Notation	Text	Text	
Range	-	-	
Default value	-	-	
Scale	-	-	

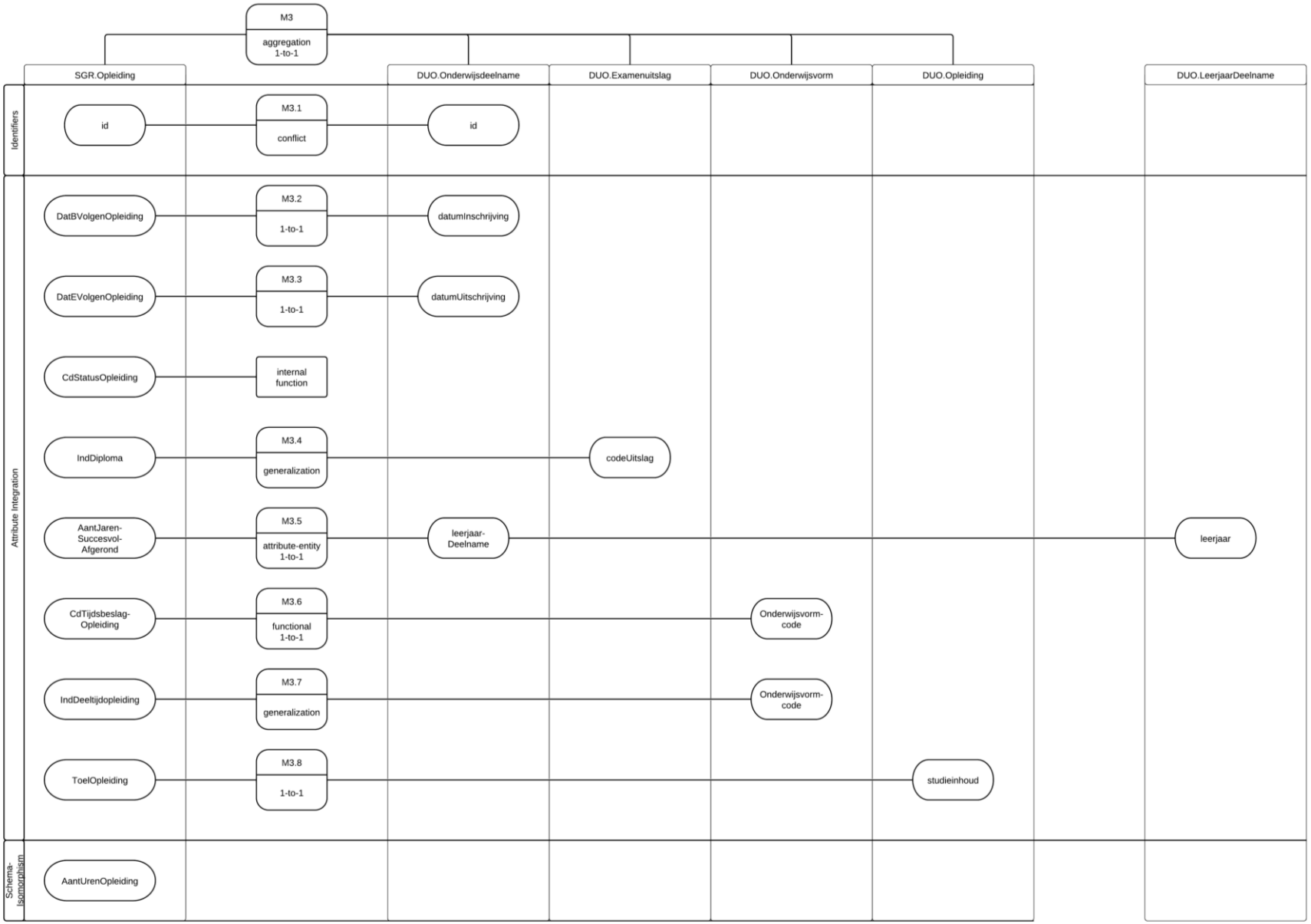
Appendix F – DUO ER Diagram (after stage 3d)



Appendix G – Case Study Mapping Representation






M3
aggregation
1-to-1
SGR.Opleiding
DUO.Onderwijsdeelname
DUO.Examenuitslag
DUO.Onderwijsvorm
DUO.Opleiding
DUO.LeerjaarDeelname
Identifiers
Attribute Integration
Schema-Isomorphism
id
M3.1
conflict
id
DatBVolgenOpleiding
M3.2
1-to-1
datumInschrijving
DatEVolgenOpleiding
M3.3
1-to-1
datumUitschrijving
CdStatusOpleiding
internal function
IndDiploma
M3.4
generalization
codeUitslag
AantJaren-Succesvol-Afgerond
M3.5
attribute-entity
1-to-1
leerjaar-Deelname
leerjaar
CdTijdsbeslag-Opleiding
M3.6
functional
1-to-1
Onderwijsvorm-code
IndDeeltijdopleiding
M3.7
generalization
Onderwijsvorm-code
ToelOpleiding
M3.8
1-to-1
studieinhoud
AantUrenOpleiding