Using DEMO as a Methodology to Specify the Semantics of Data Message

Agustina Linda Setiawati

Using DEMO as a Methodology to Specify the Semantics of Data Message

MASTER THESIS

submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

of master program

INFORMATION ARCHITECTURE

by

Agustina Linda Setiawati

Information Architecture
Faculty of Electrical Engineering,
Mathematics, and Computer Science
Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

TNO ICT
Brassersplein 2
Delft, the Netherlands
www.tno.nl

Using DEMO as a Methodology to Specify the Semantics of Data Message

Author:	Agustina Linda Setiawati
Student id:	1333593
Email:	agustina.linda@gmail.com

Abstract:

Effective communication is required for a successful service execution. It can be achived when both the service consumer and service provider understand the exchanged data message in the same way. Therefore, the specification of the semantics of data message needs to be explicitly defined in the service specification. Six requirements for specifying the semantics of data message are formulated based on the data modeling approach. DEMO is selected to be the methodology for representing the semantics of data message. Since one of the requirements is not completely fulfilled by DEMO; DEMO is extended by using ORM. Finally, the specification of the semantics of the data message of a case study is implemented by using DEMO. It indicates the DEMO usability for specifying the semantics of data messages.

Keywords: *semantics of data message*, *conceptual schema*, *DEMO*, *ORM*, *service specification*.

Acknowledgement

Bismillahirrahmanirrahim. Thank to Allah SWT, the Almighty God who made all things possible.

The preparation of this thesis document would not have been possible without the support of those around me. I would like to take this opportunity with great pleasure to express my gratitude to them.

I really appreciate TNO for giving me the chance to conduct my thesis there; it was a nice experience working there. I would like to thank for the facilities TNO has provided, including the permission of using their knowledge and tools. Especially, I would like to express gratitude to Wout and Paul that have given so much support and suggestions when I was working in TNO.

I would also express my appreciation to Prof. Dietz. Aside from being the chairman of the examination commitee, he also taught me on several occassions. Some of the course he taught is very important for this thesis research. My sincere thanks also to Dr. Albani who spent her time reviewing and giving suggestions during the time I worked on my thesis.

Some friends have also supported me in this tough time. I want to thank them for all their help. I am obliged to Puspa and Yan for giving me so many suggestions and support, as well as to Vikram for getting through this difficult time together. Gde has been so kindly providing the explanation of his thesis which is useful for this thesis. I feel grateful as well to Anggara and Baqir who have been very generous and helpful, Saputra and Pita for their support and help in finding an apartment, also to my ex-neighbors Mas Zaki and Mbak Selly, and to all my classmates, Indonesian friends, badminton friends, as well as office mates.

Even though we are separated in distance, my parents, brothers and sisters have also given me huge supports. They always encourage and motivate me. They keep reminding me that besides the effort, I must not forget to pray. I am really grateful for all things they have done for me.

It is hard to get through anything without the support of those closest to you. At last but not least, I would like to give my special thanks to my husband Habib who has provided his vital encouragement and support. Things would have been more difficult without him.

Agustina Linda Setiawati
Delft, the Netherlands
2010

Contents

List of Figures

List of Tables

List of Action Rules

Chapter 1
Introduction

Effective communication is an important aspect required for a successful service execution. It can be achieved when both the service consumer and service provider involved in the service invocation understand the exchanged data message in the same way. In this master thesis project, the investigation on data message will be performed. The project is conducted in cooperation with TNO ICT Delft.

1.1 Background

Nowadays modern civilization is characterized by complex systems and institutions that are connected with each other. Information and Communication Technology (ICT) becomes an important aspect to support their activities. The increasing of computing facilities at a very rapid pace, has led to the development of services [Ws-DIAMOND, 2007]. Services facilitate the provision of supports for systems that are connected over the network. In the Service Oriented Architecture (SOA) domain, a service is defined as an exposed piece of functionality, which is self-contained, autonomous, platform independent and can dynamically be located, invoked and (re-)combined [Papazoglou and Heuvel, 2007]. It means that as long as a system knows the interface of services, even if it uses incompatible system technologies, the system can still exploit the functionalities of services without concern how the services are implemented. This indicates that services are geared toward *loosely coupled* systems.

Two parties are involved during a service invocation, i.e., the *service consumer* and *service provider*. The party that requires the functionality of a service is called the service consumer; while the other party that provides the functionality of a service is called the service provider. A service consumer initiates a service invocation by sending a request message to the service provider. Subsequently, a service provider can respond to the request of the service consumer by executing the service function and sending the result to the service consumer. An invocation of a service is an execution of an activity which has some predefined input and output [Ws-DIAMOND, 2007]. This implies that a service invocation involves passing messages between the service provider and service consumer, in which the specification of the messages has been predefined as the input and output of a service.

In a service invocation, successful service execution is not always achieved. Faults can occur during a service execution wherein the semantic data errors become one of the reasons, such as incorrect input, incorrect produced output, etc [Ws-DIAMOND, 2007].

Incorrect input indicates that the data message sent from the service consumer to the service provider is not suitable for the service execution. While incorrect output indicates that the result of the service execution from the service provider is not satisfying the requirements of the service consumer; incorrect output can also be caused by the incorrect input. Semantic data errors such as incorrect input and output may be result of different interpretation between the service consumer and service provider. The lack of semantic information becomes one of the possible reasons. As mentioned by Linda Terlouw[1], if provider and consumer have different expectations of the behavior of a service, they are bound to get problems. This implies that the information about the input and output of services, especially the semantics of input and output, needs to be specified.

The information about the input and output of a service is provided in *service specification*. A service specification is formulated by the service provider and registered in a service registry in which service consumers can browse to find the required services. As described by Hardjosumarto, a service specification contains a complete, unequivocal, and precise description of the external view of a service (component) and describes the conditions under which a service is provided [Hardjosumarto, 2008]. Hardjosumarto himself has identified a framework for defining service specifications called the Generic Service Specification Framework. It consists of the following five areas of concern (the detailed information on the areas of concern of the framework can be found in Appendix D):

1. *Service Provider Information*
2. *Service Contract Information*
3. *Service Function Information*
4. *Service Usage Information*
5. *Terminology*

The Service Function Information comprises information that clarifies what kind of service is provided; the *Input* and *Output* required for service invocation are classified in the *Function Description* of the Service Function Information.

It is important that both the service consumer and service provider have the same interpretation or understanding about the services defined in service specifications [Hardjosumarto, 2008]. To ensure that the service provider and service consumer have the same interpretation or understanding on what is specified in the service specification, the service specification should be specified in an unambiguous and clear way which is semantically understandable [Hardjosumarto, 2008]. Hardjosumarto's Generic Service Specification Framework accommodates the necessity for the semantics specification of a service by providing the *Terminology* area of concern. Particularly, the semantics of the input and output of a service is specified in the Terminology for the Service Function Information. Thus, it is confirmed that the information about the input and output of a service, as well as the semantics information of the input and output of a service, is explicitly defined in the service specification

By specifying the semantics of the input and output of a service in service specification, it is expected that the service consumer will have the same interpretation with the service provider about the input and output of a service. Thus, both actors will specify the appropriate input and output so a successful service execution can be accomplished. Therefore, having semantics understanding about the input and output of a service is considered able to bring about *effective communication*, which is required for a successful service execution.

[1] http://www.servicespecification.com/?p=135, accessed: 15 September 2009

In effective communication there is a correlation between what the sender thinking about and what the receiver thinking about [Winbow, 2002]. In order to achieve this, the involved actors need to communicate in the same language, in which the selected notation should be understandable and unambiguous, so confusion and error can be avoided [Winbow, 2002]. The used language should convey only one meaning so that the opportunities for miscommunication are reduced to the lowest level possible. As stated by ter Bekke, differing communications lead to differing interpretations of the same thing [ter Bekke, 1992]. It is considered that a methodology to specify the semantics of the input and output of a service in a service specification is required. The methodology needs to support effective communication, so an understandable and unambiguous specification of the semantics of input and output can be produced.

The DEMO methodology[2] developed by Prof.dr.ir. J.L.G. Dietz is a methodology for the design, engineering, and implementation of organization and network of organizations; such a construction is known as *enterprise ontology* [Dietz, 2006]. An enterprise ontology provides all essential information that is necessary for the design of the supporting information systems, and at a level of abstraction that makes it also understandable for business people [Albani and Dietz, 2008]. Therefore, it is assumed that DEMO may be considered as a methodology that supports effective communication. The problem is we do not know whether DEMO methodology can be used to specify the semantic of the input and output of a service, neither how to use DEMO methodology for that purpose. In addition, we do not know what aspects are required to be specified for representing the semantics of the input and output of a service. Therefore, there is a need to further investigate the DEMO methodology in specifying the semantics of the input and output of a service.

1.2 Research Goal and Research Questions

An exchanged message in a service invocation consists of the *command* and the *data message*. A command indicates the identity of a message. As in a service invocation, a command represents the name of the service function. For instance, to invoke a service that provides the information about the flight availability, the "`getFlight`" represents a command. While, the actual values exchanged between the actors represent the data messages. For invoking the "`getFlight`", the data message will refer to, for instance, the departure location, departure date, destination, and arrival date, which are identified as the "`from`", "`departureDate`", "`destination`", and "`arrivalDate`" respectively, sent from the service consumer to the service provider. In addition, the execution result of the "`getFlight`", named, e.g., "`availableFlight`" sent from the service provider to the service consumer, can also be regarded as data message. It appears that the specification of the data messages of a service invocation refers to the input and output of a service. As exhibited in the background, this master thesis project concerns about the input and output of a service. Since the input and output of a service can be considered as data messages, this terminology will be used further on. Thus, the data messages specification represents the parameters of a service that consists of the input and output of a service, which is defined in the service specification.

As identified above, regarding the assumption on DEMO, several problems emerge. Therefore, it is necessary to further investigate the possibility of using DEMO for specifying the semantics of data message. Research on specifying the semantics of data

[2] DEMO 2.0 [Dietz, 2006]

message by using the DEMO methodology becomes the focus of this master thesis project. The research goal of this master thesis project is formulated as follow:

To confirm and demonstrate DEMO's ability in specifying the semantics of data message by complying with the requirements for specifying the semantics of data message

By referring to the main goal as stated above, this master thesis project consists of the following research questions:

1. *What requirements are formulated for specifying the semantics of data message based on the data modelling approach?*
   This question is answered by analysing the aspects of data modelling that are related to the semantics of data. From the identified aspects, the requirements are formulated by identifying the essential features available in the aspects.

2. *How DEMO fulfil all of the requirements? Are there any modifications required to be performed to DEMO?*
   This question is answered by performing theoretical study to DEMO methodology. In particular, the aspect models of DEMO methodology will be examined against the requirements. In case some requirement is not fully satisfied, DEMO will be combined or extended with other methodology.

3. *How can DEMO be used for specifying the semantics of data message?*
   In order to answer this question, the usability of DEMO in specifying the semantics of data message needs to be exhibited. Thus, DEMO will be used to specify of the semantics of the data message of a case study by complying with the formulated requirements.

According to the aforementioned research questions, as the starting point of the investigation, we need to figure out the aspects or features required for specifying the semantics of data message. In order to capture the semantics of data, we need to go beyond keywords, and specify the meaning of the described resources (data interpretations) [Smith et al., 2004]. According to ter Bekke, the meaning of information lies in the connection between data [ter Bekke, 1992]. This implies that in order to obtain the semantics of a certain datum, we need to know the relationships between the datum and the other data. In other words, we need to find out how data are structured. Therefore, it is considered that "*data modeling*" is a suitable approach as the theoretical foundation for identifying the aspects required for specifying the semantics of data message. Further on these aspects are called the "*Requirements for specifying the semantics of data message*".

After the requirements have been identified, the ability of DEMO in fulfilling the requirements will be investigated. In case DEMO is unable to satisfy the whole requirements, some modifications might be needed to be applied to DEMO, such as by combining or extending DEMO with another methodology.

At the end of this master thesis project, to figure out how to use DEMO to specify the semantics of data message, as a demonstration, the semantics of the data messages of a case study is specified. The semantics will be specified in DEMO by complying with the identified requirements.

Overall, three outcomes are provided in this master thesis project. It consists of the list of the requirements for specifying the semantics of data message, the specification of the DEMO methodology in fulfilling the requirements, and the specification of the semantics of the data messages of a case study specified in DEMO.

1.3 Master Thesis Project Phase

The phases and the expected outcomes of the master thesis project are exhibited in Figure 1-1. As seen in the figure, three outputs are expected from this master thesis project, which are denoted by the document symbol. In order to accomplish the research goal, the process is divided into several phases, in which each phase will be specified in one chapter. The master thesis project phases are explained below.

1. Formulate the problems and goal
   In this phase, the background, research goal, and research questions of this master thesis project are identified. They can be found in Chapter 1.

2. Analyze data modelling
   Data modelling is selected as the starting point to investigate the semantics of data. The aspects of data modelling that have strong relationship with data semantics will be identified. The identified aspects will be used as the basis to define the requirements for specifying the semantics of data message. The analysis to data modelling can be found in Chapter 2.

3. Formulate the requirements for specifying the semantics of data message
   Based on the identified aspects of data modelling, the requirements for specifying the semantics of data message are formulated. The requirements are specified by analyzing the essential features available in the data modelling aspects. The requirements for specifying the semantics of data message are exhibited in Chapter 3. It represents the first outcome of this master thesis project

4. Analyze DEMO in fulfilling the requirements
   The ability of the DEMO methodology in fulfilling each requirement will be investigated thoroughly. The analysis of the DEMO methodology is performed in Chapter 4. It represents the second result of this master thesis project

5. Identify the modification on DEMO
   Based on the result produced from the previous phase, we can find out whether all of the requirements for specifying the semantics of data message can be completely satisfied by DEMO or not. Apparently not all of the requirements are fully satisfied, thus, another methodology is used to extend DEMO. As explained in Chapter 5, the ORM is selected for this purpose. The additional specification as derived from the ORM completes the second result of this master thesis project.

6. Use DEMO for specifying the semantics of data message
   The usability of the DEMO methodology for specifying the semantics of data message will be exhibited. In this phase, the DEMO methodology is used to specify the semantics of the data message of a study case by complying with the formulated requirements. The specification of the semantics of the data message of the case study is exhibited in Chapter 6. It represents the third result of this master thesis project.

7. Conclude the result
   In this phase, the outcomes achieved from the research will be summarized. Some conclusions are provided. In addition, some recommendations for future work are specified as well. The conclusions and recommendations of the research can be found in Chapter 7.

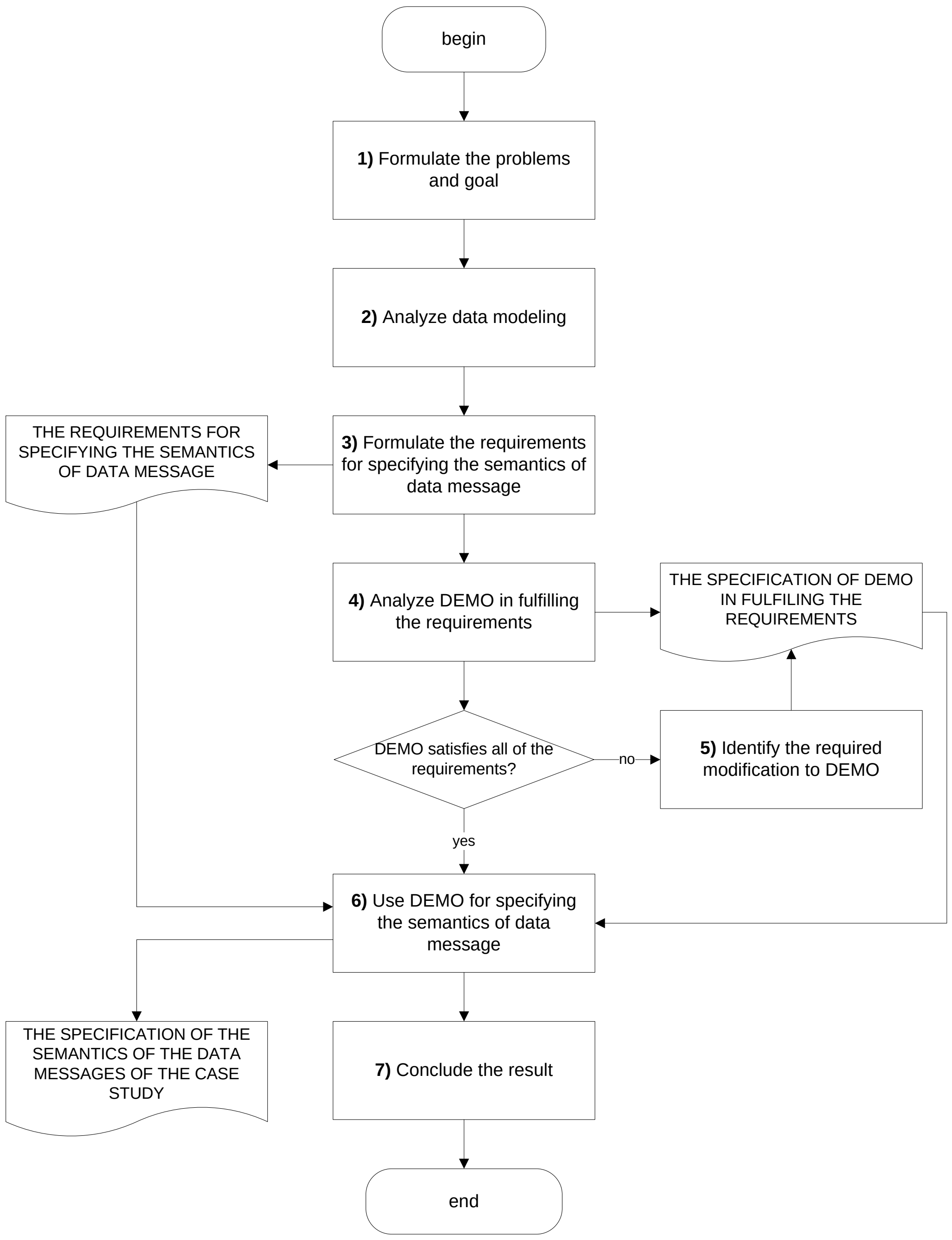

begin
1) Formulate the problems and goal
2) Analyze data modeling
3) Formulate the requirements for specifying the semantics of data message
THE REQUIREMENTS FOR SPECIFYING THE SEMANTICS OF DATA MESSAGE
4) Analyze DEMO in fulfilling the requirements
THE SPECIFICATION OF DEMO IN FULFILING THE REQUIREMENTS
DEMO satisfies all of the requirements?
no
5) Identify the required modification to DEMO
yes
6) Use DEMO for specifying the semantics of data message
THE SPECIFICATION OF THE SEMANTICS OF THE DATA MESSAGES OF THE CASE STUDY
7) Conclude the result
end

Figure 1-1 Master thesis project phases

Chapter 2
Data Modeling

Data is an important resource for every organization. It can be more valuable and useful if we can obtain some meaningful information out of it. Information represents a new structure that relates data with other data; the semantics exist in the relation between the data [ter Bekke, 1992]. In order to model data and create valuable information, a certain engineering process is required. In this chapter, data modeling will be explained. Data modeling has been selected as the starting point for investigating the semantics of data.

The definition of data model is explained in section 2.1. After that, the engineering process of data model, which is known as data modeling will be described in section 2.2. The relationships of data modeling and the semantics of data will be explained in section 2.3. Requirement analysis will be briefly introduced in section 2.4. A conclusion is provided in section 2.5.

2.1 Data Model

Dietz explained that investigating systems mostly comes down to building models and analyzing the behavior of these models. In general, a *model* can be considered as a representation, abstraction, prototype, or interpretation of a system [Dietz, 2006]. For an application or an information system involving data, a data model is required. For instance in the development of an information system that use database, data model can be considered representing the abstraction of the database. As stated by ter Bekke, data models are useful in managing properties of data processing applications [ter Bekke, 1992].

Meersman et al. explained that a data model represents the structure and integrity of the data elements of the, in principle "single", specific enterprise application(s) by which it will be used [Meersman, 2001]. This indicates that building data models of an enterprise (or a system in general) depends on the specific needs and tasks that are performed within this enterprise.

A model depends a lot on the designer interpretation about the system. Therefore, different models may be produced, which often clearly dependent on the design method selected [ter Bekke, 1992]. Such condition is called *view independence*, which indicates that the outcome of data modeling project is independent of the chosen development trajectory [ter Bekke, 1992]. Thus, for a system, the produced data models are not unique.

According to ter Bekke, a data model is characterized by [ter Bekke, 1992]:
- A collection of data structure types (the building blocks)
- A collection of operators or rules of inference, which can be applied to valid instances of the data structure types
- A collection of general integrity rules, which implicitly or explicitly define the set of consistent states or changes of states or both

It can be concluded that the applicability of a data model is fully determined by the presence the corresponding components: *structures*, *operations*, and *constraints* [ter Bekke, 1992].

2.2 Data Model Engineering

The engineering process of creating a data model is known as *data modeling*. It is a complex process involving various matters performed to data. Data modeling is a critical skill for IT professionals including: database administrators (DBAs), data modelers, business analysts and software developers [Infogoal, 2009]. Thus, it can be considered as an essential ingredient of nearly all IT projects. As long as there are data and information involved in a (complex) system, especially when the entire focus of the process is placed on data and their properties, data modeling should be performed. Since this master thesis project has a strong relationship with data, data modeling is considered suitable as the starting point for investigating the semantics of data.

Data modeling is an integrated process of arranging, collecting, and differentiating of data into workable units, mutually related by user requirements [ter Bekke, 1992]. ter Bekke mentioned that data modeling consists of a set of technique and auxiliaries, which can be applied consistently to various data (base) development projects. Overall, data modeling helps designer to define the relationships between data elements and their structures, which are required for data processing applications. The interest of data modeling is the provision of a model which can be used successfully in data processing [ter Bekke, 1992]. A data model is commonly represented in a form of schema.

In order to arrive at a useful and meaningful model, one requires taking into account only the relevant parts of the system. Consequently, the irrelevant details are discarded. The purpose of performing this act is to get an insight of the system, which is called the *abstraction* of system. Ter Bekke described abstraction as the omission of all object details, excluding the characteristics relevant to the information needs. Abstractions to be included in a model (has to be invariant), must retain their validity overtime; which aspects to consider as invariants depends on the application area [ter Bekke, 1992]. *Invariants* are defined as a number of aspects that do not change even under continuous change when modeling the dynamic aspects of reality [ter Bekke, 1992]. Thus, the creation of the relevant data abstractions is an important aspect in data modeling; the implementation details can be dealt after the abstractions have been produced.

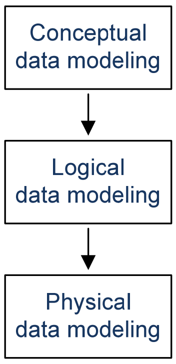


Figure 2-1 The parts of data modeling

Data modeling consists of a number of sub-phases, each emphasizing the appropriate aspect of the process, in order to satisfy the enormous diversity in user requirements and constraints [ter Bekke, 1992]. Clearly defined and verifiable results are produced at the end of each sub-phase, rendering the complete design process manageable [ter Bekke, 1992]. As shown in Figure 2-1, data modeling consists of three extensive parts. It starts from the **Conceptual data modeling**, followed by the **Logical data modeling**, and then continued with the **Physical data modeling**. Note that each part is independent of each other. However, each part needs to be consistent with the other parts in order to create a proper data model. In the following sections, each part will be briefly explained.

2.2.1 Conceptual Data Modeling

Conceptual data modeling produces high level abstraction of a system, which is represented in the form of a conceptual schema. The input of this phase is the requirements specification produced from the requirements analysis, which is performed prior to this phase. Thus, understanding and transforming requirements specification into the conceptual schema is an important aspect in the conceptual data modeling. Detailed information on the requirements analysis will be provided in section 2.4.

A single conceptual schema should represent the fundamental structure of the organization's data [Hay, 2007]. Hay explains that a conceptual data model is fundamentally a set of assertions about the nature of the organization; it is composed of entity classes, attributes describing the data they represent, and lines representing the relationships between pairs of them. The goal of conceptual schema design, where the Entity Relationship (ER) and Unified Modeling Language (UML) approaches are most useful, is to capture real-world data requirements in a simple and meaningful way that is understandable by both the database designer and the end user [Teorey et al, 2009].

A conceptual data model describes a real situation, completely independent of any technology that might be used to implement a system [Hay, 2007]. In more detail Hay explained that the same conceptual data model may be used to implement a multitude of completely different kinds of systems. The entity classes can be rendered as either relational tables or object-oriented classes; meanwhile, the relationships in a relational world become foreign keys, in an object-oriented world, they become behaviors by which each of the classes gains access to other classes [Hay, 2007]. Thus, a conceptual data model represents the abstraction of an organization, which is technology independent; this abstraction can be implemented into many realizations depending on the selected methodology or technology and also the development trajectory.

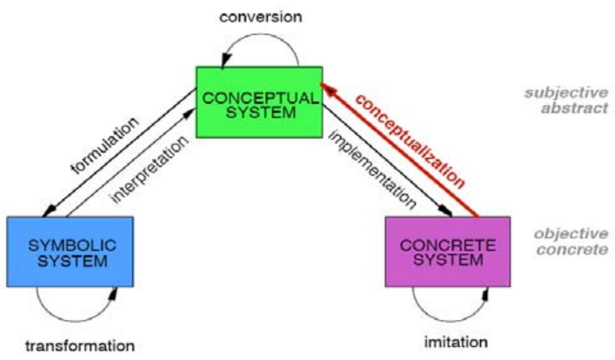


Figure 2-2 The conceptualization of model triangle [Dietz, 2006]

According to the model triangle as seen in Figure 2-2, three categories of systems can be distinguished: the *concrete systems*, *symbolic systems*, and *conceptual systems* [Dietz, 2006]. Subjective abstract means that it concerns things that are unique for every distinct subject; it represents what the subject has in mind, which is not observable by human being. On the other hand, objective concrete means that it concerns things outside the human mind and that has a large extent an existence on its own, which is observable by human beings. *Conceptualization* is a conceptual model of a concrete system [Dietz, 2006]. The conceptual model is actually something in mind, thus it is considered as a subjective abstract. The representation of conceptual model into diagram, formal language, or other symbolic system is called *formulation* [Dietz, 2006]; for example, the notation of ER-D, and UML to represent a conceptual model. According to Dietz, formulation is required for the purpose of communicating the conceptual model.

Dietz mentions that two fundamentally different types of conceptual models can be distinguished: the white-box (WB) and the black-box (BB). A white-box (WB) model conveys the construction perspective on a system, it captures the construction and the operation of a system, while abstracting from implementation details [Dietz, 2006]. Meanwhile, a black-box (BB) model of a system corresponds with the function perspective, which means one is exclusively interested in its functions and its (external) behavior [Dietz, 2006]. Dietz mentions that the relationship between the two perspectives is that the function and the behavior are brought about, and explained, by the construction and the operation of the system. However, BB models and WB models are fundamentally different types of models; there is no way of simply mapping one to the other [Dietz, 2006].

2.2.2 Logical Data Modeling

Logical data modeling produces a logical data model, which is represented in a logical schema, from a conceptual data model. It provides a logical solution to a data project. As stated in Infogoal, a logical data model provides more details than the conceptual data model which is nearly ready for the creation of a database; these details include attributes, the individual pieces of information that will be included [Infogoal, 2009]. According to Hay, a logical schema describes data in terms of a particular data management technology. In the early days, this might have been a hierarchical database management system (DBMS), or a networked one; in more recent times, this describes the tables and columns of a relational DBMS, the classes of an object-oriented program, or XML tags [Hay, 2007].

2.2.3 Physical Data Modeling

Physical data modeling is the last phase in data modeling; it creates a physical data model from a logical data model. It describes the implementation of data in a physical database [Infogoal, 2009]. As described by ter Bekke, physical data model consists of database schema for the relevant database management system (DBMS) and the database produced. It converts the formal specifications into a schema which can be applied to DBMS [ter Bekke, 1992]. In this phase, technical design aspects are taken into consideration. Thus, it depends a lot on the selected implementation technology, for instance the relational DBMS.

2.3 Data Model versus Data Semantics

A complete implementation of an information system related to data project is not required to be performed in this master thesis project. Above all, in this master thesis project we only require an understanding to the semantics of data messages. Data and the relationships between them are considered as the important aspect that should be paid attention. Thus, part(s) of data modeling that does not focus on data relationships will be discarded.

Apparently, the logical data modeling and physical data modeling need to be discarded. It is because both of them do not offer any new notion related to data relationships; on the other hand, they emphasize on data implementation, which depends a lot on the selected technology. The logical data modeling defines logical schemas, which focus on the implementation strategy for the selected technology. Moreover in the physical data modeling, the physical schemas that are ready to be implemented are defined based on the specifications of the selected technology.

On the other hand, a conceptual data model represents the fundamental structure of the organization's data that represents the nature of organization. It consists of entity classes and attributes that are significant to the organization, as well as the relationships between them. Such an organization's data structure, according to Hay, indicates a coherent structure that represents the *semantics of an organization*. It is this coherent structure that should be the basis for any system design effort, regardless of the technology [Hay, 2007].

In creating a conceptual schema, the designer has full responsibility in the process of understanding and transforming the requirements specification into the conceptual schema. Since a logical schema and physical schema is built based on the conceptual schema, it can be stated that a conceptual schema helps to ensure correctness, clarity, adaptability and productivity of an information system [Halpin, 2001]. Therefore, the conceptual data modeling is considered as a critical phase in data modeling. This statement is encouraged by ter Bekke, which stated that conceptual data modeling is the main phase in data modeling [ter Bekke, 1992]. In addition, the logical data modeling and physical data modeling concern about the implementation strategy, thus, several modifications might be performed to the conceptual schema in order to accomplish an effective and efficient implementation strategy. There is possibility that the modifications change the preceding semantics defined earlier in the conceptual schema. As stated by Meersman, the relational database schema that is obtained from conceptual schemas, which are flattened into tables, it often loss most information about the roles and concepts involved [Meersman, 2001].

Data represents the resource available in an organization, which are structured in a conceptual data model. It is understood that data are the essential components of a conceptual data model. Therefore, it is considered that the semantics of data can be found in the conceptual data model. In other words, we can say that the semantics of an organization comprises the semantics of the data. Thus, a conceptual data model is considered capable to define the semantics of data.

As compared to the other parts of data modeling, the conceptual data modeling is considered to be the only part that has a strong relationship with data semantics. In order to accomplish effective communication, which is required in service executions, conceptual schema needs to be specified by using concepts and language that people can readily understand [Halpin, 2001].

2.4 Requirements Analysis

The notion of requirements analysis has been mentioned earlier in the conceptual data modeling section. This process needs to be performed prior to data modeling. The result of requirements analysis, which is the requirements specification, is necessary for the input of the conceptual data modeling.

In the requirements analysis, elements that are considered relevant and important in the system are determined. For examples, the data required for the system, as well as their relationships. Several methods can be used to define requirements, such as by interviewing the involved actors, or analyzing the relevant documents. Above all, since the requirements specification is needed for conceptual data modeling, it has big influence on defining the conceptual data model. In other words, it affects the semantics of the conceptual schema. Since a conceptual schema is defined based on the requirements specification, it is important to formulate the requirements specification in an understandable form. In addition, the scope of the modeled system is also determined in the requirements analysis. Therefore, it is necessary to consider the requirements analysis as another critical aspect required for determining the semantics of data.

2.5 Conclusion

Data modeling has been selected as the starting point for investigating the semantics of data. It consists of three main parts, the conceptual data modeling, logical data modeling, and physical data modeling. Apparently from the three parts, only the conceptual data modeling is considered having a strong relationship with data semantics. The semantics of data message is considered able to be derived from its conceptual schema. In addition to conceptual data modeling, the requirements analysis becomes another important aspect for data semantics since it has big influence to data modeling, in particular for the conceptual data modeling. Therefore, both the requirements analysis and conceptual data modeling are regarded as the important aspects required for defining data semantics. Both of them should be specified in an understandable representation.

Chapter 3 The Requirements for Specifying the Semantics of Data Message

From the previous chapter it was found out that the *requirements analysis* and *conceptual data modeling* are the aspects of data modeling that have strong relationship with data semantics. In this chapter, the requirements for specifying the semantics of data message will be formulated based on both aspects. However, both aspects can only be used to specify the *semantics of an organization*, not in particular the *semantics of data messages*. Due to by applying the requirements analysis and conceptual data modeling, only the conceptual data model(s) of an organization will be produced. As mentioned in the previous chapter, data is the components of the semantics of organization, thus, the semantics of data message are contained in the conceptual data model. This implies that an additional process needs to be performed to define the semantics of data messages from the semantics of organization.

Two phases for identifying the requirements for specifying the semantics of data messages will be explained in this chapter:

1. The first phase is defining the requirements for specifying the semantics of an organization. These requirements can be found in section 3.1.
2. The second phase is determining the requirement for specifying the semantics of the data of interest. The data of interest represents the data message that the semantics need to be found out. This requirement is exhibited in section 3.2.

The formulated requirements for specifying the semantics of data messages will consist of the requirements identified from both phases. To validate the formulated requirements, a comparison of the requirements against the ORM's procedure for creating conceptual schema is provided in section 3.3. The conclusion of this chapter is provided in section 3.4.

3.1 Requirements Identification for Specifying the Semantics of an Organization

This section describes the requirements for specifying the semantics of an organization. The essential features of the requirements analysis and conceptual data modelling will be identified and used for formulating the requirements.

3.1.1 The Requirement 1 Identification from the Requirements Analysis

The requirements analysis is an important step in any data project lifecycle. It is required to be performed prior to the conceptual data modeling, since the result is required for the input of the conceptual data modeling. As stated in [Batini et al., 1992], the design of a schema is influenced by the kinds of initial requirements available. In the requirements analysis, all important and relevant aspects of the system or organization need to be identified. These aspects depend on the selected method. For example, in the ER approach, one needs to define the data elements, the relationship among them, and also the available constraints.

Toerey mentions that the requirement analysis for data modeling is typically the most labor intensive, since the data designer needs to determine exactly what the data is to be used for and what it must contain. Several methods for gathering requirements can be performed, such as interviewing the involved actors, and also analyzing the relevant documents. All of the requirements should be satisfied and well documented in a detailed organization's requirements specification. In addition, organization's requirements can be used as the foundation to determine the scope or the boundary of the modeled system. Based on the above explanation, the requirement 1 for specifying the semantics of data message is identified as seen in Table 3-1 below.

Table 3-1 The requirement 1 for specifying the semantics of data message

Requirement 1: *The identification of organization's requirements*

3.1.2 Requirements Identification from the Conceptual Data Modeling

In the following part, the essential features that need to be available in conceptual data models will be identified. These features represent the requirements for specifying conceptual data models.

3.1.2.1 The Requirement 2 Identification

The important and relevant aspects of organization have been identified in the organization's requirements as specified in the Requirement 1 above. After that, the organization's requirements need to be translated to construct a conceptual data model. The first thing to do is to identify the elements or information items of organization required for its conceptual data model. Validation may become the main activity in the transformation process. By validating we filter organization's requirements, so only the proper elements will be identified and the unnecessary elements can be discarded. For some methodology, a more complex process may be performed.

In addition to identifying the elements of organization required for conceptual data models, these elements should also be organized. It can be done by classifying the elements according to their roles. Roles are usually defined in each methodology, so we just can follow the specified roles in the selected methodology. For instance in the ER approach, the specified roles are *entity*, *attribute*, and *relationship*. It means if we use ER as the selected methodology we need to classify the elements of organization as entities, attributes, and also relationships. Based on the aforementioned description, the requirement 2 for specifying the semantics of data message is formulated as shown in Table 3-2.

Table 3-2 The requirement 2 for specifying the semantics of data message

Requirement 2: *The identification and classification of the elements of organization*

3.1.2.2 The Requirement 3 Identification

Relationships denote the elements that represent associations among objects (or entity types). Available relationships may have been identified during the fulfillment of the previous requirement. In addition, several specifications need to be defined for every relationship, those are:

- *The degree of relationships, such as binary, ternary, etc.*
  A binary relationship is a relationship between two distinct objects or entity types. A ternary relationship is a relationship between three distinct objects or entity types and so on. A binary relationship may relate an entity type to itself; such a binary relationship is called a *ring*. A relationship on more than two entity types is called n-ary relationship. An n-ary relationship with n > 2 can often be replaced by a number of distinct binary relationships, and this is a good idea if the replacement expresses true binary relationships for the system. However, in some cases, a ternary relationship can not be decomposed into expressive binary relationship.

- *The cardinality of entity participating in a relationship, as well as the optional or mandatory existence.*
  The concepts of minimum and maximum cardinality in which an entity participates in a relationship are illustrates in Figure 3-1. Figure 3-1 (a), (b), and (c) represent entities E and F on the left and right, respectively, by two sets; elements of the two sets are connected by a line exactly when a relationship R relates the two entity occurrences represented. Thus, the connecting lines themselves represent instances of the relation R.

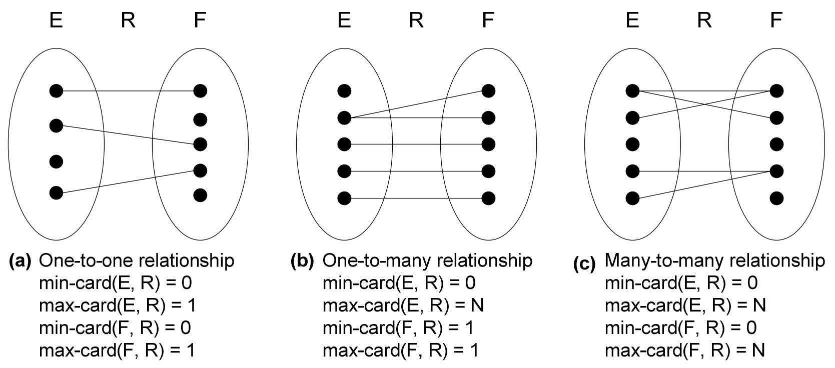


Figure 3-1 Examples of relationships R between two entities E and F [Teorey et al, 2009]

The minimal cardinality, or min-card of E in R, denoted as `min-card(E,R)`, is the minimum number of mappings in which each element of E can participate. If `min-card(E,R)=0`, we can say that E has an **optional** participation in the relationship R, because some elements of E may not be mapped by the relationship R to elements of F. If `min-card(E,R)>0`, we can say that E has a

mandatory participation in the relationship R, because each element of E must correspond with at least one element of F. The maximal cardinality, or max-card of E in R, denoted as `max-card(E,R)`, is the maximum number of mappings in which elements of E can participate. If `max-card(E,R)=1`, then E is said to have a **single-valued** participation in the relationship R. If `max-card(E,R)=N`, then E is said to be **multivalued** in the relationship.

If `max-card(E,R)=1` and `max-card(F,R)=1`, it is called **one-to-one** relationship. One-to-one relationship indicates that both E and F are single-valued. If `max-card(E,R)=N` and `max-card(F,R)=1`, it is called **one-to-many** relationship. If `max-card(E,R)=1` and `max-card(F,R)=N`, it is called **many-to-one** relationship. Many-to-one relationship or one-to-many relationship denotes that E is single-valued and F is multivalued, or the reverse. Finally, if `max-card(E,R)=N` and `max-card(F,R)=N`, where N stands for any number greater than 1, it is called **many-to-many** relationship. Many-to-many relationship signifies both entities E and F are multi-valued in the relationship. For an entity E takes part in a relation R with `min-card(E,R)=x` (x is either 0 or 1) and `max-card(E,R)=y` (y is either 1 or N), we can use a notation to represent the minimum-maximum pair (x, y): `card(E,R)=(x,y)`.

Based on the explanation above, the requirement 3 for specifying the semantics of data message is formulated as exhibited in Table 3-3.

Table 3-3 The requirement 3 for specifying the semantics of data message

Requirement 3: *The specifications of the relationships between objects*

3.1.2.3 The Requirement 4 Identification

An important aspect that affects conceptual schema is constraint. Much of the meaning of the model of data required by application can only be expressed as constraints [Cooper and Qin, 2006]. By expressing constraints explicitly in conceptual data models, we can control how and when those constraints are imposed. The constraints which are made on the structure of schemata should also be accessible so that the users can have a clearer understanding of the data model being used and so that the model itself can be modified is required [Cooper and Qin, 2006]. This implies that constraints are introduced for semantic and integrity reasons [ter Bekke, 1992].

Several constraints in relationship have been mentioned in the Requirement 3. In addition to those type of constraints, there are other types of constraint can still be specified. One type of constraint that is commonly specified in conceptual data models is constraint that is applicable to the value of the elements of conceptual data models. Such constraints are identified in order to determine the restrictions or limitations of the elements. According ter Bekke, several types of restriction that can be specified to values are:

- *Representation*
  It defines how the values are specified, such as the primitive data type of the values, how many digits are required, how many decimal digits, and etc.
- *Enumeration*

It itemizes the possible values for the element. For example the element `working_day` should be filled in by any of these values: {'`Monday`', '`Tuesday`', '`Wednesday`', '`Thursday`', and '`Friday`'}.

- *Range*
  It determines the possible range for the element's value, e.g., grade is between 0 and 10, employee_age is (20...65), and so on.
- *Pattern*
  Values should be specified according to the defined pattern. For instance, student_id is specified by two letter department code and four numbers for student number (XX9999).

The types of constraints and how they are represented is considered depending on the selected methodology that is used to represent conceptual data models. Therefore, for each selected methodology, there are various types of constraint that can be specified. The necessity on specifying constraints is formulated in the requirement 4 for specifying the semantics of data message as defined in Table 3-4 below.

Table 3-4 The requirement 4 for specifying the semantics of data message

Requirement 4: *The specification of (additional) constraints*

3.1.2.4 The Requirement 5 Identification

As an abstraction of an organization, the following concepts need to be applied in a conceptual schema: *aggregation* and *generalization*. The necessity on the two aspects is formulated as the requirement 5 for specifying the semantics of data message as exhibited in Table 3-5.

- **Aggregation**
  Aggregation abstracts a relationship between objects to a higher level aggregate object [ter Bekke, 1992]. In more detail, ter Bekke explained that aggregation is the type of abstraction where a certain number of different properties are combined into a new one, the salient point being that new, named objects are established [ter Bekke, 1992]. As stated by Batini, an aggregation defines a new class from a set of (other) classes that represent its component parts.

- **Generalization**
  Generalization abstracts a number of common characteristics to a higher level generalized object [ter Bekke, 1992]. The idea is that several entities with common attributes can be generalized into a higher-level supertype entity, or, alternatively, a general entity can be decomposed into lower-level subtype entities [Teorey et al, 2009]. Batini mentioned that a generalization defines a subset relationship between the elements of two (or more) classes. It is extremely useful because of its fundamental inheritance property: in a generalization, all the abstractions defined for the generic class are inherited by all the subset classes [Batini et al., 1992]. Specialization is the term used to denote the inverse of generalization; it represents the subtype of an object.

Table 3-5 The requirement 5 for specifying the semantics of data message

Requirement 5: *The specification of the available abstractions between objects*

3.2 Requirement Identification for Specifying the Semantics of the Data of Interest

By having all of the previous requirements satisfied, the conceptual data model of an organization should be able to be constructed. Since a conceptual data model represents the semantics of the modeled organization, not in particular a data message, an additional step to formulate the semantics of data is required. In this additional step, we need to focus on a particular part of conceptual data model that is considered representing the data of interest, due to the semantics of the data is comprised in the conceptual data model. The selected part should be proper to denote the data that the semantics need to be specified. By concentrating on a certain part of a conceptual data model, we can obtain the semantics of that part, and also understand the relationship of that part in the system. Nevertheless, there is a possibility that the semantics of data is equivalent to the semantics of organization. Such a condition may happen when the whole system is considered appropriate representing data message. In this case, this implies that the semantics of data message is represented by the conceptual data model.

Focus on a certain part of conceptual data model is performed by grouping the related elements (or information items) that are considered suitable as the components of that part. If available, documentation about the data can be used as reference during the grouping process. Grouping can be regarded as an operation that combines the identified elements including their relationships to perform a higher level construct that represents the data of interest. The group of related elements can be named by following the main element that is selected in that group. Obviously the selected naming should represent the group, which should also be appropriate for the represented data message. Based on the aforementioned description, the formulation of the requirement 6 for specifying the semantics of data message is shown in Table 3-6.

Table 3-6 The requirement 6 for specifying the semantics of data message

Requirement 6: *The data message synthesis*

3.3 Comparison against the CSDP

The Conceptual Schema Design Procedure (CSDP) is the ORM’s procedure to create conceptual schemas. It focuses on the analysis and design of data, particularly it specifies the *information structure* of the application, the *types of fact* that are of interest, *constraints* on these, and perhaps *derivation rules* for deriving some facts from others [Halpin, 2001]. Halpin mentions that the CSDP consists of the following steps:

1. Transform familiar information examples into elementary facts, and apply quality checks
2. Draw the fact types, and apply a population check
3. Check for entity types that should be combined, and note any arithmetic derivations
4. Add uniqueness constraints, and check arity of fact types
5. Add mandatory role constraints, and check for logical derivations
6. Add value, set comparison and subtyping constraints
7. Add other constraints and perform final checks

Since most of the requirements for specifying the semantics of data message are formulated based on the conceptual data modeling, thus, it is considered appropriate to validate the requirements, particularly the requirements those are related to the creation of conceptual schemas (the Requirement 2, 3, 4, and 5) against the CSDP. The validation can be performed by comparing the four requirements against the procedure, so it can be identified which steps fulfilling which requirements and whether the four requirements are all applicable for creating a conceptual schema. In the following part, the comparison between the Requirement 2, 3, 4, and 5 against the steps of the CSDP will be performed.

In the step 1 of the CSDP, the familiar information examples need to be transformed into elementary facts, and also quality checks need to be performed. In a quality checks, we need to ensure that objects are well identified [Halpin, 2001]. The transformation of familiar information and quality checks performed in the step 1 is comparable to the transformation of organization's requirements to identify and classify the elements of organization as specified in the Requirement 2. An elementary fact asserts that a particular object has a property, or that one or more objects participate in a relationship, where that relationship can not be expressed as a conjunction of simpler (or shorter) facts [Halpin, 2001]. Therefore, the identification of elementary facts can be considered representing the specifications of the relationships between objects as identified in the Requirement 3. Therefore, the Requirement 2 and 3 are considered fulfilled by the step 1 of the CSDP.

The step 2 of the CSDP is to draw a draft diagram of the fact types and apply a population check. Though useful for validating the model with the client and for understanding constraints, the sample population is not part of the conceptual diagram itself [Halpin, 2001]. Apparently the step 2 represents a step that is ORM specific implementation. It is considered that the implementation of a methodology is not necessary to be mentioned in the requirements for specifying the semantics of data message, since it is obvious that each methodology has its own way of implementation. In general, requirements should identify the aspects that need to be identified and specified, not how to implement them. Thus, no requirement is fulfilled by the step 2 of the CSDP.

The step 3 of the CSDP is to check for entity types that should be combined, and note any arithmetic derivations. The first aspect of the step 3 refers to the combination of several entity types to create a new entity that is suitable for representing the combined entity types. Thus, it indicates the fulfillment of the Requirement 2, which is the identification of an object. In addition, since the creation of the new object may be performed by combining several entities, it may be considered as an aggregation abstraction. Therefore, part of the Requirement 5 is fulfilled by the step 3. The second aspect of the step 3 is to see if some fact types can be derived from others by arithmetic. An arithmetic derivation will be in the form of rule to specify or derive another fact type. Such a kind of rule can be considered as constraint. Therefore, the specification of arithmetic derivations is considered fulfilling the Requirement 4, which is the specification of (additional) constraints. Therefore, by performing the step 3 of the CSDP, the requirement 2, 4, and 5 for specifying the semantics of data message are satisfied.

The step 4 of the CSDP is to add uniqueness constraints and check the arity of the fact types. Uniqueness constraints are used to assert that entries in one or more roles occur there at most once [Halpin, 2001]. A uniqueness constraint is represented by the maximum cardinality of the entity participating in a relationship, which is equal to 1. In addition, arity indicates the degree of relationships such as binary, ternary, etc. Therefore, checking the arity of the fact types, it means defining the degree of relationships. The specification of uniqueness constraints and arity indicate the fulfillment of the Requirement 3.

The step 5 of the CSDP is to add mandatory role constraints, and check for logical derivations. A role is mandatory (or total) for an object type if every object of that type must be known to play that role [Halpin, 2001]. Mandatory role constraints represent the specifications that exist in the relationships between objects. It can be indicated by minimum cardinality that has value 1. Therefore, the mandatory role constraints denote the fulfillment of the requirement 3, which is the specification of relationship. The second stage of the step 5 is to check for logical derivations, i.e., can some fact type be derived from others without the use of arithmetic? This implies that some rules are required to define some logical derivations between fact types. As mentioned above, such kind of rules can be considered as constraints. Therefore, similar to the arithmetic derivations, the logical derivations are considered fulfilling the Requirement 4.

In the step 6 of the CSDP we add any value, set comparison and subtyping constraints. Value constraints specify a list of possible values for a value type [Halpin, 2001]. Set comparison constraints specify subset, equality or exclusion constraints between the compatible roles or sequences of compatible roles [Halpin, 2001]. Since they represent the constraints specification, the value and set comparison constraints are considered fulfilling the requirement 4. Accordingly, subtyping constraints signifies the specification of subtyping, which defines the subtypes and supertypes and the link between them. Subtyping indicates generalization abstraction between objects. Thus, the requirement 5 is fulfilled by the step 6.

The step 7 of the CSDP is to add other constraints and perform final checks. Adding other constraints represents the specification of additional constraints, which is stated in the requirement 4. Therefore, the step 7 is considered fulfilling the requirement 4. Final checks are performed to ensure that the schema is consistent with the original examples, avoids redundancy, and is complete [Halpin, 2001]. Such an activity should be performed as part of the modeling process. It is considered not necessary to specify this activity in the requirement.

Summary of the comparison of the four requirements against the CSDP is listed in Table 3-7. As exhibited by the table, it is found out that the four requirements are satisfied by the CSDP. In addition, no additional requirements are identified. Since the step that does not fulfill any requirement indicates the step that is ORM specific implementation. This implies that the four requirements are applicable for creating conceptual data models.

Table 3-7 Comparison of the CSDP against the Requirements

Step of the CSDP	**The requirements for creating a conceptual schema**			
	Requirement 2	Requirement 3	Requirement 4	Requirement 5
Step 1	√	√		
Step 2				
Step 3	√		√	√
Step 4		√		
Step 5		√	√	
Step 6			√	√
Step 7			√	

Unfortunately, the CSDP does not specify any step that is comparable to the Requirement 1 and 6. Therefore, validation of the Requirement 1 and 6 can not be performed against the CSDP.

3.4 Conclusion

This chapter performs analysis to identify the requirements for specifying the semantics of data messages. Most of the requirements are derived from the requirements analysis and conceptual data modeling that are used to create the conceptual data model of an organization. In addition, another requirement is formulated to synthesize data message from the conceptual data model. The semantics of data message is specified by grouping the relevant elements in the conceptual data model that are considered representing the semantics of data message. At the end six requirements for specifying the semantics of data message are specified, it means the first research question has been answered. In order to specify the semantics of data message, all of these requirements should be satisfied, if possible by following the order of the requirements. In Table 3-8 the requirements for specifying the semantics of data message are listed.

Table 3-8 The requirements for the semantics of data message

Requirement 1: *The identification of organization's requirements*
Requirement 2: *The identification and classification of the elements of organization*
Requirement 3: *The specifications of the relationships between objects*
Requirement 4: *The specification of (additional) constraints*
Requirement 5: *The specification of the available abstractions between objects*
Requirement 6: *The data message synthesis*

Based on the analysis performed during the formulation of the requirements for specifying the semantics of data message, it is found out that the conceptual schema of organization needs to be modeled in the first place. Afterward, we need to concentrate on a certain part of the model that is considered suitable to represent the data we are interested to. There is a possibility that the semantics of data message is represented by the conceptual data model. This happens if the whole part of the model indicates the data message.

The CSDP is used to validate some parts of the requirements those are related to the creation of conceptual data models. From the validation of the requirement 2, 3, 4 and 5 against the CSDP, it is understood that the four requirements are applicable for creating conceptual data models. No additional requirement is identified. However, the CSDP could not be used for validating the requirement 1 and 6, since it does not contain any step that is relevant for both requirements.

Chapter 4
DEMO

In order to find out the possibility of using DEMO for specifying the semantics of data message, in this chapter, the ability of DEMO (Design and Engineering Methodology for Organizations) in fulfilling the requirements for specifying the semantics of data message will be investigated. DEMO will be briefly explained in section 4.1. Afterward, the relationships between DEMO and conceptualization will be explained in section 4.2. The possibility of DEMO in satisfying the requirements for specifying the semantics of data message will be investigated in section 4.3. A conclusion will be provided in section 4.4.

4.1 Introduction to DEMO

DEMO is a methodology for constructing an *enterprise ontology*. This methodology is based on Ψ-theory (PSI-theory), a theory about the operation of organizations. An enterprise ontology is a formal and explicit specification of a shared conceptualization among a community of people of an enterprise (or part of it) [Dietz, 2006] (the detailed information on enterprise ontology can be found in [Dietz, 2006]). DEMO consists of four aspect models in which the ontological knowledge of (the organization of) an enterprise is expressed, as seen in Figure 4-1. The way in which they are placed in the triangular shape represents their mutual relationships. Each DEMO aspect model is explained below; the information is taken from [Dietz, 2006].

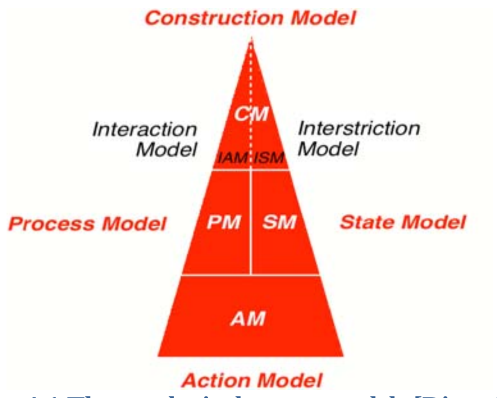


Figure 4-1 The ontological aspect models [Dietz, 2006]

1. The **Constructional Model** (CM) specifies the identified *transaction types* and the associated *actor roles*, as well as the *information links* between the actor roles and the information banks (the collective name for production banks and coordination banks). A dashed line splits the CM triangle into two parts. The left part is the **Interaction Model** (IAM); it shows the active influences between actor roles: the execution of transactions. The right part is the **Interstriction Model** (ISM); it shows the passive influences between the actor roles; i.e., the taking into account by an actor role of existing facts being active.

2. The **Process Model** (PM) contains, for every transaction type in the CM, the specific *transaction pattern* of the transaction type. This is either the complete transaction pattern or part of it, such that it comprises at least the basic pattern. In other words, it details the CM by specifying the state space as well as the transaction space of the C-world. State space is the set of allowed or lawful states; meanwhile, transaction space is the set of allowed or lawful sequences of transactions.

3. The **Action Model** (AM) specifies the *action rules* that serve as guidelines for the actors in dealing with their agenda. It contains one or more action rules for every agendum type. The action rules are expressed in a pseudo-algorithmic language, by which an optimal balance is achieved between readability and preciseness. The AM is in a very literal sense the basis of the other aspect models since it contains all information that is (also) contained in the CM, PM, and SM, but in a different, and not so easily accessible.

4. The **State Model** (SM) specifies the state space of the P-world: the *object classes* and *fact types*, the *result types*, and the ontological coexsistence *rules*. The transition space of P-world is not contained in the SM since it is fully derivable from the transition space of the C-world. The SM is put on top of AM, as seen in Figure 4-1 because it is directly based on the AM; it specifies all object classes, fact types, and ontological coexsistence rules that are contained in the AM. On the other side, the SM can also be viewed as the detailing of a part of the CM, namely, the contents of the information banks (coordination and production banks).

The representation of the aspect models in the diagrams is listed in Table 4-1 below.

Table 4-1 The representation of the aspect models in the diagrams

Aspect Models	Diagrams
Constructional Model (CM): ▪ Interaction Model (IAM) ▪ Interstriction Model (ISM)	Organization Construction Diagram (OCD): ▪ Actor Transaction Diagram (ATD) ▪ Actor Bank Diagram (ABD)
Process Model (PM)	Process Structure Diagram (PSD)
Action Model (AM)	Action Rule Specifications (ARS)
State Model (SM)	Object Fact Diagram (OFD)

Figure 4-2 shows the diagram types in which the aspect models are visualized, and also the three most useful cross-model table types. The logical sequence of producing the aspect models is anti-clockwise, starting from the Interaction Model (IAM). IAM is expressed in an Actor Transaction Diagram (ATD) and a Transaction Result Table (TRT). Next, the Process Structure Diagram (PSD) is produced, and after that the Action Rule Specifications (ARS). Next, the SM is produced, expressed in an Object Fact Diagram (OFD) and an Object Property List (OPL). Then we are able to complete the PM with the Information Use Table (IUT). Lastly, the ISM is produced, consisting of an Actor Bank Diagram (ABD) and a Bank Contents Table (BCT). Usually, an

Actor Bank Diagram is drawn as an extension of the Actor Transaction Diagram; together they constitute the Organization Construction Diagram (OCD).

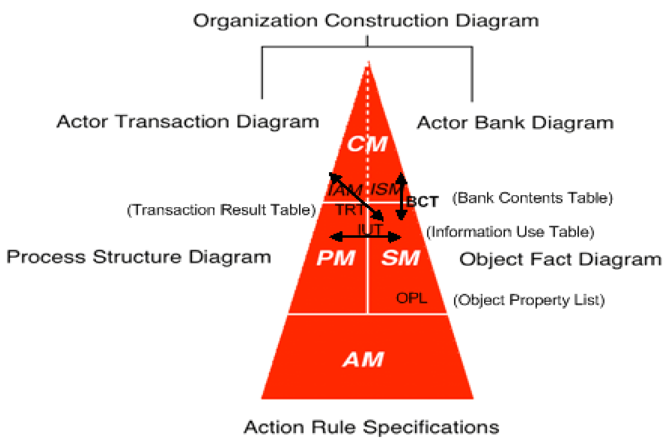

Organization Construction Diagram
Actor Transaction Diagram
Actor Bank Diagram
CM
IAM
ISM
BCT
(Bank Contents Table)
(Transaction Result Table)
TRT
IUT
(Information Use Table)
Process Structure Diagram
PM
SM
Object Fact Diagram
OPL
(Object Property List)
AM
Action Rule Specifications

Figure 4-2 The diagrams and cross model tables [Dietz, 2006]

4.2 Conceptualization and DEMO

An enterprise ontology, which is created by using the DEMO methodology, signifies the conceptualization of an organization that consists of the construction and operation of organization. The conceptualization of a system is also the main concern in the conceptual data modeling. This indicates the similarity between the DEMO methodology and conceptual data modeling, both of them represent the conceptualization of a system. However, at this point, it is still unknown, whether all of the aspects required for specifying conceptual data models can be found in DEMO. As explained in chapter 2, there are two fundamentally different types of conceptual models; the WB model and BB model. DEMO models are classified as the white-box models. It is because DEMO is used to express the construction and operation of organization, as indicated by the white-box model.

We understand that the conceptual data modeling is the main aspect in the requirements for specifying the semantics of data message. Since the DEMO methodology has several aspect models with the corresponding diagrams, it is necessary to identify which DEMO's aspect model that is suitable for representing conceptual data models, as not all of the aspect models may be required and suitable for that purpose. The identification of DEMO's aspect model that is suitable for representing conceptual data models is explained below.

The IAM defines the transactions performed by actor roles. While, the ISM defines the information banks required by actor roles in performing transactions. The actor roles, transaction types, and information banks arranged in the CM are considered not representing the fundamental structure of organization's data. It is because the CM, concerns about the operations performed by actor roles, in which the information about the production banks and coordination banks are maintained as well. Due to this reason, the CM is considered not suitable for representing conceptual data models.

The PM specifies the transaction pattern for each transaction identified in the CM. In other words, the PM only details the transaction types specified in the CM by adhering to a particular transaction pattern. The transaction pattern as specified by the PM is considered not representing the fundamental structure of organization's data. Therefore, the PM is also considered not suitable for representing conceptual data models.

An action rule in the AM specifies what has to be performed by actor roles in dealing with their agenda. Action rules are expressed in a pseudo-algorithmic language. Thus, no schema is created by the AM. As described in chapter 2, the outcome of the conceptual data modeling is expressed in conceptual schema. A conceptual schema will assist the reader of the schema to understand the abstraction of system (or organization); while an algorithmic representation may not be easy to understand by all reader, particularly the business people. Since it may contain of a lot of details that make it not so easy to understand, it is considered that the AM is not suitable for representing conceptual data model.

The SM consists of the object classes, fact types, the result types, and the ontological coexsistence rules exist in organization. Organization's data can be considered as object classes. While, fact types and result types can be considered representing the relationships between object classes. Additionally ontological coexsistence rules denote the applicable constraints in organization. If these elements are arranged, they are considered appropriate for representing the structured organization's data. This assumption is supported by Dumay, which states that the SM of DEMO is a conceptual schema of the things and facts that appear to be relevant, where the related information of the organization is modeled [Dumay, 2004].

Overall, we can conclude that from the four DEMO aspect models, only the SM is considered suitable for representing conceptual data models. However, since the SM is specified based on the other aspect models of DEMO, they are still required in the modeling process.

4.3 DEMO for Specifying the Semantics of Data Message

The possibility of using DEMO for specifying the semantics of data message will be explored in this section. How DEMO fulfills the requirements for specifying the semantics of data message will be investigated thoroughly. In total six requirements for specifying the semantics of data message are available as identified in Chapter 3; DEMO will be analyzed against each requirement in the following sections.

4.3.1 Fulfillment of the Requirement 1

In this section the possibility of DEMO in satisfying the Requirement 1, the identification of organization's requirements, is investigated. In this case, the Requirement 1 can be fulfilled by identifying how DEMO specifies organization's requirements.

In [Dietz, 2006], Dietz explains a method to acquire the basis for a correct and complete set of aspect models of an enterprise ontology. Part of this method is regarded appropriate as the requirements analysis. This method can be used to specify organization's requirements that will become the input for the IAM. The main activity performed in this method is analyzing the available documentation about the enterprise, of whatever kind, and in whatever form.

The summary of the part of the method in DEMO for the requirements analysis is provided below, which is taken from [Dietz, 2006]. This method is slightly adjusted to make it appropriate for the requirements analysis in data modeling, so the proper organization's requirements can be provided.

1. **The Performa-Informa-Forma Analysis**
   In this step all available pieces of knowledge are divided into three sets, according to the distinction axiom. As mentioned in the *distinction axiom*, there are three distinct human abilities playing a role in the operation of actors, called the *performa*, *informa*, and *forma*. The summary of distinction axiom for coordination and production can be seen in Figure 4-3. The forma ability concerns the form aspects of communication and information. The informa ability concerns the content aspects of communication and information. The performa ability concerns the bringing about of new, original things, directly or indirectly by communication.

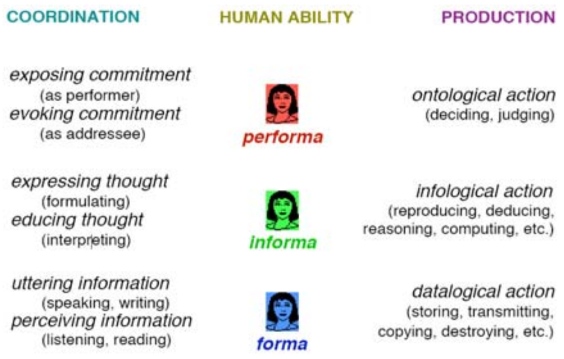


Figure 4-3 The summary of the distinction axiom

2. **The Coordination-Actors-Production Analysis**
   The Performa items are divided into C-acts/results, P-acts/results, and actor roles, according to the operation axiom. The *operation axiom* states that the operation of an enterprise is constituted by the activities of actor roles, which are fulfilled by subjects. In doing so, these subjects perform two kinds of acts: *production acts* and *coordination acts*. These acts have definite results: *production facts* and *coordination facts*, respectively, which can also be referred to as P-results and C-results. By performing production acts (P-acts) the subjects contribute to bringing about the goods and/or services that are delivered to the environment of the enterprise. By performing coordination acts (C-acts) subjects enter into and comply with commitments towards each other regarding the performance of production acts. The graphical representation of the operation axiom can be found in Figure 4-4.

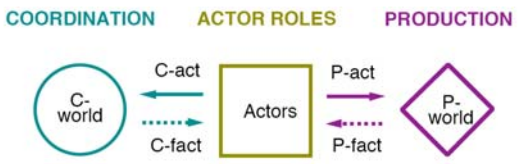


Figure 4-4 Graphical representation of the operation axiom

3. **The Transaction Types Identification**
   The *transaction axiom* states that coordination acts are performed as steps in universal patterns, which are called transactions; they always involve two actor roles and are aimed at achieving a particular result. An illustration that shows the basic pattern of the transaction axiom can be seen in Figure 4-5. In this step, we are going to cluster the identified C-acts/facts and P-acts/results into transactions based on the transaction axiom. These transactions are arranged into a list of transaction types.

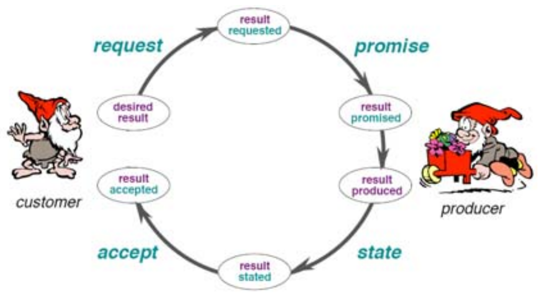


Figure 4-5 An illustration for the transaction axiom

Note that this method is only an aid and not a *dogma*. Therefore, one may freely iterate through these steps, and even skip a step. At the end, the organization's requirements will be specified in the form of a list of the identified transaction types. Based on the explanation above, we have identified that the Requirement 1 can be satisfied by DEMO.

4.3.2 Fulfillment of the Requirement 2, 3, 4, 5

In this section, the possibility of DEMO in fulfilling the Requirement 2, Requirement 3, Requirement 4, and Requirement 5 will be explored. These requirements represent the important aspects that exist in the conceptual data modeling. The main activity in the conceptual data modeling is to understand and transform requirements specification to create conceptual data models. This activity will be carried out in DEMO by following the logical sequence of producing the aspect models, starts from the CM and ends with the SM as the conceptual schema, which has been identified earlier.

4.3.2.1 Fulfillment of the Requirement 2

The process starts by transforming the list of transaction types, which has been produced in the requirements analysis, to create the CM. Dietz has explained a method to create the TRT and ATD of the CM, which is explained below.

1. **The Transaction Pattern Synthesis**
   In order to accomplish the transaction axiom, the transaction types will be completed by specifying the result of each transaction type; the generated *result type*. For every transaction type, the result type is precisely formulated. Precise formulation of the result type is required, since it affects the conceptual schema that will be produced. The criterion for correctness (of the result type) is that the instantiation of the variable in the result formulation (or the combination of the variables, if there is more than one) identifies an entity uniquely [Dietz, 2006]. In addition, in this step we can validate the identified transaction types and also eliminate the unnecessary transaction types. The Transaction Result Table (TRT) can now be produced.

2. **The Result Structure Analysis**
   The composition axiom states that every transaction is enclosed in some other transaction, or is a customer transaction of the organization under consideration, or is a self-activation transaction. In this step, one needs to exhibit the available dependencies between the production acts or results. If the dependency is such that the bringing about of a result B is initiated during the process of bringing about a result A, and that the completion of A has to wait

for B to completed, then B is a component of A. This step helps in determining the causal and conditional relationships between transactions.

3. **The Construction Synthesis**
   For every transaction type the initiating actor role and the executing actor role are identified based on the transaction axiom. This is the first step in producing the Actor Transaction Diagram (ATD).

4. **The Organization Synthesis**
   A definite choice has to be made as to what part of the construction will be taken as the organization to be studied and what part will become its environment. It is a minor step to decide what belongs to the kernel of the organization and what belongs to the environment, as well as what constitutes the interface between them. The ATD can now be finalized.

By applying this method, the Interaction Model (IAM) of the CM can be constructed. IAM is expressed in an ATD and a TRT. The IAM of an organization consists of the transaction types in which the identified actor roles participate as the initiator or executor. All actor roles are depicted in the ATD, but there is a boundary dividing the set of all (relevant) actor roles into the kernel of the organization (known as the composition), and the environment. Thus, by implementing the ATD, the scope or boundary of the organization can be defined. Legend of the ATD is exhibited in Figure 4-6, and basic construct in the ATD is shown in Figure 4-7.

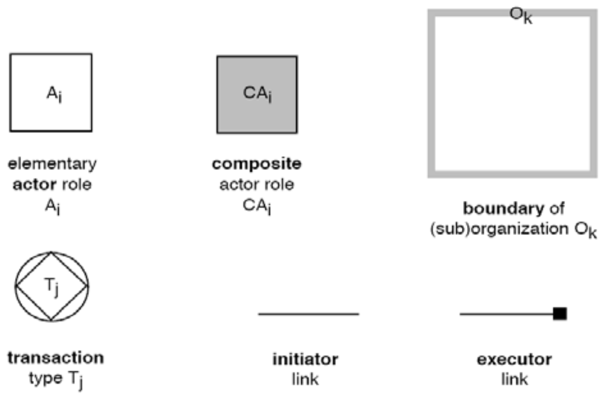

A_i
elementary actor role A_i
CA_i
composite actor role CA_i
O_k
boundary of (sub)organization O_k
T_j
transaction type T_j
initiator link
executor link

Figure 4-6 Legend of the ATD (interaction)

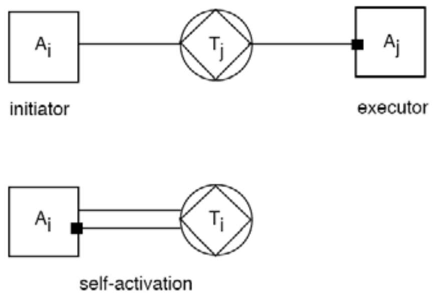


Figure 4-7 Basic construct in the ATD

The next aspect model that needs to be implemented after the IAM is the Process Model (PM). A PM is expressed in a Process Structure Diagram (PSD) and an Information Use Table (IUT), which can be produced after the SM is finalized. The PSD specifies for every included transaction type, the process steps that are allowed to be taken to the next phase. The process steps are specified based on the transaction pattern as defined in the transaction axiom. A process step is a C-result and its causing

C-act. Legend of the PSD can be seen in Figure 4-8. An IUT specifies, for every object class, fact type, and result type from the State Model (SM), the process steps of the PM in which its instances are used; these steps can be derived from the Action Model (AM).

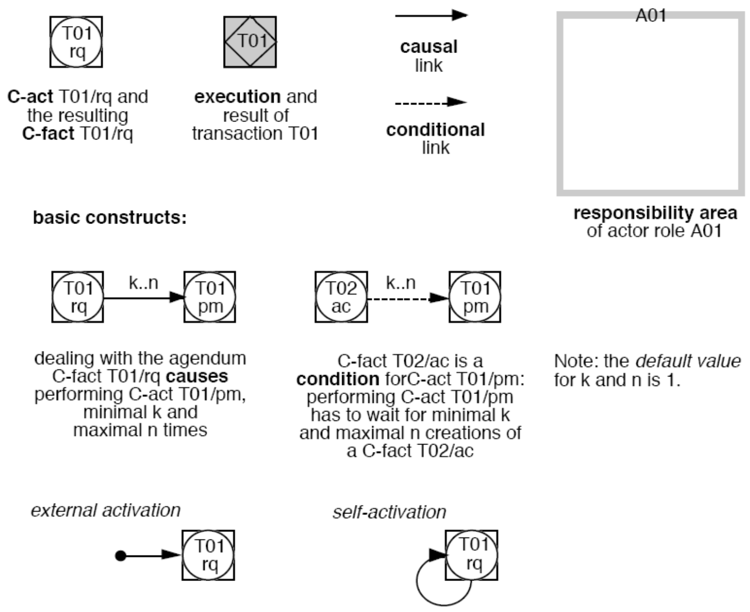
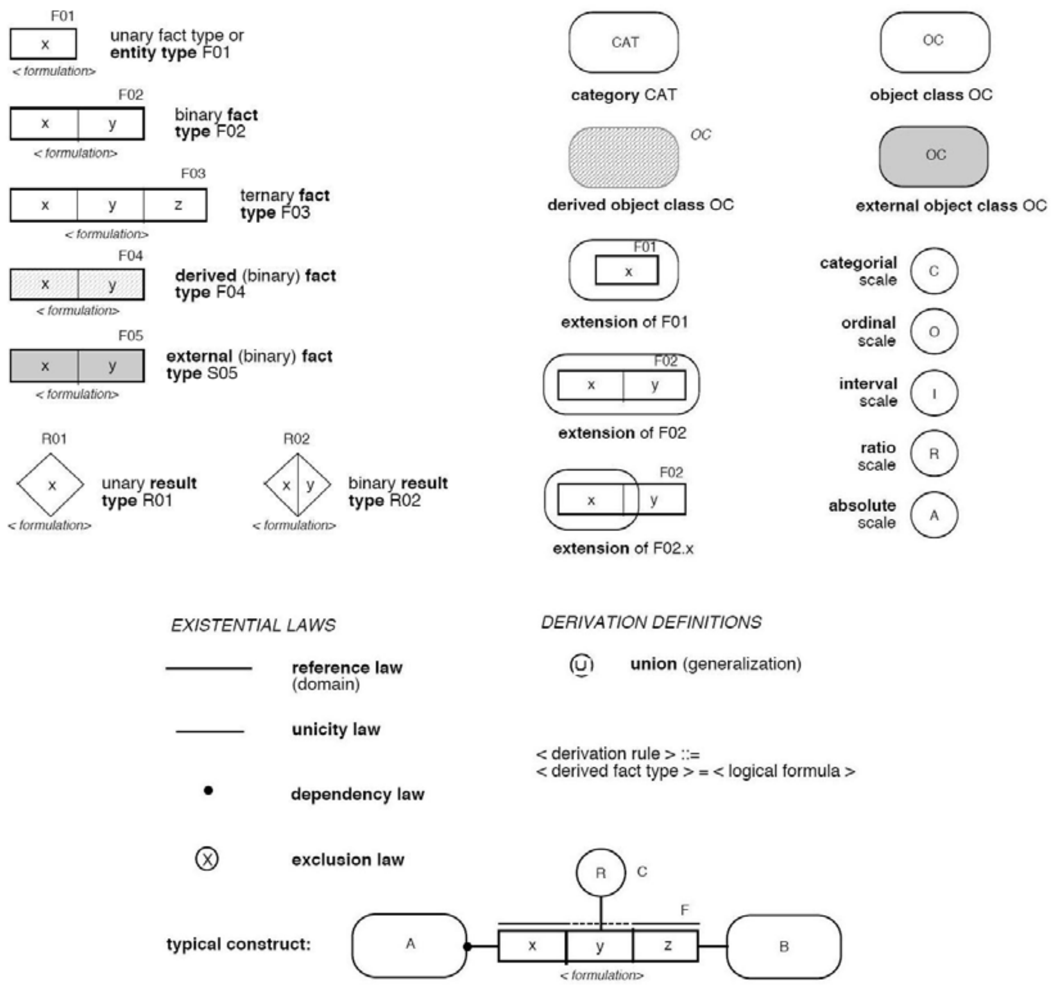

T01 rq
C-act T01/rq and the resulting C-fact T01/rq
T01
execution and result of transaction T01
causal link
conditional link
A01
responsibility area of actor role A01
basic constructs:
T01 rq
k..n
T01 pm
dealing with the agendum C-fact T01/rq causes performing C-act T01/pm, minimal k and maximal n times
T02 ac
k..n
T01 pm
C-fact T02/ac is a condition forC-act T01/pm: performing C-act T01/pm has to wait for minimal k and maximal n creations of a C-fact T02/ac
Note: the default value for k and n is 1.
external activation
T01 rq
self-activation
T01 rq

Figure 4-8 Legend of the PSD

The Action Model (AM) is the next aspect model of DEMO that needs to be implemented. It is the most detailed and comprehensive aspect model. The AM of an organization consists of a set of action rules for every agendum type. Those action rules, which are specified in the Action Rule Specifications (ARS), only act as guideline for an actor. Without a complete AM, it is principally not possible to produce a correct state model as well as a correct interstriction model [Dietz, 2006].

Next, the SM is produced after the AM. The SM specifies the state space of the P-world: the *object classes* and *fact types*, the *result types*, and the ontological *coexistence rules*. An SM is expressed in an Object Fact Diagram (OFD) and an Object Property List (OPL). An OPL is used to specify fact types that are proper (mathematical) functions, and of which the range is a set of values. The fact types that are listed in an OPL are not included in the diagram; they are called properties (of object classes). The information required to be specified in an OPL is the *property name*, the *object class* in which the property belong to, and the *scale* of the property type. The content of both OFD and OPL of an organization are completely determined by its AM. Thus, only the information items that are relevant for the operation of the organization are included. In principle, one can find the categories, object classes, fact types, and result types, as well as all pertaining existential laws and all derivation rules, in the action rule specifications [Dietz, 2006]. Legend of the OFD is shown in Figure 4-9.

The OFD is based on the WOSL (World Ontology Specification Language). WOSL is used to express ontological models, consisting of concepts or predicates, which represent individual facts in the world. It is expressed in graphical notation, which is adopted from the graphical notation that is applied by one of the fact oriented conceptual modeling language, named ORM. ORM is used as the basis of WOSL, it is because the similarity between the ontology of the world and the conceptual schema of a database [Dietz, 2006]. This explanation supports the previous statement that the SM is suitable for representing conceptual data models. In addition, by expressing the model in terms of natural concepts, like objects and roles, it provides a conceptual approach to modeling [Halpin, 2001].



The domain of role x of fact type F is object class A.
The domain of role y of fact type F is ratio scale C.
The domain of role z of fact type F is object class B.
There is a unicity law for the combination of roles x and z:
a tuple < a,-,b > can only appear once in a population of F.
There is a dependency law for A:
for every a ∈ A there must be a tuple < a,- ,- > in F.

Figure 4-9 Legend of the OFD

The ontological model of a world consists of the specification of its state space and its transaction space [Dietz, 2006]. *State space* is the set of allowed or lawful state; it is specified by means of the state base and the existence laws [Dietz, 2006]. As described by Dietz, the *state base* is the set of statum types of which instances may exist in a state of the world; the *existence laws* determine the inclusion or exclusion of the coexistence of stata. *Transition space* is the set of allowed of lawful sequence of transitions; it is specified by the transition base and the occurrence laws [Dietz, 2006]. Dietz explained the *transition base* as the set of factum types in which instances may occur in the world, and every instance has a time stamp as the event time. The *occurrence laws* determine the order in which facta are required or allowed to occur [Dietz, 2006]. To sum up, the ontology of the world represents the set of facts comprising statum types and factum types.

A *statum* (plural: *stata*) is something that is the case, has always been the case, and will always be the case; it is constant [Dietz, 2006]. It represents an inherent property of a thing or an inherent relationship between things [Dietz, 2006]. Stata are subject to existence laws. The examples of stata are "`the ISBN of book title T is I`", "`the maximum number of new student accepted in school S for year Y is N`", etc. Dietz explained a *factum* (plural: *facta*) as the result or the effect of an act. Facta are subject to occurrence laws. Facta can be considered as status changes of a concept of some type [Dietz, 2006]. The examples of facta are "`the phone bill for period P has been paid`", "`the packet D has been delivered`", etc.

Overall, the object classes, fact types, result types, and the ontological coexistence rules expressed in the SM represent an ontological model of the world. We have identified earlier that the SM is suitable for representing a conceptual data model; thus, conceptual data model represented by the SM will consist of the set of statum types and factum types of the ontological model of the world (of the organization under consideration). Given that the role for representing conceptual data models has been fulfilled by the SM, it is considered not necessary to continue the modeling process in DEMO, since we also have illustrated earlier in section 4.2 that the ISM is not suitable for specifying a conceptual data model. In addition, the IUT of the PSD is also not necessary to be implemented, since it only defines the process steps used by the object classes, fact types, and result types identified in the SM.

The Requirement 2 signifies the necessity to identify and classify the elements of organization under consideration. As described above, the SM identifies the essential elements of organization from the AM. In the SM, these elements are categorized as object classes, fact types, and result types, as well as all pertaining existential laws and all derivation rules as depicted in the OFD, besides, the properties of the object classes are listed in the OPL; it denotes how the SM classifying the information items. Thus, the Requirement 2 for specifying the semantics of data message is fulfilled by the SM of DEMO.

4.3.2.2 Fulfillment of the Requirement 3

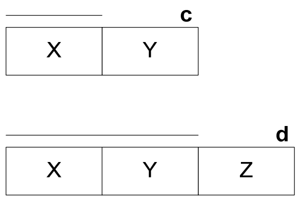
The Requirement 3 signifies the necessity to specify the relationships between objects. It is necessary to figure out how the degree of the relationships and the cardinality of the entity participating in the relationship, as well as the optional or mandatory existence are specified in the SM.

As aforementioned in section 4.2, fact types and result types of the OFD are considered representing the relationships between object classes. Fact types indicate statum types, while result types are comparable to factum types. A statum type represents an inherent property of a thing or an inherent relationship between things. It is obvious that statum types in the SM denote the relationships between objects. On the other hand, a factum type represents the result or the effect of an act. Thus, we can regard that factum types represent the relationships between actions and their effect. Since statum types indicate a more common type of relationship, how statum types are specified will be explained in detail in the following part.

Statum types can be denoted *intensionally* or *extensionally* [Dietz, 2006]. Intensional means the notation of the statum type represented as a *unary*, *binary*, *ternary*, etc; meanwhile, extensional means the notation of a statum type represented as a class (a set of similar objects) [Dietz, 2006]. The notions of intention and extension are dual, meaning that the intensional definition and the extensional definition of the statum type are semantically equivalent [Dietz, 2006]. Some illustrations for intensional and extensional notations of statum types can be found in Figure 4-9, Figure 4-17, Figure 4-19, and Figure 4-20.

The degree of relationships is considered similar with the *arity* of statum types. The arity of statum types is shown by the number of the placeholders for objects in statum types. The number of placeholders in a statum type represents the number of involved objects in a relationship. A binary statum type has two placeholders for objects; a ternary statum type has three placeholders, and so on. Therefore, a binary fact type is considered similar to binary relationship, and a ternary fact type is similar to ternary relationship. A binary statum type denotes a relationship between two distinct objects or entity types, and a ternary statum type denotes a relationship between three distinct objects or entity types, and so on. The illustration of binary and ternary fact types can be found in the legend of the OFD as shown in Figure 4-9.

As mentioned above, the SM consists of several **existential laws**: *reference law*, *unicity law*, *dependency law*, and *exclusion law* (see Figure 4-9). From these four laws, the unicity law is considered representing the cardinality of the entity participating in a relationship. The **unicity law** explains that the instances of an object involved in a relationship can only appear once in that relationship. This implies that the unicity law constraints the maximum cardinality in which an instance of an object can appear in a relationship, which is at most 1. However, it can not be used to specify the maximum cardinality more than 1. The unicity law is implemented by a bar over the object role in which the instances must be appearing only once. Examples of unicity law and their notation can be seen in Figure 4-10.



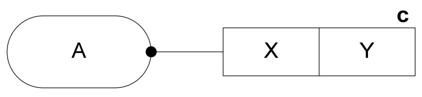
Notation of a unary ***unicity law***:
~◊ (**c**(x, y) & **c**(x, z) & y≠z)

Notation of a binary ***unicity law***:
~◊ (**d**(x, y, z) & **d**(x, y, w) & z≠w)

◊ : possibility ~ : negation

Figure 4-10 Examples of the unicity law [Dietz, 2006]

The optional and mandatory existence of the instance of relationships is influenced by minimum cardinality. Particularly for mandatory existence, the minimum cardinality must be more that 0. This implies that all of the instances of the object in the relationship must have at least one corresponding values in the relationship. Such condition can be achieved by applying the **dependency law**. On the other hand, no particular rule is required to be applied to the optional existence. The dependency law is illustrated by a black dot on the corner of the object with mandatory role. An example of the dependency law is illustrated in Figure 4-11. The explanation of the dependency law applied in Figure 4-18 (note: ignore the partition) is that for every membership *x* there must be a person *y* such that **member**(*x*, *y*) holds.



Notation of a ***dependency law*** for object class A
□ (**a**(x) => **c**(x, -))
Since it also holds: □ (**c**(x, -) => **a**(x))
(because of the reference constraint),
a(x) and **c**(x, y) are existentially dependent

□: necessity
=>: strict implication

Figure 4-11 Example of the dependency law [Dietz, 2006]

All in all, for binary relationship, there are four possible uniqueness constraint patterns: many-to-one (n:1), one-to-many (1:n), one-to-one (1:1), and many-to-many (m:n), and four possible mandatory role patterns: only the left role mandatory, only the right role mandatory, both roles mandatory, both roles optional. By taking into account both relationship constraints; there are 16 possible combinations for binary relationships. The 16 cases can be represented in the SM of DEMO as illustrated in Figure 4-12. Thus, it is confirmed that the Requirement 3 is also satisfied by the SM.

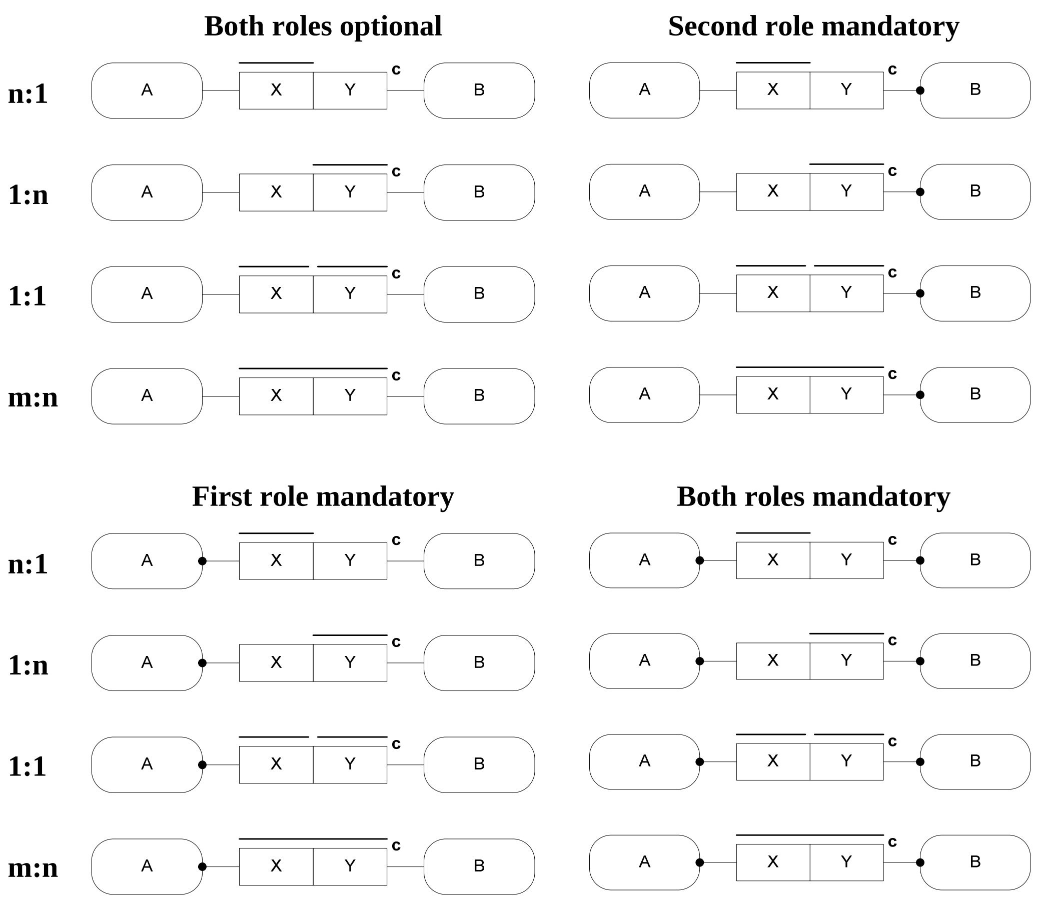
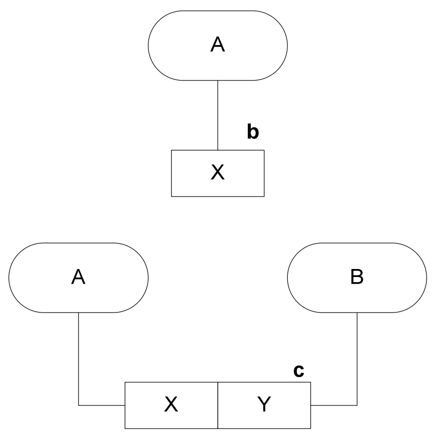


Figure 4-12 Binary relationship constraints in the SM of DEMO



Notation of a ***reference law*** for the unary statum type **b**
□ (**b**(x) => **a**(x))
◊ (**a**(x) => **b**(x))
A is called the domain of **b**

Notation of a ***reference law*** for the binary statum type **c**
□ (**c**(x, y) => **a**(x) & **b**(y))
◊ (**a**(x) => **c**(x, -))
◊ (**b**(x) => **c**(-, y))
A is called the domain of **c**.x
B is called the domain of **c**.y

□: necessity ◊: possibility =>: strict implication

Figure 4-13 Reference law for the unary and binary statum type [Dietz, 2006]

4.3.2.3 Fulfillment of the Requirement 4

The Requirement 4 identifies the necessity for specifying (additional) constraints. From the rest of the existential law of DEMO, it appears that the *reference law* and *exclusion law* can be used to specify constraints. As shown in Figure 4-13, the reference law is depicted as a connector line, it links a category or an object class to its role in the fact type. The **reference law** defines the domain of the fact type. The notation of the reference law can be seen in Figure 4-13.

The **exclusion law** prohibits the instances of a certain role in a fact type to be the same with the instance of another role in a fact type. The representation of an exclusion law and the notation is illustrated in Figure 4-14.

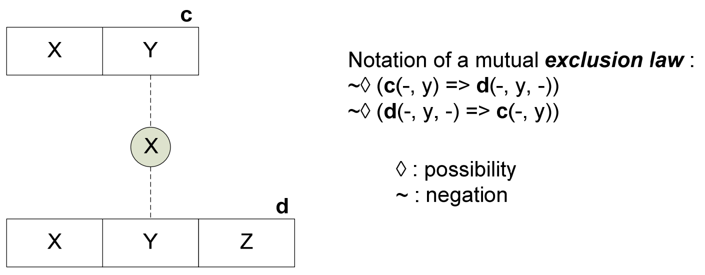

c
X
Y
X
d
X
Y
Z
Notation of a mutual ***exclusion law*** :
~◊ (**c**(-, y) => **d**(-, y, -))
~◊ (**d**(-, y, -) => **c**(-, y))
◊ : possibility
~ : negation

Figure 4-14 Exclusion law notation [Dietz, 2006]

The following example, as illustrated in Figure 4-15, shows different implementations of exclusion law: (a) No person both wrote a book and reviewed book, (b) No person wrote and reviewed the same book.

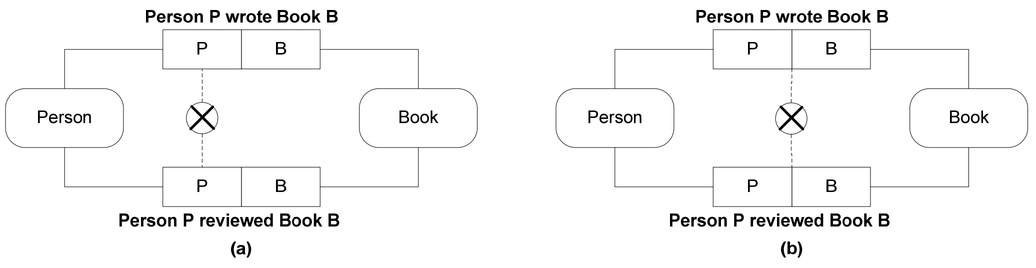


Figure 4-15 Examples of exclusion law

In DEMO, the application constraints are specified as action rules in the AM that act as guidelines for the actors in dealing with their agenda. These action rules are used to derive the categories, object classes, fact types, result types, existential laws, and derivation rules for the SM. Fact types that are proper (mathematical) functions, and of which the range is a set of values, are regarded as the properties of the categories or object classes. These properties are listed in the *property type* column of OPL. Some of these properties can be *derived fact types*. In OPL, derived fact types are indicated by an asterisk between brackets right after the property name. For each derived fact type, there is a derivation rule that states how to specify the derived fact type. Derived fact types are commonly placed underneath OPL, as a note. An example of derived fact types are area = height * width, as illustrated in Figure 4-16 below.

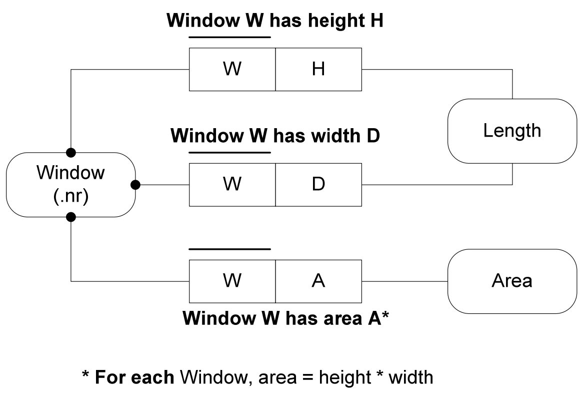


Figure 4-16 Example of derivation rule

OPL that consists of properties (including the derived fact types) and their scale can be considered specifying **value constraints**. Scale represents the type expression, indicates the domain in which the attribute is based, e.g., `String`, `Date`, etc. However, the value constraints specification in OPL are considered not sufficient, due to it only specifies to which object the properties belong, and also the scale of the properties. As, value constraint is the most common type of constraint type that is specified in data modeling, more complete specifications of value constraints should be supported. For that reason, more detailed constraints should be specified in conceptual data models.

It is considered that the Requirement 4 is only partially fulfilled by the SM of DEMO. Therefore, the SM of DEMO should be modified and/or combined with the other methodology, particularly for the purpose of improving the constraints specification. Another notation or method is required to extend the SM of the DEMO methodology.

4.3.2.4 Fulfillment of the Requirement 5

The Requirement 5 denotes the necessity of the *aggregation* and *generalization* abstraction to be the part of conceptual data models. Thus, the ability of the SM in specifying aggregation and generalization abstraction needs to be identified. In the OFD, apparently four kinds of statum type derivations are distinguished: *partition*, *aggregation*, *specialization*, and *generalization*.

Partition is defining an (new) object class as part of the available statum type. The notations of deriving statum types as partitions are illustrated in Figure 4-17.

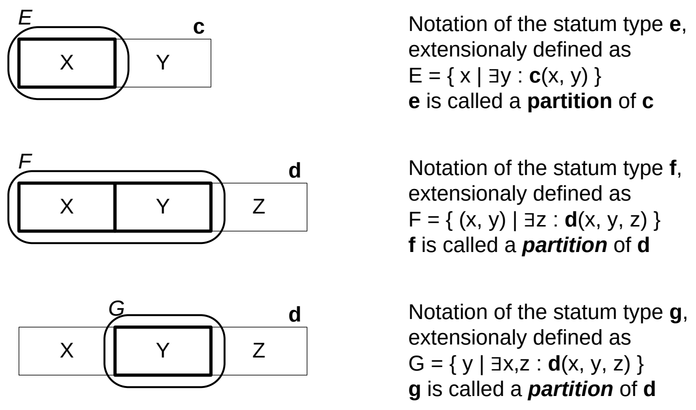

E
c
X
Y
Notation of the statum type **e**,
extensionaly defined as
E = { x | ∃y : **c**(x, y) }
e is called a **partition** of **c**
F
d
X
Y
Z
Notation of the statum type **f**,
extensionaly defined as
F = { (x, y) | ∃z : **d**(x, y, z) }
f is called a ***partition*** of **d**
G
d
X
Y
Z
Notation of the statum type **g**,
extensionaly defined as
G = { y | ∃x,z : **d**(x, y, z) }
g is called a ***partition*** of **d**

Figure 4-17 Deriving statum types as partitions [Dietz, 2006]

Dietz shows an example of partition as illustrated in Figure 4-18, which is explained as follow. The derived object class *MEMBER* is defined as the set of persons who occur in role Y in the population of the statum type **member** at some time, which is the exact set of members at that time [Dietz, 2006].

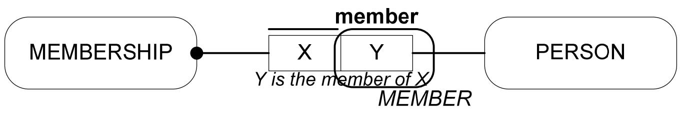


Figure 4-18 Example of a partition [Dietz, 2006]

Aggregation means conceiving a composite object types as one (new) object type [Dietz, 2006]. Dietz also states that aggregation is commonly used to create a number of binary fact types which are derived from ternary and higher arity fact types. The example of the notations of aggregation is shown in Figure 4-19. The definition of the category E in (a) is semantically equivalent to the construction (b); note that the (b) construction is recommended.

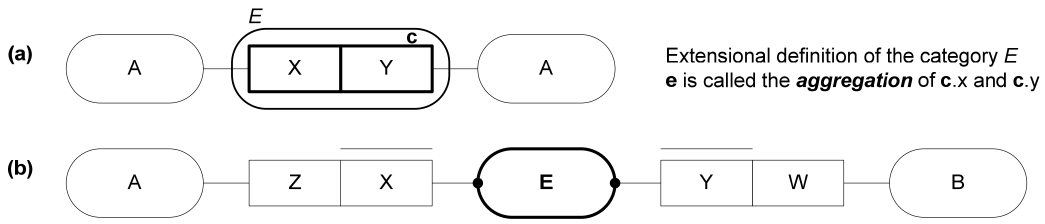

E
(a)
A
c
X
Y
A
Extensional definition of the category E
e is called the aggregation of c.x and c.y
(b)
A
Z
X
E
Y
W
B

Figure 4-19 The notations of aggregation [Dietz, 2006]

Specialization and generalization are often considered to be each other's inverses. However, in DEMO, both abstractions are distinguished based on the different way of identifying the objects in the corresponding classes [Dietz, 2006]. As described by Dietz, a **specialization** type is always ultimately a subtype of a category (or of a class that is the generalization of a number of categories). The objects in the extension of the specialization type are identified as objects in the category to which they belong. On the other hand, Dietz explained **generalization** type as always ultimately the union of two or more categories. The example of the notation of specialization and generalization is illustrated in Figure 4-20 below.

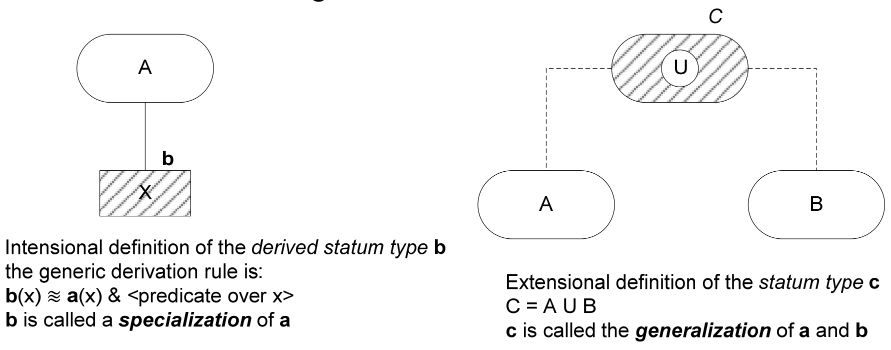

A
b
X
Intensional definition of the derived statum type b
the generic derivation rule is:
b(x) ≈ a(x) & <predicate over x>
b is called a specialization of a
C
U
A
B
Extensional definition of the statum type c
C = A U B
c is called the generalization of a and b

Figure 4-20 Example of the notation of specialization and aggregation [Dietz, 2006]

As illustrated in the aforementioned description, we can conclude that the SM satisfies the Requirement 5. In addition, the SM also provides additional abstractions: the partition and specialization. Overall, these statum type derivations support the definition or creation of new statum types. Particularly for the specialization and generalization, both of them support sub-typing.

4.3.3 Fulfillment of the Requirement 6

The Requirement 6 denotes the data message synthesis from conceptual data models. It is performed by grouping or classifying the related elements of organization depicted in the conceptual data model that are considered suitable for representing data message. It is an additional step required for specifying the semantics of data, which is performed after creating conceptual data model. DEMO itself does not define any specification for grouping the elements of the SM. Thus, in order to fulfill this requirement, DEMO needs to be extended with an additional step required for synthesizing data. This additional step is performed by means of grouping the categories, object classes, fact types, result types, existential laws, and the related derivation rules of the SM that are considered appropriate to represent data message. In order to facilitate this additional step, it is better to have all of the SM information items in one schema. It other words, it is expected that in addition to the object types, the property types, which are usually listed in OPL, are depicted in OFD as well. By having all of the information items in one schema, we can easily identify the information items that are classified as part of the group that describes the semantics of data.

However, if the produced conceptual data model might become too complicated and confusing when all of the information items are depicted in one schema, then we can keep (some of) the property types listed in OPL. By exhibiting the property types in OPL, OFD may become less voluminous and less complicated. To indicate which property types is part of the group, only the property types those are considered representing the semantics of data message are listed in OPL, the other property types can be discarded. By listing only the relevant property types in OPL, OPL will not be too spacious and only the relevant property types can be found. This should be applied as well to the derivation rules, only the derivation rules that are considered representing the semantics of data message are exhibited. By performing this additional step to the SM of DEMO, we can consider that DEMO satisfies the Requirement 6.

4.4 Conclusion

The conceptualization represented in DEMO is classified as the white box model of organization; it represents the operation and construction of organization. From several DEMO aspect models, only the SM of DEMO is considered suitable for representing conceptual data models. Since the SM is developed based on the other aspect models of DEMO; these aspect models still have essential roles in specifying conceptual data models. The sequence of the modeling process itself starts from the ATD and TRT of the IAM, followed by the PSD of the PM, and then continued by the ARS of the AM. After that the OFD and OPL of the SM can be derived from the AM. The ISM that consists of ABD and BCT as well as the IUT of the PM can be discarded, particularly since the role for specifying conceptual data models has been fulfilled by the SM. The required aspect models of DEMO and their sequence are shown in Figure 4-21 below. As several processes need to be performed to create conceptual data models, the modeling process in DEMO from the specification of organization's requirements to conceptual data model requires a significant amount of time.

Even though it takes some time, the modeling process in DEMO is considered not too complicated. In DEMO, the organization's requirements are considered represented in a straightforward format that makes them easy to understand. From the transaction types listed in the organization's requirements specification, the result of the transactions can immediately be identified. Based on that information the TRT and IAM can be created, followed by the PSD, ARS, and then the OFD and OPL. In addition, the organization's requirements help defining the scope or boundary of the organization, which later on will be depicted in the ATD. By defining the scope of organization, only the relevant parts of organization are taken into account. The sequence confirms that there is a flow in the modeling process. A sequence in a modeling process is possible when the representation of each aspect model accommodates the representation of the other aspect model, which can be the previous or the next one. In addition, it can work effectively when each aspect model has a convenient representation, which means it is compact and readily understandable form. Thus, we can say that DEMO (graphical) representation is considerably a convenient one.

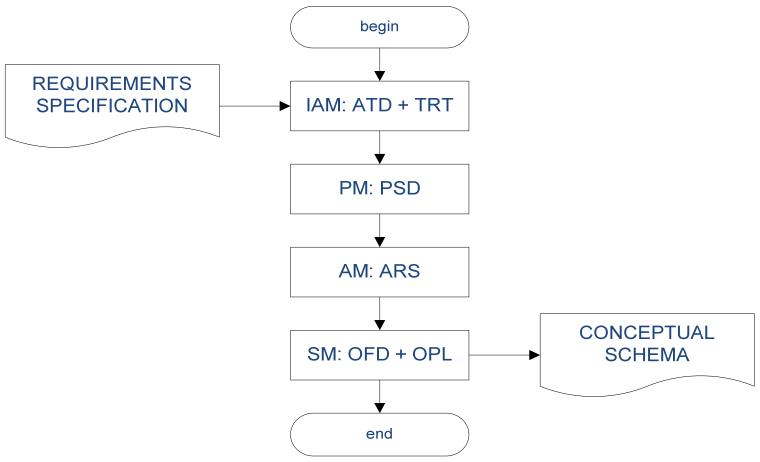


Figure 4-21 The sequence of DEMO aspect model for fulfilling the requirements

After analyzing DEMO against the formulated requirements, it is found that all of the requirements for specifying the semantics of data message can be satisfied by DEMO, even though an additional step is necessary to be performed to fulfill the Requirement 6. Thus, it can be stated that it is possible to use DEMO for specifying the semantics of data message. This implies that the second research question of this master thesis project has been answered, as well as the second output of this master thesis project has been produced. However, when analyzing DEMO against the Requirement 4, it is found that DEMO still lacks the specification for formulating constraints. This indicates that the SM of DEMO needs to be combined and/or modified, particularly for the purpose of improving the constraint specifications. Thus, another notation or methodology is required to extend the SM of DEMO. The ORM is selected for that purpose; it will be explained in detail in Chapter 5.

From analyzing DEMO against the Requirement 6, it is favored to depict all of the information items of the SM in one schema. Thus, in addition to the object types, the property types are depicted as well in OFD. It is because the classification or grouping of the elements of organization is considerably easier to be performed and understood when all of the information items are exhibited in one schema. In case the property types are listed in OPL and some derivational rules are specified, only the relevant property types and derivation rules that are considered representing the semantics of data message are exhibited.

Chapter 5
ORM

Object Role Modeling (ORM) is fact oriented modeling language which is commonly used to model database conceptual schema. In this master thesis project, ORM will be used to extend OFD, particularly to improve the constraints specification. ORM is selected for this purpose because of the similarity of the graphic representation with the SM, due to the fact that the diagramming technique of the OFD is based on ORM [Dietz, 2006]. In addition, ORM is considered as a method with complete notations that makes it expressive, including for specifying constraints.

An introduction to ORM will be provided in section 5.1. Several constraints specification available in the ORM that can be used to extend OFD is explained in section 5.2. In section 5.3, the conclusion of this chapter is provided.

5.1 Introduction to ORM

ORM is a fact-oriented modeling approach for modeling business domain information in terms of the underlying facts of interest, where all facts and rules may be verbalized in language readily understandable by non-technical users of those business domains [Halpin, 2009]. It defines graphical and textual languages that can be used in the modeling process. According to Halpin, for information modeling, fact-oriented graphical notations are typically far more expressive than those provided by other notations, such as UML and ER. The rich graphical notation makes it easier to detect and express constraints, and to visually transform schemas into equivalent alternatives [Halpin, 2009]. In addition, Halpin states that fact-oriented textual languages are based on formal subsets of native languages, so are easier to understand by business people than technical languages like UML's Object Constraints Language (OCL).

In the ORM, attributes are not used as a base construct [Halpin, 2009]. Instead, all fact structures are expressed as fact types (relationship types); these may be unary (e.g., Person smokes), binary (e.g., Person was born on Date), ternary (e.g., Person visited Country in Year), and so on. Avoiding attributes has several advantages [Halpin, 2009]:

- *Semantic stability*. It minimizes the impact of change caused by the need to record something about an attribute. As stated by Teorey, ORM avoids semantic instability by always using relationships instead of attributes.
- *Natural verbalization*. All facts and rules may be easily verbalized in sentences understandable to the domain expert

- *Populatability*. Sample fact populations may be conveniently provided in fact tables
- *Null-avoidance*. No nulls occur in populations of base fact types, which must be elementary or existential

This implies that the attribute free-nature designates facilitating communication with all stakeholders.

A fact type results from applying a logical predicate to a sequence of one or more object types [Halpin, 2009]. Each predicate comprises a named sequence of one or more roles (parts played in the relationship). A predicate itself is a sentence with object holes, one for each role, with each role depicted as a box and played by exactly one object type. Symbol 6, 7 or 8, and 9 in Figure 5-1 depict the unary, binary, and ternary predicate respectively; predicates of higher arity (number of roles) are allowed. The list of the main graphical symbols of the ORM is depicted in Figure 5-1. The detailed information of the symbols can be found in Table A-1 in Appendix A.

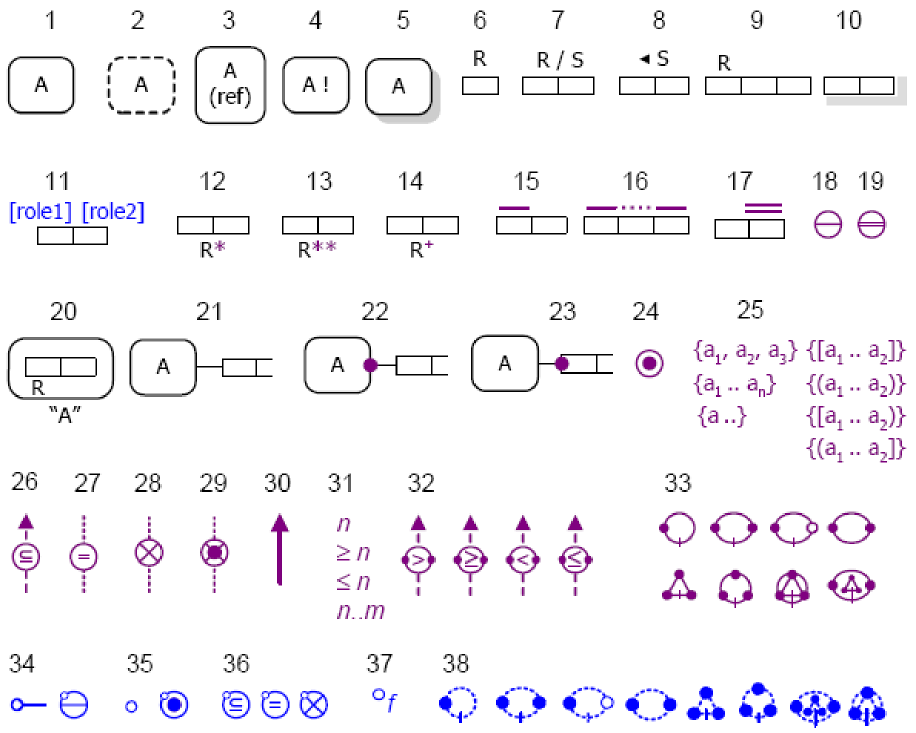


Figure 5-1 Main ORM graphic symbols [Halpin, 2009]

Symbols in violet represent necessary or hard constraints, meaning no violation is permitted. They are called *alethic* constraints. On the other hand, the blue symbols represent *deontic* rules, which means soft. For soft constraints violations are accepted but other action is taken to minimize their occurrence. As compared to the alethic symbols, the diotic symbols add an "o", or soften lines to dashed lines.

5.2 Constraints Specification in ORM

As described in the ORM notations in Table A-1, ORM is rich of graphical notations particularly for expressing constraints. Several additional constraints of the ORM are considered suitable for extending the SM of DEMO. Each of the suitable constraints will be explained in the following sub-sections.

5.2.1 Reference Scheme

ORM classifies object types into *entity types* (non-lexical objects) and *value types* (lexical objects), as illustrated respectively by symbol 1 and 2 in Figure 5-1. According to Teorey, each entity type needs to be identified by a *reference scheme*, due to entity types may be referenced in different ways, and typically change their state over time [Teorey et al, 2009]. In addition, Teorey states that value types, on the other hand, are constants (e.g., character string), and basically denote themselves, so they do not require a reference scheme to be declared. For analysis and validation purposes, we need to ensure that humans have a way to identify objects in their normal communication [Teorey et al, 2009]. Teorey explained that for conceptual analysis, such human-oriented reference schemes must be supplied; a value-based identification scheme is required for use by humans in communicating about the objects. Examples of human-oriented reference schemes are the employee numbers, car registration numbers, etc.

OFD of the SM does not distinguish object classes into entity types and value types as the ORM does. Yet, the SM defines property types, which are listed in OPL. Property types are commonly constants and they usually represent the attributes of the object classes. However, only the property types that embody the fact types of the ontological world (the C-world and P-world) are listed in OPL. None of those properties are apparently suitable as reference schemas. Therefore, for communication purpose, it is considered that the reference schema of object classes is necessary to be added in conceptual data models. By analyzing the available documentation of the system, the information that can be used as the reference schema of object classes should be able to be provided. This implies that reference scheme can be applied in the OFD.

Reference schemes may be abbreviated by enclosing the reference mode in parentheses as shown by symbol 3 in Figure 5-1. It denotes that the reference scheme is *implicitly* specified. An example of implicitly specified reference scheme is illustrated in Figure 5-2 (b). According to Teorey, if an entity type has more that one candidate reference scheme, one may be declared *preferred* to assist verbalization of instances (or to reflect the actual business practice). A preferred reference scheme is depicted by using the preferred uniqueness constraint, as shown by symbol 17 in Figure 5-1. A *preferred reference scheme* for an entity type maps each instance of it onto a unique, identifying value (or a combination of values) [Teorey et al, 2009]. An example of *explicitly* specified reference scheme is illustrated in Figure 5-2 (a).

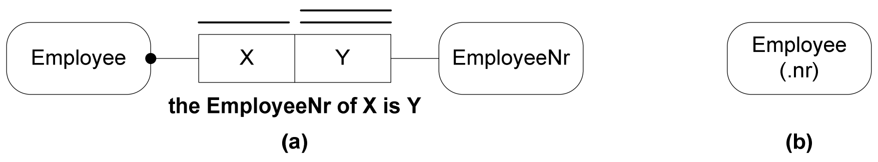


Figure 5-2 A reference scheme in the OFD, shown (a) explicitly and (b) implicitly

5.2.2 (Preferred) External Uniqueness Constraint

External uniqueness constraint implies that instances of the role combination in the join of those predicates are unique. It may be applied to two or more roles from different predicates by connecting them with dotted lines. The notation is represented by symbol 18 in Figure 5-1, and the preferred external uniqueness constraint is represented by symbol 19; it is shown as a circled uniqueness bar. To clarify this concept an example taken from Halpin is illustrated in Figure 5-3, if a state is identified by combining its state code and country, an external uniqueness constraint is added to the roles played by `Statecode` and `Country`.

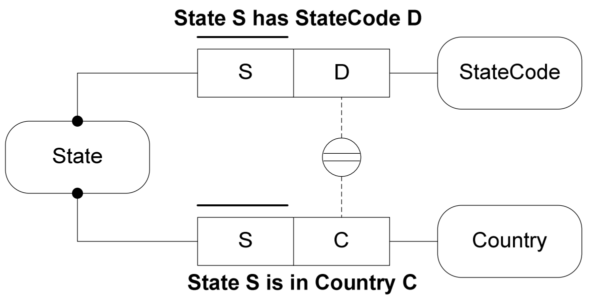
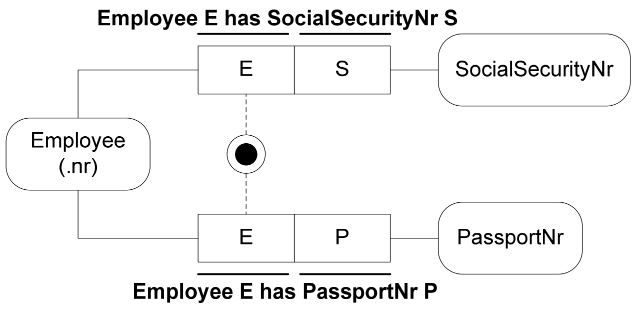


Figure 5-3 Example of external uniqueness constraint

5.2.3 *Inclusive-or Constraint*

The inclusive-or (disjunctive mandatory) constraint is a constraint that is applied to two or more roles. It states that all instances of the object type population must play at least one of those roles. An inclusive-or constraint is represented by connecting the roles by dotted lines to a circled dot as seen in symbol 24 in Figure 5-1. Teorey describes an example of inclusive-or as illustrated in Figure 5-4 as follow: an employee has a social security number or passport number.


Employee E has SocialSecurityNr S
E
S
SocialSecurityNr
Employee (.nr)
E
P
PassportNr
Employee E has PassportNr P
Figure 5-4 Example of inclusive-or constraint

5.2.4 *Value constraints*

Value constraints are specified to restrict the population of an object type or role [Halpin, 2009]. As explained by Halpin, the population can be specified to a finite set of values either in full (enumeration), by start and end values (range), or some combination of both (mixture). The values themselves are primitive data values, typically character strings or numbers. The constraint is shown by declaring the possible values in braces besides either the value type, or an entity type with a reference mode. An ordered range may be declared separating end values by "…", meanwhile, for continuous ranges a square/round bracket indicates an end value is included/excluded, for example, "`{(0…10]}`" denotes the positive real numbers up to 10.

There are several cases in which ORM does not provide the specification, such as initial/default value, ordered set specification, etc. In case another specification needs to be defined but can not be accommodated by the three types of specification mentioned earlier, it is possible to use textual constraint, for example: `comitteeSize must be an odd number`. The notation of the value constraint can be seen in symbol 25 of Figure 5-1. Examples of value constraint are shown in Figure 5-5.

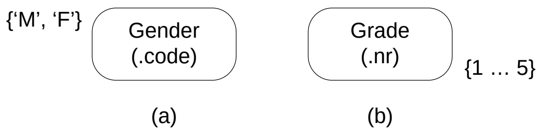


Figure 5-5 Examples of value constraint (a) enumeration (b) range

As aforementioned, value constraints can also be applied to roles. An example on this case is shown in Figure 5-6. The example defines value constraints on the `minimumMultiplicity` and `maximumMultiplicity`.

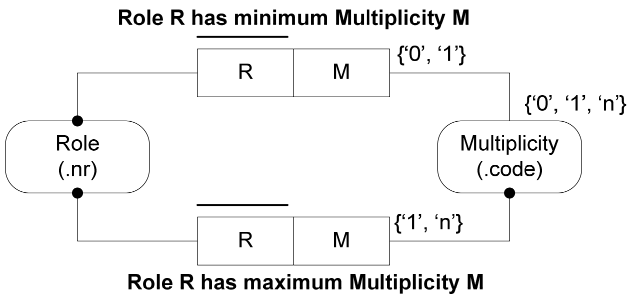


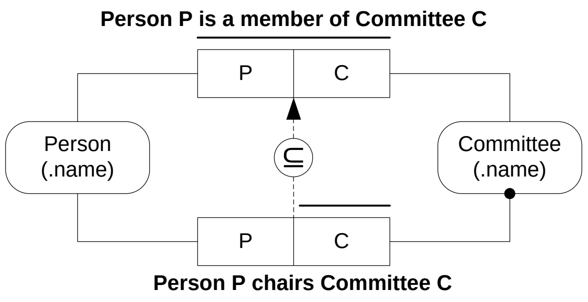
Figure 5-6 Example of value constraint on roles [Halpin, 2005]

5.2.5 Set Comparison Constraints

This type of constraints declares a subset, equality, or exclusion relationship between the populations of role sequences. The exclusion constraint of the ORM is comparable to the exclusion law of DEMO that has been explained in the previous chapter, thus it is not necessary to mention it again here.

5.2.5.1 Subset Constraint

Subset constraint restricts the population of the first sequence to be a subset of the second. It is represented as a dotted arrow with a circled subset symbol as shown in symbol 26 in Figure 5-1. An example taken from Teorey is provided in Figure 5-7, the subset constraint indicates that any person who chairs a committee must be a member of that committee.


Person P is a member of Committee C
P
C
Person
(.name)
⊆
Committee
(.name)
P
C
Person P chairs Committee C
Figure 5-7 Example of Subset constraint

Subset constraint can also be used to restrict the population of a role to be the subset of the population of another compatible role. For instance, all citizens are residents (not necessarily of the same country) can be depicted as seen in Figure 5-8.

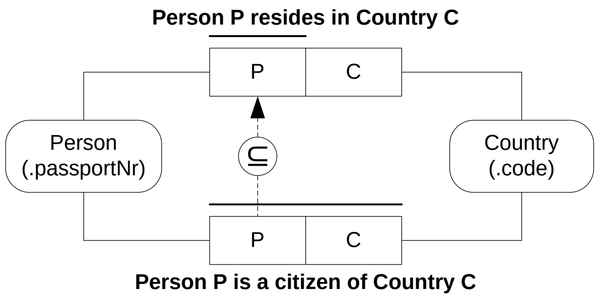

Person P resides in Country C
P
C
Person
(.passportNr)
⊆
Country
(.code)
P
C
Person P is a citizen of Country C

Figure 5-8 A single role subset constraint

5.2.5.2 Equality Constraint

An equality constraint indicates that the population of the role sequences must always be equal. An equality constraint between two compatible role sequences is shorthand for two subset constraints (one in either direction) [Teorey et al, 2009]. It is depicted as a dotted line with a circled "=" symbol. Teorey stated if two roles played by an object type are mandatory, then an equality constraint between them is implied (and therefore not shown). An adjusted example of equality constraint taken from Teorey is provided in Figure 5-9. As depicted in the figure, if a patient's systolic blood pressure is measured, so is his or her diastolic blood pressure (and vice versa); in other words, either both measurements are taken or neither.

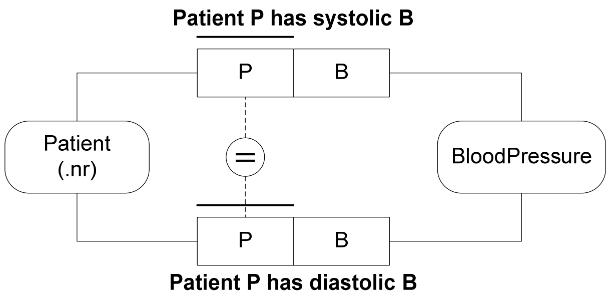


Figure 5-9 Example of equality constraint

5.2.6 *Exclusive-or Constraint*

An exclusive-or constraint indicates that each instance of a class plays exactly one role from a specified set of alternatives. The exclusion-or constraint is simply an orthogonal combination of a disjunctive mandatory role (inclusive-or) constraint (circled dot) and an exclusion constraint (circled "X"), the notation can be seen in symbol 29 of Figure 5-1. An example of exclusive-or taken from Teorey is illustrated in Figure 5-10. The figure indicates that each account is used by a person or corporation but not both.

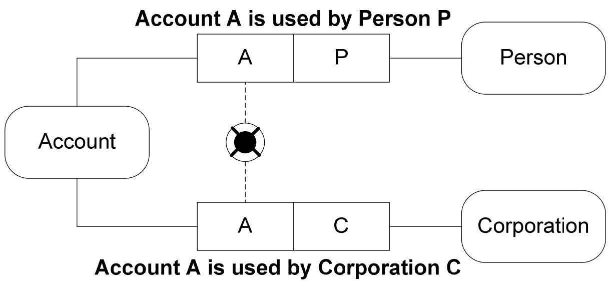

Account A is used by Person P
A
P
Person
Account
A
C
Corporation
Account A is used by Corporation C

Figure 5-10 Example of exclusive-or constraint

5.2.7 Frequency Constraint

Frequency constraint is applied to a role sequence. The notation of this constraint is depicted in symbol 31 of Figure 5-1. This constraint indicates that instances that play those roles must do:

- solely *exactly n* times (*n*)
- *at least n* times (≥ *n*)
- *at most n* times (≤ n)
- *at least n and at most m* times (n … m)

An adjusted example of frequency constraint taken from Teorey is illustrated in Figure 5-11. Suppose that we wish to record the nicknames and colors of country flags and we agree to record at most two nicknames for any given flag and that nicknames apply to only one flag. The extended OFD model of this example is shown in Figure 5-11.

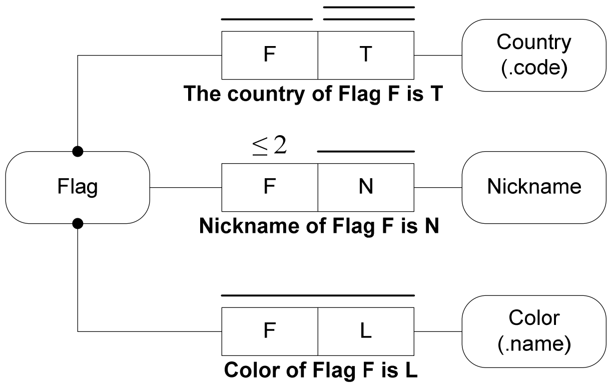


Figure 5-11 Example of frequency constraint

As shown in Figure 5-11, Flag object type has a reference scheme Country, which explicitly represented. The ≤ 2 frequency constraint indicates that each flag has at most two nicknames, and the unicity law applied to Nicknames implies that each nickname refers to at most one flag.

5.2.8 Ring constraints

As mentioned earlier, ring relationship signifies a binary relationship that relates an entity type to itself. It means that a ring fact type at least has two roles which are played by the same entity type. A ring constraint applies a logical restriction to such role pairs [Halpin, 1999]. As shown in symbol 33, Figure 5-1, there are six types of built-in ring constraints: *irreflexive*, *asymmetric*, *antisymmetric*, *symmetric*, *intransitive*, and *acyclic*; as well as two combination constraints: *intransitive and acyclic*, and *intransitive and asymmetric* (other combinations are possible). However, there is a rule that should be complied with when combining or using ring constraints. Halpin has defined the ring-constraint interface as shown in Figure 5-12, so that we can not enter a ring constraint that is implied by, or incompatible with, one that we already chosen.

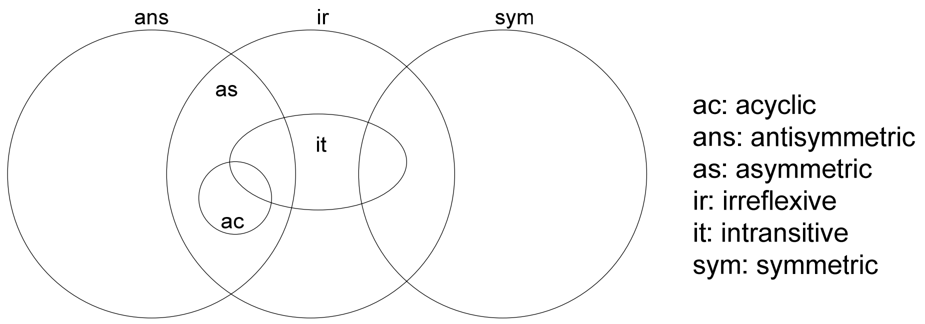
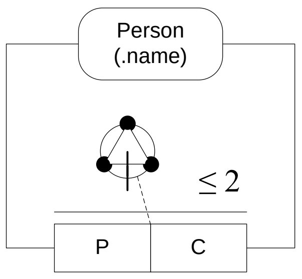


Figure 5-12 Ring constraint interface [Halpin, 1999]

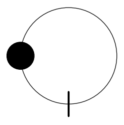
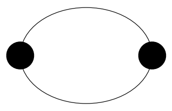
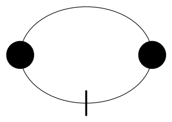
An example of ring constraint implementation is illustrated in Figure 5-13. The description of ring constraints as shown in Figure 5-1 is explained in Table 5-1 below.

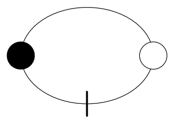
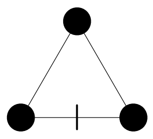
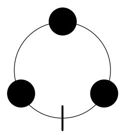
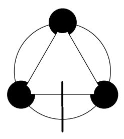
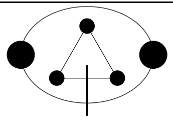


Person P is a parent of Person C

Figure 5-13 Example of ring constraint

Table 5-1 Ring constraints description

No	Name	Symbol	Description
1.	irreflexive (ir)		Irreflexive means that the two roles cannot be filled by the same instance of an object [Murphy, 2009]. For instance, the descendent of a person can not be the person itself.
2.	symmetric (sym)		Symmetric constraint means if a relationship exists between two instances of an object, then the same relationship exists in the opposite direction [Murphy, 2009], e.g., if Ohio borders Indiana, then Indiana borders Ohio.
3.	asymmetric (as)		Assymetric constraint is mostly the opposite of symmetric. If a relationship exists between two instances of an object, then the same relationship does not exist in the opposite direction [Murphy, 2009]. A person is a parent of another person is asymmetric. Note that asymmetry implies irreflexivity.

No	Name	Symbol	Description
4.	antisymmetric (ans)		Antisymmetric constraint is kind of like symmetric with asymmetry in specific instances. So in the antisymmetric relationship, if the same instance of an object is playing both roles of a fact, then it is symmetric, but if different instances of the object are playing the roles of a fact, then it is asymmetric [Murphy, 2009]. Here is an example of an antisymmetric fact type: Number is greater than or equal to Number, it is symmetric as long as the numbers are the same, otherwise it is asymmetric [Murphy, 2009].
5.	intransitive (it)		This type of constraint has to do with chaining relationships together [Murphy, 2009]. An intransitive constraint means we can not add any arrows that jump over one node to provide an alternate path to the target node. As illustrated in Figure 5-13, the intransitive constraint means nobody is a parent of any of his/her grandchildren. Note that intransitivity implies irreflexivity.
6.	acyclic (ac)		An acyclic constraint means there can not be any cycles or loops in the graph. As illustrated in Figure 5-13, the acyclic constraint means that nobody can be one of his/her own descendants. Note that acyclic implies asymetric which implies irreflexive.
7.	intransitive and acyclic		This constraint represents the combination of intransitive and acyclic constraints.
8.	intransitive and asymmetric		This constraint represents the combination of intransitive and asymmetric.

5.3 Conclusion

In order to satisfy all of the requirements for specifying the semantics of data message, the SM of DEMO needs to be extended. The ORM has been selected for this purpose, especially due to the similarity of the graphic representation. In addition, the ORM is considerably an expressive method, with complete notations, including for specifying constraints required for extending the SM. The items in the ORM that can be used to extend the constraint aspect in order to completely fulfill the Requirement 4 are:

1. Reference scheme
2. External uniqueness constraint

3. Inclusive-or constraint
4. Value constraint
5. Set comparison constraint
6. Exclusive constraint
7. Frequency constraint
8. Ring constraint

The constraints extension as specified based on the ORM conclude the answer of the research question number two. Thus, the DEMO specification in fulfilling the requirements can be completed.

Chapter 6
Case Study

In order to understand how to use DEMO for specifying the semantics of data message, in this chapter the semantics of the data messages of a case study will be specified in DEMO, by complying with the requirements for specifying the semantics of data message. The selected case study is a standard developed by SETU, used for ordering and selecting a temporary worker, which is called the *Standard for Ordering and Selection*. SETU is a Dutch acronym stand for "*Stichting Elektronische Transacties Uitzendbranche*", which is translated as "Organization for Electronic Transactions in the Staffing Industry". It is a non-profit organization that creates and maintains standards for exchange of electronic data in Dutch Staffing Industry. An introduction to the case is provided in section 6.1. Analysis to the case in fulfilling the requirements is performed in section 6.2. The conclusion of this chapter can be found in section 6.3.

6.1 Introduction to the Case

The summary of the SETU Standard for Ordering and Selection is provided in this section, the detailed information of the standard can be found in [Enting et al., 2008].

The SETU Standard for Ordering and Selection deals with electronically sending information related to ordering and selecting a temporary worker for an open position. It represents the first step on hiring a temporary worker. As shown in Figure 6-1, the involved actors in the process are the Staffing Company and Staffing Customer; the parties may take different roles as listed in Table 6-1.

Staffing Company
Ordering and Selection
Staffing Customer

Figure 6-1 Parties for ordering and selection [Enting et al., 2008]

A Staffing Customer can notify a Staffing Company for an open position. This indicates that the Staffing Customer has requested the Staffing Company to provide employee(s) to fill in the open position. Afterward, a Staffing Company can provide an offer as the response to the request of the Staffing Customer, or even reject it if the Staffing Company is unable to fulfill the request. Upon receiving an offer from a Staffing Company, the Staffing Customer can respond with a reservation, acceptance or rejection. In addition, an order of a temporary worker can also be sent by the Staffing

Customer to the Staffing Company. In receiving an order from a Staffing Customer, the Staffing Company has to fulfill it; a declination must not be performed.

Table 6-1 Parties Roles for ordering and selection [Enting et al., 2008]

Party	Role	Description
Staffing Company	Supplier	The Staffing Company acts as the supplier of (temporary) workers. After receiving an open position, it starts looking for a suitable worker and offers his worker to the consumer. The Staffing Company may also receive an order from a procurement system.
	Consumer	In some cases the Staffing Company acts as a consumer of (temporary) workers. This may occur if the Staffing Company has to supply a worker that is not enlisted at the Staffing Company and the Staffing Company orders a (temporary) worker to another staffing company.
Staffing Customer	Consumer	The Staffing Customer acts as a consumer of (temporary) worker. After sending an open position to one or more suppliers, it waits for the suppliers to offer temporary worker. The staffing customer can also place an order using a procurement system.

The complete events supported in the Standard for Ordering and Selection are mentioned below[3].

1. `CreatePosition`: the consumer has an open position and creates a message to indicate this to the supplier. The supplier is expected to respond with an offer.
2. `UpdatePosition`: the open position at the consumer changes and the consumer sends a message to update the information at the supplier. The supplier is expected to respond with an offer of an updated offer.
3. `CreateOrder`: the consumer creates an order and sends this order to the supplier.
4. `ClosePosition`: the position at the consumer is fulfilled, or does not need to be fulfilled anymore; the consumer sends a close message to the supplier. The supplier is not expected to respond. The `ClosePosition` message only indicates that the supplier is not expected to respond anymore, the position is not archived. The position can be reopened after sending the `ClosePosition` message.
5. `ReopenPosition`: the position was closed, but needs to be fulfilled again; the supplier is notified of this change. The supplier is expected to respond with an offer.
6. `ArchivePosition`: the position is not open anymore and the position is archived. The supplier is notified of this change and is not expected to respond. This message terminates the position; the position can not be reopened. If the position becomes available again a new `CreatePosition` message has to be sent.
7. `RejectPosition`: the supplier has received a position and rejects this position. The consumer is not expected to respond.
8. `CreateOffer`: the supplier has received a position and responds to this with an offer. The consumer is expected to respond with a reservation, acceptance or rejection.
9. `UpdateOffer`: the supplier has received an updated position, or the information in the offer has changed, the supplier sends an updated offer to the consumer. The consumer is expected to respond with a reservation, acceptance, or rejection.

[3] The complete description of the events can be found in Appendix B.1

10. `ReserveOffer`: the consumer has received an offer, and places a reservation on the offer. The supplier is not expected to respond.
11. `AcceptOffer`: the consumer has received an offer, may have placed a reservation, and accepts the offer. The supplier is not expected to respond.
12. `WithdrawOffer`: the supplier withdraws the offer because the person offered is not available anymore. The consumer is not expected to respond.
13. `RejectOffer`: the consumer rejects the offer and notifies the supplier of this rejection. The supplier is expected to respond with an update or new offer.

For each event, a message is sent from the Staffing Company to the Staffing Customer, and vice versa. The events and the respective message are listed in Table 6-2. All together, four types of message are specified in the Standard for Ordering and Selection: `Position`, `PositionStatus`, `Offer`, and `OfferStatus`. The definitions and description of the messages are exhibited in Appendix B (B.2, B.3, B.4, and B.5 respectively).

Table 6-2 Message per event [Enting et al., 2008]

No	Event	Message	Status
1.	`CreatePosition`	Position	New
2.	`UpdatePosition`	Position	Revised
3.	`CreateOrder`	Position	New
4.	`ClosePosition`	PositionStatus	Closed
5.	`ReopenPosition`	PositionStatus	Reopened
6.	`ArchivePosition`	PositionStatus	Cancelled
7.	`RejectPosition`	PositionStatus	x:Rejected
8.	`CreateOffer`	Offer	New
9.	`UpdateOffer`	Offer	Revised
10.	`ReserveOffer`	OfferStatus	Pending
11.	`AcceptOffer`	OfferStatus	Accepted
12.	`WithdrawOffer`	OfferStatus	Withdrawn
13.	`RejectOffer`	OfferStatus	Rejected

6.2 Case Analysis

In the case analysis, the semantics of the four types of messages in the Ordering and Selection will be specified. Those four message types are the `Position`, `PositionStatus`, `Offer`, and `OfferStatus` messages. The analysis will be performed by acting upon the specified requirements for specifying the semantics of data message. The Staffing Company is selected to be the organization under consideration.

6.2.1 Implementation of the Requirement 1

In order to fulfill the Requirement 1 for specifying the semantics of data message, the organization's requirements need to be identified. In DEMO, the organization's requirements are represented in a list of the identified transaction types. DEMO specifies a method for the requirements analysis that consists of:

1. *The Performa-Informa-Forma Analysis*
2. *The Coordination-Actors-Production Analysis*
3. *The Transaction Type Identification*

The implementation of the step 1 and step 2 of the method can be found in Appendix C. The analysis to cluster the C-acts/results and P-acts/results that have been identified from the step 1 and 2 is required to be performed to identify the transaction types as the result of the step 3. In general, the identification is performed by analyzing the description of the case, including the supported events in the Standard for Ordering and Selection. These events represent the implementation level that also covers the message exchanges in the Ordering and Selection. Only the transactions that happen within the boundary of the Staffing Company need to be considered. The identified transaction types represent the ontological actions of organization, which are abstracted from the implementation or operational level of organization.

In the Standard for Ordering and Selection, two transactions are identified between the Staffing Customer and Staffing Company. In the two transactions, the Staffing Customer becomes the initiator and the Staffing Company becomes the executor. The first transaction represents the fulfillment of the open position in the Staffing Customer. It covers the offer preparation by the Staffing Company, followed by the offer selection by the Staffing Customer. This transaction is completed when the Staffing Customer accept the offer of employee from the Staffing Company to fill in the open position. Further on, this transaction is identified as the transaction T01. The second transaction represents the completion of an order from the Staffing Customer by the Staffing Company. Actually, an order also represents an open position. However, while performing this transaction, the Staffing Customer does not expect any offer from the Staffing Company. It is because, when performing an order, the Staffing Customer already had the information regarding the offer from the Staffing Company. It means that an offer has been created earlier. An order can be regarded as the Staffing Customer desire to have possession to the employee provided in the offer. It is considered that this transaction is completed when the Staffing Customer accept a statement from the Staffing Company that the order has been confirmed. Further on, this transaction is identified as the transaction T03. The transaction T03 denotes the completion of an order by the Staffing Company. Therefore, this transaction is considered suitable for representing the `CreateOrder` event.

To clarify the correlation between the available events in the standard and the transaction T01, each event will be analyzed based on the transaction axiom. The `CreatePosition` event is considered suitable for representing the request act of the transaction T01 (T01/rq). Performing the `UpdatePosition` event means that the request (T01/rq) is cancelled by the Staffing Customer, which is followed by allowing this by the Staffing Company and then quitting the transaction T01, and after that the Staffing Customer immediately starts a new one by performing the request act (T01/rq) again. The `ReopenPosition` event is considered initiating the same actions as performed in the `UpdatePosition` event. First, the request has to be cancelled, and then new request can be performed again. The difference is in the `ReopenPosition` event the cancellation of the request will represent the `ClosePosition` event. Therefore, we can say that the three events (`CreatePosition`, `UpdatePosition`, and `ReopenPosition`) initiate the fulfillment of the open position in the transaction T01. Therefore, the same ontological actions might be performed when these events are executed.

Performing the `ClosePosition` event means that the Staffing Company is expected to stop responding; it can happen as well when the open position has been fulfilled. Therefore, in addition to the cancellation of a request, `ClosePosition` can also represent the acceptance act of the transaction T01 (T01/ac). It seems that, in performing `ArchivePosition`, the Staffing Company is expected to respond the same way as in the `ClosePosition` event, which is to stop responding. Therefore `ArchivePosition` can be considered representing the cancellation of a request and also the acceptance of the transaction. We can say that the `ClosePosition` event similar with `ArchivePosition` event since the same ontological actions are performed.

In DEMO, the pattern of a **cancellation of a request** is illustrated as shown in Figure 6-2. The cancel act by the Staffing Customer brings the process in the discussion state cancelled. The cancellation can be allowed or even refused by the Staffing Company. When allowed is reached, the Staffing Customer can quit the transaction, if not requested will remain the case.

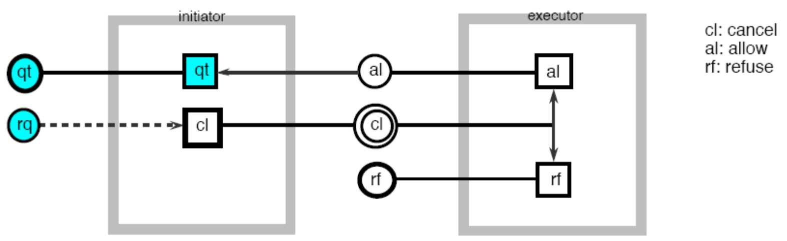


Figure 6-2 Cancellation pattern of a request [Dietz, 2006]

Upon receiving a request from the Staffing Customer, the Staffing Company can select to promise (T01/pm) or even decline the request (T01/dc). A decline represents a condition in which the promise is cancelled, and it is allowed by the Staffing Customer. Figure 6-3 exhibits the pattern of a **cancellation of a promise**. As shown in the figure, an allowed cancellation of a promise will cause the transaction in the discussion state declined. A decline of the transaction T01 is considered represented by the `RejectPosition` event since it represents the refusal of the Staffing Company to provide employee for filling in the open position.

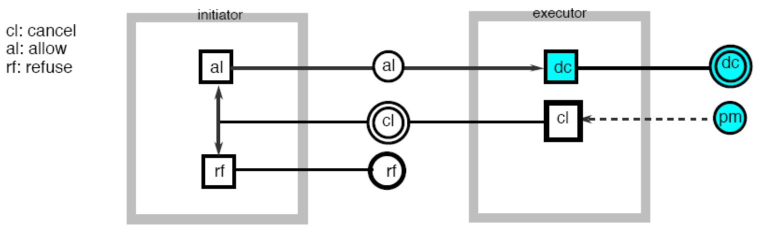


Figure 6-3 Cancellation pattern of a promise [Dietz, 2006]

The production act of the transaction T01 (T01/ex) is denoted by the preparation of an offer by the Staffing Company. It is considered that the `CreateOffer` event is suitable for representing the preparation of the offer by the Staffing Company to fulfill the open position. After that, the Staffing Company can perform the state action (T01/st) to indicate the offer to the Staffing Customer. It is possible for the Staffing Company to cancel its statement. The pattern of a **cancellation of a statement** is shown in Figure 6-4. The cancel act by the Staffing Company brings the process in the discussion state cancelled. The cancellation can be allowed or even refused by the Staffing Customer. The `WithdrawOffer` event is considered suitable to represent the cancellation of a statement that is allowed by the Staffing Company. The `UpdateOffer` event indicates that the statement is cancelled by the Staffing Company, which is then allowed by the Staffing Customer, and after that the Staffing Company redo the production act again.

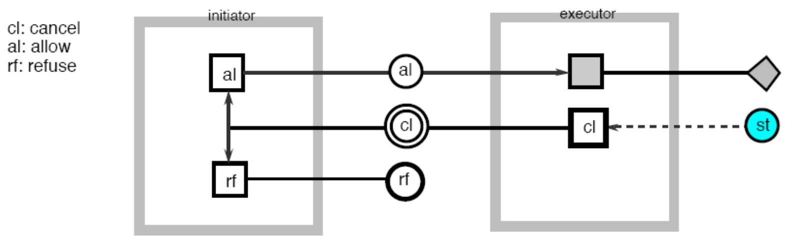


Figure 6-4 Cancellation pattern of a statement [Dietz, 2006]

If the Staffing Customer considers that the offer from the Staffing Company can satisfy the open position, the Staffing Customer can perform an acceptance act (T01/ac). In addition to the `ClosePosition` and `ArchivePosition` events, apparently the `ReserveOffer` and `AcceptOffer` are suitable as well to represent the acceptance act. Besides acceptance, the Staffing Customer can also reject the offer (T01/rj). A rejection represents the cancellation of an acceptance. The pattern of a **cancellation of an acceptance** is exhibited in Figure 6-5. An allowed cancellation to an acceptance will cause the transaction in the discussion state rejected. It is considered that the `RejectOffer` event is suitable to represent the cancellation of an acceptance since this event indicates the rejection of the Staffing Customer to the offer.

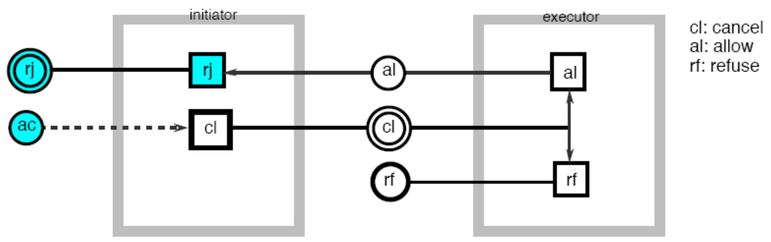


Figure 6-5 Cancellation pattern of an acceptance [Dietz, 2006]

A certain preparation may need to be performed by the Staffing Company when receiving a request to fill in the open position from the Staffing Customer. Thus, another transaction can be added to fulfill this purpose. Since the transaction happens inside the Staffing Company, the initiator and executor of the transaction will be the Staffing Company. This transaction is completed when the offer has been created. This transaction can be considered representing the `CreateOffer` event, which earlier was identified as the production act of the transaction T01. In addition, this transaction can also represent the `UpdateOffer` event, since updating an offer means that the request is cancelled and a new request to prepare an offer is performed again. This transaction is required to be the part of the transaction T01, since for each open position an offer is required to be prepared. Further on, this additional transaction is indicated as the transaction T02.

The identified transaction types are arranged and given identifier as listed in Table 6-3. The transaction types listed in the following table represents the fulfillment of the Requirement 1 for specifying the semantics of data message.

Table 6-3 Transaction types

Transaction ID	Transaction Type
T01	open position fulfillment
T02	offer creation
T03	order completion

6.2.2 *Implementation of the Requirement 2, 3, 4, 5*

How to use DEMO to fulfill the requirements related to the conceptual data modeling (i.e., Requirement 2, 3, 4, 5) are shown in this section. The complete process of creating the SM of the case will be illustrated here.

6.2.2.1 The Transaction Pattern Synthesis

In this step, the transaction types as listed in Table 6-3 will be completed by specifying the result of each transaction types. The list of the transaction types and their result is shown in Table 6-4 below.

Table 6-4 Transaction pattern

Transaction Type		Transaction Result	
T01	open position fulfillment	R01	open position T has been fulfilled
T02	offer creation	R02	offer O has been created
T03	order completion	R03	order R has been completed

6.2.2.2 The Result Structure Analysis

The dependencies between the transactions need to be identified. Dependencies exist if one or more transactions become the components of a certain transaction. This means the result of the components is a condition for bringing about the corresponding transaction result. For the ordering and selection the dependencies can be identified as follow.

T02 is part of T01. It means that the existence of the result R02 (*offer O has been created*) is a condition for bringing about the corresponding result R01 (*open position T has been fulfilled*). This indicates that the position fulfillment is considered to be completed as soon as new offer for the position is created. Consequently, R02 is regarded to be the component of R01; T02 is regarded to be enclosed in T01.

As stated in the description, the Staffing Company can also act as a consumer. This may occur only when the Staffing Company receives an order of a person that is not enlisted in the Staffing Company. When it happens, the Staffing Company needs to send order(s) to the other Staffing Company(s), since a received order needs to be fulfilled. Sending orders by the Staffing Company is not necessarily performed when receiving a T01 with a request of a person that is not enlisted, since in T01 it can be declined. On the other hand, in T03 when the requested person is not available the Staffing Company needs to send order(s) by initiating another T03 to the other staffing company(s). Thus, we can state that the existence of the result R03 (*order R has been completed*) may become the condition for bringing another R03. This indicates that an order completion may need another order completion to be performed. Consequently, R03 is regarded to be the component of R03; T03 is regarded to be enclosed in T03.

6.2.2.3 The Construction Synthesis

The actor role that is the initiator and the executor of every transaction type will be determined in this section. The list of the actors for Ordering and Selection can be found in Table 6-5. As seen in the table below, the actor role name "*order receiver*" is actually not longer appropriate if one considers that A03 becomes the initiator for T03; "*consumer*" might become a better actor role name. However, the naming should not be a big problem, since initiating T03 matches with the operation cycle of A03, as explained earlier.

Table 6-5 Construction synthesis

Transaction Type		Initiator		Executor	
T01	open position fulfillment	CA01	consumer	A01	supplier
T02	offer creation	A01	supplier	A02	offer creator
T03	order completion	CA01	consumer	A03	order receiver
		A03	order receiver	CA02	order receiver

6.2.2.4 The Organization Synthesis

In this section, the boundary of the organization of the enterprise under consideration is settled. The division of the actor roles and transaction types for the Ordering and Selection is listed in Table 6-6. It shows the internal and environmental actor roles, as well as the internal and interface transaction types. Interface transaction types are transaction types of which one of the participating actor roles belongs to the composition and the other belongs to the environment [Dietz, 2006].

Table 6-6 Organization synthesis

Internal actor roles	A01 A02 A03	supplier offer creator order receiver
Environmental actor roles	CA01 CA02	consumer order receiver
Internal transaction types	T02	offer creation
Interface transaction types	T01 T03	open position fulfillment order completion

6.2.2.5 The IAM: TRT and ATD

The TRT of the Standard for Ordering and Selection is taken from the transaction pattern synthesis; it is shown in Table 6-7.

Table 6-7 The TRT of the Standard for Ordering and Selection

Transaction Type		Transaction Result	
T01	open position fulfillment	R01	open position T has been fulfilled
T02	offer creation	R02	offer O has been created
T03	order completion	R03	order R has been completed

The interface transaction types as listed in Table 6-6 are used to draw the global ATD of the Ordering and Selection, as exhibited in Figure 6-6.

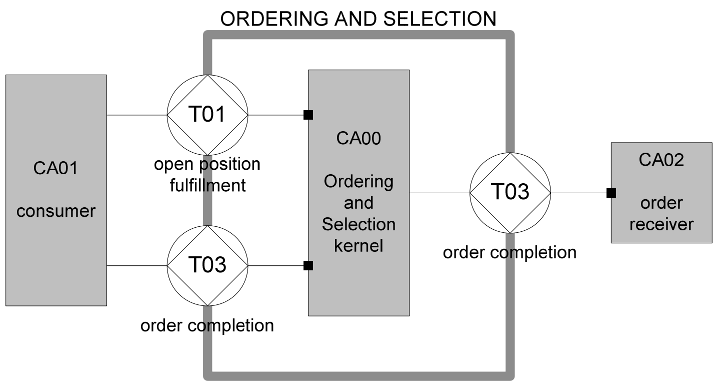


Figure 6-6 Global ATD of the ordering and selection

It is a convention to number the kernel CA00; however, the numbering of the other composite actor roles is arbitrary such as CA01, etc [Dietz, 2006]. The actor role CA01 (*consumer*) represents the Staffing Customer that has the open position and requests the Staffing Company (the kernel of ordering and selection) to provide an offer of employee(s) to fill in the open position. In addition, a consumer can also send an order to the Staffing Company. CA01 (*consumer*) becomes the initiator of the transaction type T01 (*open position fulfillment*) and transaction type T03 (*order completion*). The actor role CA02 (*order receiver*) represents another Staffing Company to which CA00 sends an order.

The actor role CA00 consists of three elementary actor roles: A01, A02, and A03. A01 is the executor of T01, and also the initiator of T02. A02 becomes the executor of T02. A03 is the executor of T03 and may also become the initiator of T03. The complete detailed ATD of the ordering and selection is shown in Figure 6-7.

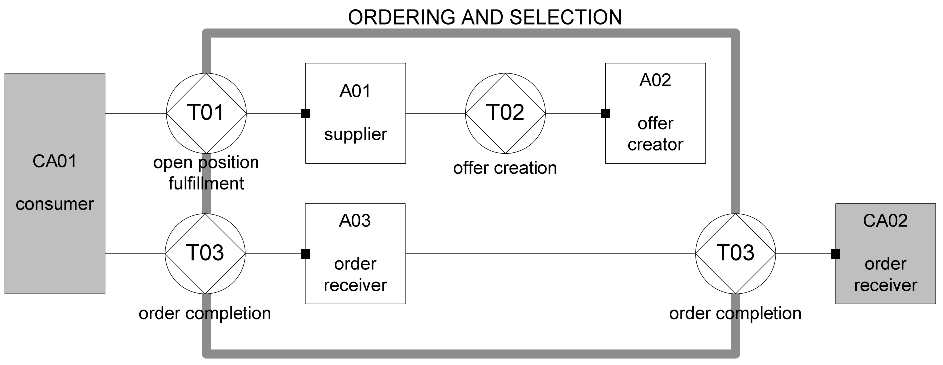


Figure 6-7 Complete detailed ATD of the ordering and selection

6.2.2.6 The PM

6.2.2.6.1 PSD of business process 1

Figure 6-8 exhibits the PSD for the actor role A01 and A02 that is indicated as business process 1. As shown in the PSD, transaction T02 is enclosed in a T01.

If a request (T01/rq) to fill in an open position is received, the responsible actor role (A01) checks whether the condition to continue the transaction is achieved. If the condition is not achieved, it is possible to perform a rejection or cancellation (T01/dc). Otherwise, a promise act (T01/pm) can be performed. If T01 is declined, the actor role CA01 can select to quit the transaction or perform another T01/rq. As soon as a

T01/pm is performed, in dealing with the result, A01 can perform these three acts: T01/cancel(pm), T02/rq, and T01/ex.

Firstly, A01 performs a cancellation to the promise by initiating T01/cancel(pm). If the cancellation is allowed, the transaction will be in the discussion state "declined", meaning that T01/dc is performed. If the cancellation is refused, "promised" remains to be the cased. The allowed cancellation of a promise also represents the `RejectPosition` event. The cancellation is optional, which is indicated by the cardinality range 0..1.

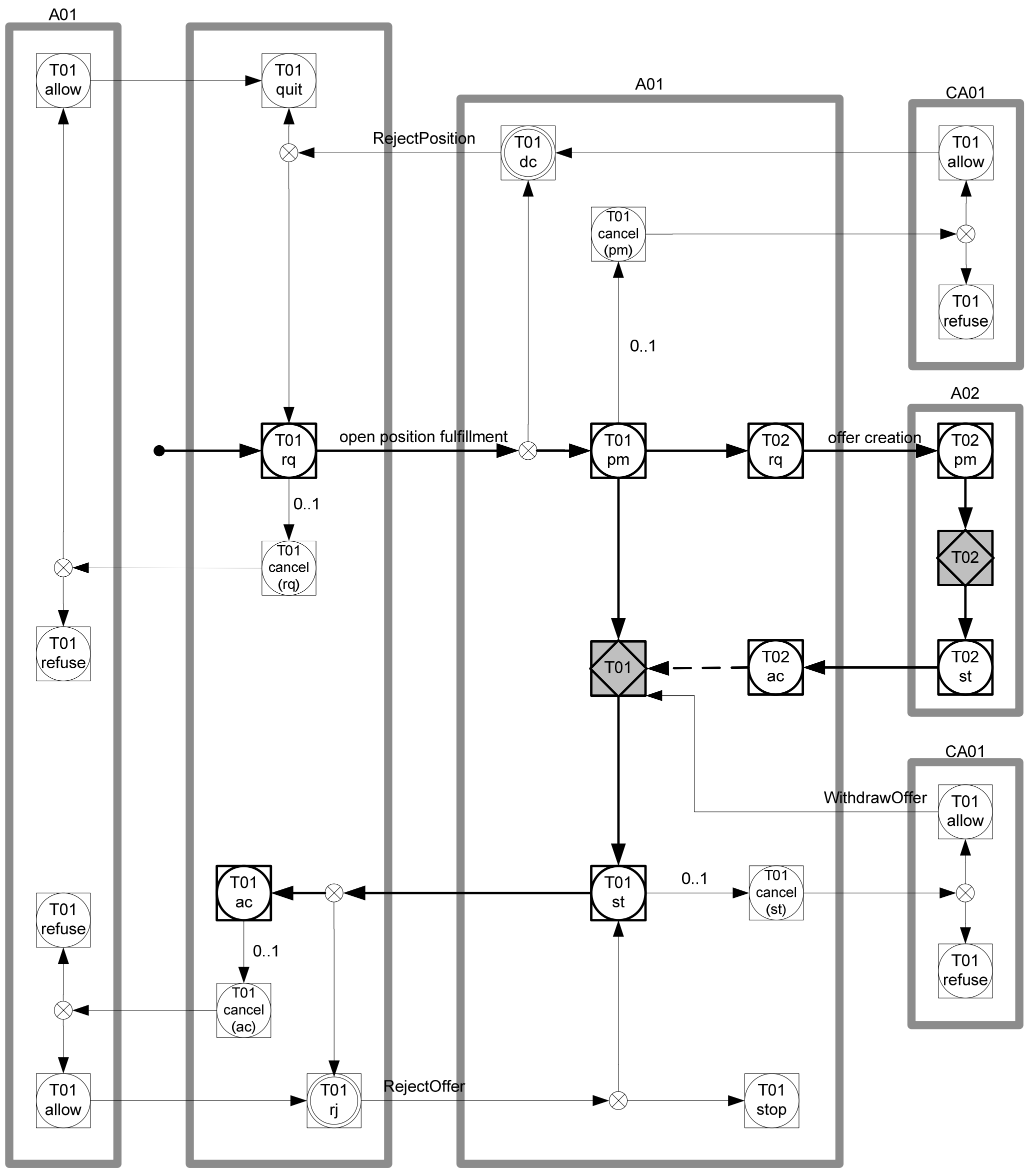


Figure 6-8 PSD of business process 1

Secondly, A01 performs T02/rq. In dealing with a request of an open position from the consumer, A01 needs to initiate a T02 (*offer creation*) to create an offer of employee(s).

Thirdly, A01 performs T01/ex. It appears that the result of T02 becomes the wait condition to perform T01/ex. It means that to perform T01/ex, it has to wait until a transaction T02 successfully completed, in which an offer has been prepared for the consumer. After the execution, A01 can perform a statement act (T01/st), for indicating the offer of employee to the consumer (CA01).

A01 can perform a cancellation to the statement by performing T01/cancel(st). CA01 can select to refuse or allow the cancellation. The cancellation of the statement is optional, which is indicated by the cardinality range 0..1. An allowed cancellation of statement is considered representing the `WithdrawOffer` event. Performing an allowed cancellation to the statement may cause A01 to perform another T01/ex. It means another T02 is required to be performed again.

In dealing with the result of T01/st, the actor role CA01 can respond by performing a T01/ac or T01/rj. An acceptance can also be cancelled, that will initiate a T01/cancel(ac). When it is allowed the transaction will be in the discussion state "rejected", meaning that T01/rj is performed. The cancellation of the acceptance is optional, which is indicated by the cardinality range 0..1. A T01/rj is considered representing the `RejectOffer` event.

6.2.2.6.2 PSD of business process 2

Figure 6-9 exhibits the PSD for the actor role A03 that is indicated as business process 2. If a request (T03/rq) to fulfill an order is received, the actor role A03 needs to respond with a promise (T03/pm), since a decline should not be performed. After performing a promise, A03 can perform T03/rq and T03/ex.

A T03/rq needs to be performed if the requested person in the order is not enlisted in the supplier A03. T03 is initiated by A03 to the other supplier (CA02) that is considered having the requested person in the order. T03 performed by A03 is an optional transaction. It is indicated by the cardinality range 0..n, meaning that at most n transactions T03 are initiated by A03. This number equals the number of the orders sent by the Staffing Company to n other suppliers. As exhibited in the figure, another transaction T03 is enclosed in a T03.

Apparently, the result of T03 becomes the wait condition to perform T03/ex. As shown in Figure 6-9 there is a conditional link from T03/ac to T03/ex with a cardinality range 0..n. Meaning that in dealing with the P-result, T03/ex has to wait until the C-result T03/ac has been created, if the corresponding T03 is available. At most n T03 can be completed successfully.

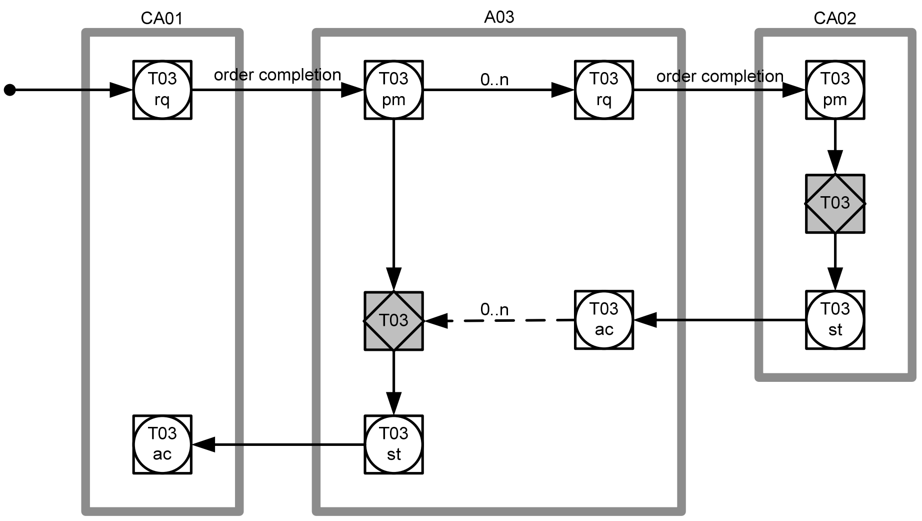


Figure 6-9 PSD of business process 2

6.2.2.7 The AM

6.2.2.7.1 ASM for business process 1

The first agendum to be dealt with by actor A01 is the condition of a T01 being requested. The action rule for A01 in dealing with a T01 being requested is specified in Rule 6-1. T indicates the specification of the `open position`, and the specification of the *customer* that opens the position is indicated by C.

After generating the required entities, A01 needs to checks whether the following conditions are achieved:

- The type of the open position is RFQ
- The status of the open position is new, update, or reopen
- The quantity of the open position is affordable
- There is employee to be offered by the supplier
- The preferred employee in the open position is enlisted in the supplier (A01)
- The reason for opening the position is acceptable
- The reason of change when the open position is updated, is acceptable
- The response time for the supplier to respond the open position is acceptable
- The validity date of the open position is valid, particularly when the supplier providing the offer
- The contract type of the open position is acceptable
- The framework agreement of the open position is agreed
- The CLA of the open position is acceptable

If those conditions are achieved, a promise is performed. Or else, a decline is performed.

Rule 6-1 Action rule for A01 in dealing T01 being requested

```
on requested T01(T) with customer(T) = C
   if  < (type(T) is an RFQ) and
      (status(T) is new or update or reopen) and
      (quantity(T) is affordable) and
      (employee(S) to be offered is available) and
      (the preferred employee(T) is enlisted in the supplier) and
      (reason of position(T) is acceptable) and
      (reason of change(T) is acceptable) and
      (response time(T) is acceptable) and
      (validity date(T) is still valid) and
      (contract type(T) is acceptable) and
      (framework agreement(T) is agreed) and
      (cla(T) is acceptable) > →
         promise T01(T)
   ◊  not < (type(T) is an RFQ) and
      (status(T) is new or update or reopen) and
      (quantity(T) is affordable) and
      (employee(S) to be offered is available) and
      (the preferred employee(T) is enlisted in the supplier) and
      (reason of position(T) is acceptable) and
      (reason of change(T) is acceptable) and
      (response time(T) is acceptable) and
      (validity date(T) is still valid) and
      (contract type(T) is acceptable) and
      (framework agreement(T) is agreed) and
      (cla(T) is acceptable) > →
         decline T01(T)
   fi
no
```

Rule 6-2 exhibits the action rule for A01 in dealing with the condition of T01 being promised. After the C-result of a promise is created, actor role A01 needs to ensure whether the employee to be offered by the supplier is still available. If the condition is fulfilled, the request act of T02 can be performed to A02. If the employee that will be offered is not available anymore, a cancel to the promise of T01 is performed. In initiating T02, A01 needs to generate an entity of the type `offer`, indicated by O. The specification of the *supplier* of the offer is indicated by S. The *offered employee* is indicated by E.

Rule 6-2 Action rule for A01 in dealing T01 being promised

```
on promised T01(T)
   if  < employee(S) to be offered is still available > →
      request T02(new O) with supplier(O) = S and
      offered employee(O) = E
   ◊  not < employee to be offered is available > →
      cancel T01(T)/pm
no
```

The first agendum to be dealt with by the actor A02 is the condition of a T02 being requested. In dealing with the C-result of a request, A02 needs to check whether the following conditions are achieved:

- The qualification of the offered employee is suitable for the required specification of the open position.
- The offered rate is suitable for the rate of the open position
- The offered minimum salary rate is suitable for the minimum salary range of the open position
- The offered maximum salary rate is suitable for the maximum salary range of the open position
- The offered hours per week is suitable for the hours per week of the open position
- The offered days per week is suitable for the days per week of the open position
- The available date of the offered employee is suitable for the work date of the open position
- The shift type of the open position is acceptable
- The work site of the open position is acceptable
- The reason of change of the offer is acceptable

If the conditions are achieved, a promise act of T02 is performed. On the other hand, a decline act is performed. The action rule for A02 in dealing T02 being requested is shown in Rule 6-3.

Rule 6-3 Action rule for A02 in dealing T02 being requested

```
on requested T02(O) with supplier(O) = S and offered employee(O) = E
   if < (qualification of the offered employee(S) is suitable for
      the required qualification(T)) and
      (rate(O) is suitable for rate(T)) and
      (min salary rate(O) is suitable for min salary range(T)) and
      (max salary rate(O) is suitable for max salary range(T)) and
      (hours per week(O) is suitable for hours per week(T)) and
      (days per week(O) is suitable for days per week(T)) and
      (available date of the offered employee(O) is suitable for work
      date(T)) and
      (shift type(T) is acceptable) and
      (work site(T) is acceptable) and
      (reason of change(O) is acceptable) > →
         promise T02(O)
   ◊  not < (qualification of the offered employee(S) is suitable for
      the required qualification(T)) and
      (rate(O) is suitable for rate(T)) and
      (min salary rate(O) is suitable for min salary range(T)) and
      (max salary rate(O) is suitable for max salary range(T)) and
      (hours per week(O) is suitable for hours per week(T)) and
      (days per week(O) is suitable for days per week(T)) and
      (available date of the offered employee(O) is suitable for work
      date(T)) and
      (shift type(T) is acceptable) and
      (work site(T) is acceptable) and
      (reason of change(O) is acceptable) > →
         decline T02(O)
   fi
no
```

The action rule for A02 in dealing with T02 being promised is exhibited in Figure 6-4. In dealing with the C-result of a promise, A02 performs an execute and followed by a state act of T02.

Rule 6-4 Action rule for A02 in dealing T02 being promised

```
on promised T02(O)
   execute T02(O)
   state T02(O)
no
```

The action rules for A01 in dealing with the condition of T02 being stated and accepted is exhibited in Rule 6-5 below. In dealing with the C-result of a statement, if the offered employee is still available, an accept act of T02 is performed. Otherwise, a reject act of T02 is performed. In dealing with the condition of T02 being accepted by A01, an execute followed by a state act will be performed by A01.

Rule 6-5 Action rules for A01 in dealing T02 being stated and accepted

```
on stated T02(O)
   if < offered employee(O) is still available > →
      accept T02(O)
   ◊  not < offered employee(O) is still available > →
      reject T02(O)
   fi
no

on accepted T02(O)
   execute T01(T)
   state T01(T)
no
```

The action rule for A01 in dealing with T01 being rejected and statement cancellation allowed is shown in Rule 6-6. In dealing with a condition of T01 being rejected by the consumer (CA01), if A01 considers that another offer can be stated, a state followed by a cancellation to the statement will be performed. However, if A01 considers that another offer can not be stated, a stop of T01 will be performed. If the cancellation to the statement is allowed by CA01, an execution followed by a statement will be performed by A01. It appears that in performing T01/ex (for the second or more time), A01 needs to perform the request and the acceptance of T02 again, which are required for the execution of T01. Performing another request of T02, meaning that another offer required to be created; it can also represent the `updateOffer` event.

Rule 6-6 Action rules for A01 in dealing T01 being rejected and cancellation allowed

```
on rejected T01(T)
   if < another offer O is state-able > →
      state T01(T)
      cancel T01(T)/st
   ◊  not < another offer O is state-able > →
      stop T01(T)
   fi
no

on allowed T01(T)/cancel(st)
   execute T01(T)
   state T01(T)
no
```

6.2.2.7.2 ASM for business process 2

The action rule for A03 in dealing with a condition of a T03 being requested is exhibited in Rule 6-7. The specification of the *order* is indicated by R. The specification of the *customer* that sends the order is indicated by C, and the specification of the *preferred employee* is indicated by E. In dealing with the C-result of a request, A03 will check whether the type of the received request is an order, and the status of the order is new. If the conditions are achieved, a promise act of a T03 is performed. Otherwise, a decline act of a T03 is performed. In the action rule, the pattern for declination is still specified, even though in practice a declination to an order will not be performed, since A03 has an obligation to fulfill the received order.

Rule 6-7 Action rule for A03 in dealing T03 being requested

```
on requested T03(R) with customer(R) = C and preferred employee(R) = E
   if < (type(R) is an order) and (status(R) is new) > →
      promise T03(R)
   ◊  not < (type(R) is an order) and (status(R) is new) > →
      decline T03(R)
   fi
no
```

In dealing with a condition of T03 being promised, A03 needs to ensure whether the employee preferred by the requester (CA01) is enlisted in the supplier (A03). In case the employee is enlisted, an execute act, followed by a state act of T03 can be performed by A03. Otherwise, since an order must be fulfilled, A03 can perform a request act of T03 to the other supplier. In performing a request act of T03, A03 needs to generate an *order* entity, which is indicated by R. The specification of the customer is indicated by C, and the specification of the preferred employee is indicated by E. The action rule for A03 in dealing T03 being promised is shown in Rule 6-8.

Rule 6-8 Action rule for A03 in dealing T03 being promised

```
on promised T03(R)
   if < the preferred employee(R) is enlisted in the supplier > →
      execute T03(R)
      state T03(R)
   ◊  not < the preferred employee(R) is enlisted in the supplier > →
      request T03(new R) with customer(R) = C and preferred
      employee(R) = E
   fi
no
```

The action rules for A03 in dealing T03 being stated and accepted is exhibited in Rule 6-9. These action rules are only performed when A03 becomes the initiator of T03 that sends an order to the other supplier. In dealing with the C-result of a statement, A03 needs to ensure that the order has been confirmed. If the confirmation is acceptable, an accept act of T03 is performed, or else, a reject act is performed. In dealing with the C-result of an acceptance of T03, A03 can immediately perform an execution and state act of T03.

Rule 6-9 Action rules for A03 in dealing T03 being stated and accepted

```
on stated T03(R)
   if < confirmation of order R is acceptable > →
      accept T03(R)
   ◊  not < confirmation of order R is acceptable > →
      reject T03(R)
   fi
no

on accepted T03(R)
   execute T03(R)
   state T03(R)
no
```

6.2.2.8 The SM: OFD and OPL

The object classes, fact types, result types, and the pertaining existential laws are depicted in the OFD, which is exhibited in Figure 6-10. The entities of the OFD are identified from the AM, it exhibits only the elements or information items that are relevant for the operational of the organization. The OFFER signifies the core category of which the instances are created in the ordering and selection. Several external object classes (colored gray) are identified as well; those are POSITION, OPEN POSITION, ORDER, TYPE, STATUS, REASON, TIME FRAME, QUALIFICATION, PERSON, COMPANY, HOURS PER DAYS, DAYS PER WEEK, RATE, SALARY RANGE, CONTRACT TYPE, CLA, FRAMEWORK AGREEMENT, SHIFT TYPE, and WORK SITE. An ORDER object can be considered as an internal object class, since the Staffing Company may become a consumer that generates and sends an ORDER during the initiation of T03 (*order completion*). In addition, the OFD defines graphically the object classes EMPLOYEE, SUPPLIER, CUSTOMER, and FULFILLED OPEN POSITION. The OFD also depicts the binary and ternary fact types. QUANTITY is depicted as a ratio; a scale with the type NUMBER is identified for QUANTITY. On the other hand, RESPONSE TIME is depicted as an interval; a scale with the type TIME is identified for RESPONSE TIME. The result types of the transaction are exhibited as well in the OFD.

Apparently all of the information items those are identified in the AM have been depicted in the OFD. Therefore, the OPL shown in Table 6-8 consists only the information items those are suitable as the attributes of the object classes of the OFD. Only the information items that are relevant for the operation of the organization are listed in the OPL. Thus, we still need to refer to the documentation to identify the property types that will be listed in the OPL. Some of the scales of the properties specified in the OPL are not primitive types, such as the competence property type which is specified with the scale COMPETENCE, the project property type which is specified with the scale PROJECT, the contact property type which is specified with the scale CONTACT, etc. This implies that those properties can still be specified in a more granularity; meaning that some primitive type properties can be specified for them. As an example, we can specify `CONTACT` with the following properties: `telephone`, `mobile`, `fax`, `email address`, etc. The detail of the properties of each entity can be found in [Enting et al., 2008]. Those details are not exhibited in the SM to keep it consistent with the messages definition (as exhibited in Appendix B) and also to prevent it from being too voluminous. In addition to the OFD and OPL, the derivation rules of the SM are specified as well, which are listed in Rule 6-10.

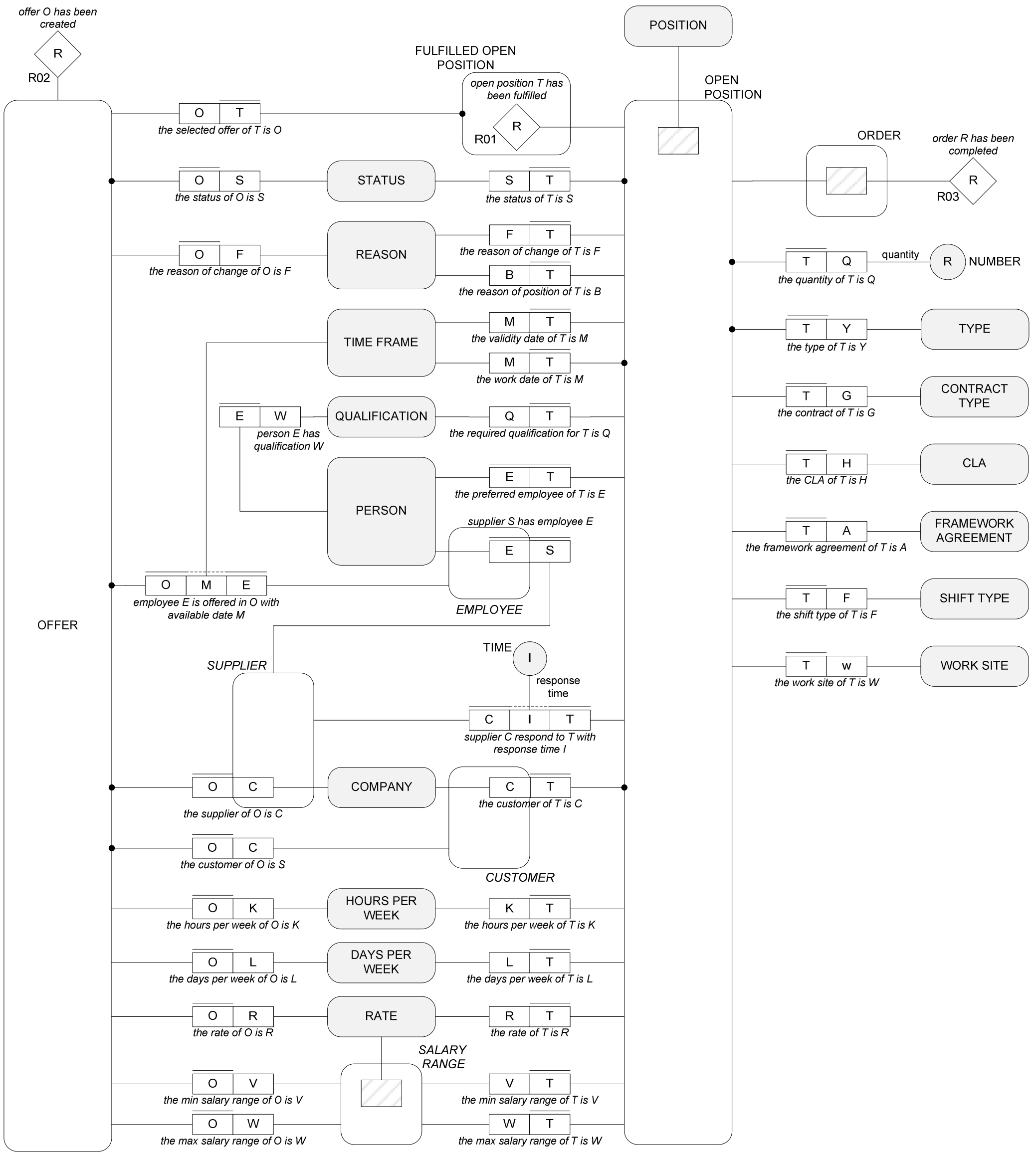


Figure 6-10 OFD of the ordering and selection (not extended)

Table 6-8 OPL of the ordering and selection

Property type	Object class	Scale
`startDate (*)`	`TIME FRAME`	`JULIAN DATE`
`endDate (*)`	`TIME FRAME`	`JULIAN DATE`
`jobTitle`	`POSITION`	`TEXT`
`positionDescription`	`POSITION`	`TEXT`
`competence`	`QUALIFICATION`	`COMPETENCE`
`license`	`QUALIFICATION`	`LICENSE`
`education`	`QUALIFICATION`	`EDUCATION`
`contact`	`COMPANY, PERSON`	`CONTACT`
`project`	`CUSTOMER`	`PROJECT`
`department`	`CUSTOMER`	`DEPARTMENT`
`multiVendorDistribution`	`OPEN POSITION`	`BOOLEAN`

comments	OPEN POSITION, OFFER	TEXT
name	PERSON	TEXT
address	PERSON	ADDRESS
birthDate	PERSON	DATE
sex (*)	PERSON	SEX
employmentHistory	PERSON	EMPLOYMENT HISTORY
identificationDocument	PERSON	IDENTIFICATION DOCUMENT

Rule 6-10 The derivation rules (not extended)

The derivation rules:

```
* QUANTITY > 0
* STATUS has allowed values ‘New’, ‘Revised’, ‘Closed’, ‘Reopened’,
  ‘Cancelled’, ‘x:Rejected’, ‘Pending’, ‘Accepted’, ‘Withdrawn’,
  ‘Rejected’
* STATUS of OPEN POSITION has allowed values ‘New’, ‘Revised’,
  ‘Closed’, ‘Reopened’, ‘Cancelled’, ‘x:Rejected’, ‘Pending’
* STATUS of OFFER has allowed values ‘New’, ‘Revised’, ‘Pending’,
  ‘Accepted’, ‘Withdrawn’, ‘Rejected’
* TYPE has allowed values ‘RFQ’, ‘ORDER’
* OPEN POSITION(x) = POSITION(x) and (∃y ∈ PERSON | x is the request
  for y to fill in x)
* ORDER(x) = OPEN POSITION(x) and (∃y ∈ TYPE | y of x is ‘ORDER’)
* startDate(TIME FRAME) ≤ endDate(TIME FRAME)
* SEX has allowed value ‘M’, ‘F’
* RESPONSE TIME ≤ endDate(VALIDITY)
* startDate(AVAILABLE DATE) ≤ startDate(WORK DATE)
* endDate(AVAILABLE DATE) ≥ endDate(WORK DATE)
* 0 ≤ DAYS PER WEEK ≤ 7
* 0 ≤ HOURS PER WEEK ≤ 168
* MIN SALARY RANGE ≤ MAX SALARY RANGE
* SALARY RANGE(x) = RATE(x) and (∃y ∈ PERIOD | x is the amounts paid
  to the employee regularly in every y)
* CONTRACT TYPE has allowed values ‘recruitment and selection’,
  ‘secondment’, ‘temporary staffing’
```

The OFD exhibited above is not the extended version. It means that the constraint specifications which are derived from the ORM have not been applied. Therefore, in the following part, the additional constraint specifications will be applied and exhibited as the extended version of the OFD, which is shown in Figure 6-11. Most of the constraint specifications are taken from the derivation rules. The same OPL as exhibited in Table 6-8 is applicable for the extended OFD since no constraints are specified based on the OPL, and the updated derivation rules are shown below the OFD as listed in Rule 6-11.

As explained in Chapter 5, for communication purpose, it is considered necessary to depict reference schemes in conceptual data models. For the standards for ordering and selection, the following information items are considered suitable as reference schemes: `rfqOrderId`, `orderId`, `offerId`, `customerId`, `customerSubId`, `supplierId`, `employeeId`, and `jobId`. The `offerId` and `jobId` are implicitly specified; they respectively represent the reference scheme for the OFFER and POSITION object class. On the other hand the `rfqOrderId`, `orderId`, `customerId`, `supplierId`, and `employeeId` are explicitly specified. They are respectively represent the reference schema of the OPEN POSITION, ORDER, CUSTOMER, SUPPLIER, and EMPLOYEE object. The `customerSubId` is combined with `customerId` to represent an external uniqueness constraint.

In addition, from the derivation rules, the specification of the allowed values for the `STATUS`, `STATUS` of the `OPEN POSITION`, `STATUS` of the `OFFER`, `TYPE`, and `CONTRACT TYPE` are specified graphically in the OFD. The value constraints of `DAYS PER WEEK` and `HOURS PER WEEK` are depicted as well in the OFD.

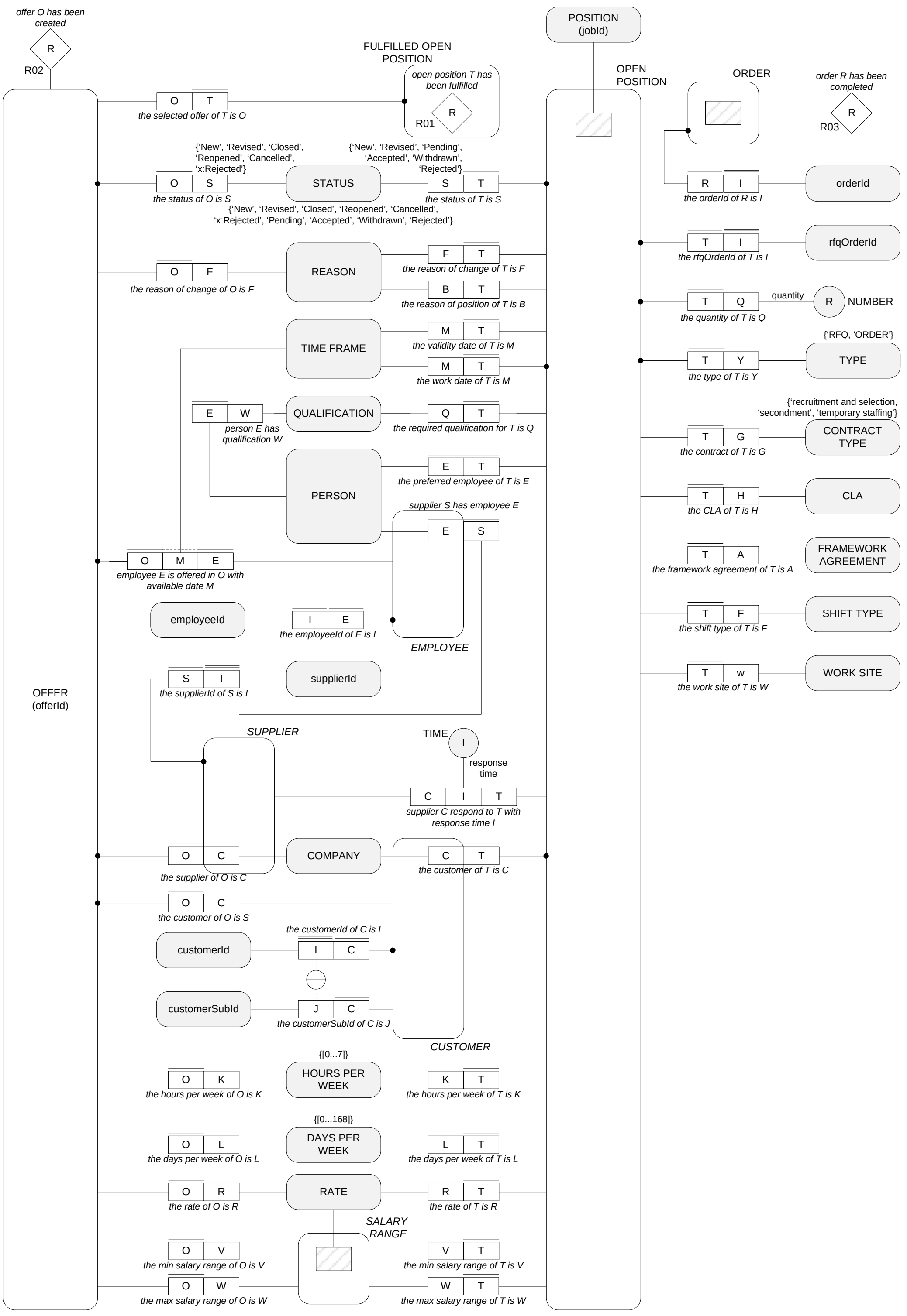

offer O has been created
R02
FULFILLED OPEN POSITION
open position T has been fulfilled
R01
POSITION (jobId)
OPEN POSITION
ORDER
order R has been completed
R03
the selected offer of T is O
{'New', 'Revised', 'Closed', 'Reopened', 'Cancelled', 'x:Rejected'}
{'New', 'Revised', 'Pending', 'Accepted', 'Withdrawn', 'Rejected'}
STATUS
the status of O is S
the status of T is S
{'New', 'Revised', 'Closed', 'Reopened', 'Cancelled', 'x:Rejected', 'Pending', 'Accepted', 'Withdrawn', 'Rejected'}
the orderId of R is I
orderId
the rfqOrderId of T is I
rfqOrderId
REASON
the reason of change of O is F
the reason of change of T is F
the reason of position of T is B
the quantity of T is Q
quantity
NUMBER
TIME FRAME
the validity date of T is M
the work date of T is M
{'RFQ, 'ORDER'}
TYPE
the type of T is Y
QUALIFICATION
person E has qualification W
the required qualification for T is Q
{'recruitment and selection, 'secondment', 'temporary staffing'}
CONTRACT TYPE
the contract of T is G
PERSON
the preferred employee of T is E
supplier S has employee E
CLA
the CLA of T is H
employee E is offered in O with available date M
FRAMEWORK AGREEMENT
the framework agreement of T is A
employeeId
the employeeId of E is I
EMPLOYEE
SHIFT TYPE
the shift type of T is F
WORK SITE
the work site of T is W
supplierId
the supplierId of S is I
OFFER (offerId)
SUPPLIER
TIME
response time
supplier C respond to T with response time I
COMPANY
the supplier of O is C
the customer of T is C
the customer of O is S
the customerId of C is I
customerId
customerSubId
the customerSubId of C is J
CUSTOMER
{[0...7]}
HOURS PER WEEK
the hours per week of O is K
the hours per week of T is K
{[0...168]}
DAYS PER WEEK
the days per week of O is L
the days per week of T is L
RATE
the rate of O is R
the rate of T is R
SALARY RANGE
the min salary range of O is V
the min salary range of T is V
the max salary range of O is W
the max salary range of T is W

Figure 6-11 OFD of the ordering and selection (extended)

Rule 6-11 The derivation rules (updated)

The (updated) derivation rules:

```
* QUANTITY > 0
* OPEN POSITION(x) = POSITION(x) and (∃y ∈ PERSON | x is the request
  for y to fill in x)
* ORDER(x) = OPEN POSITION(x) and (∃y ∈ TYPE | y of x is 'ORDER')
* startDate(TIME FRAME) ≤ endDate(TIME FRAME)
* SEX has allowed value 'M', 'F'
* RESPONSE TIME ≤ endDate(VALIDITY)
* startDate(AVAILABLE DATE) ≤ startDate(WORK DATE)
* endDate(AVAILABLE DATE) ≥ endDate(WORK DATE)
* MIN SALARY RANGE ≤ MAX SALARY RANGE
* SALARY RANGE(x) = RATE(x) and (∃y ∈ PERIOD | x is the amounts paid
  to the employee regularly in every y)
```

The fulfillment of the Requirements 2, 3, 4 and 5 in the case is explained below. The identification and classification of the elements of the organization as exhibited in the OFD signifies the fulfillment of the Requirement 2. The requirement 3 is fulfilled by the fact types that represent the relations between the available object types, such as the fact type that links the OPEN POSITION object and the STATUS object, and the fact type that links the OFFER object and the PERSON object, etc. The Requirement 4 is apparently fulfilled as well. The reference law, unicity law, dependency law, and also external uniqueness constraint are exhibited in the OFD. Reference schemes are also identified. In addition, the value constraints are specified for STATUS, TYPE, etc. The abstractions between the elements of the organization are demonstrated in the OFD. They signify the fulfillment of the Requirement 5 as exhibited in: the OPEN POSITION as the specialization of POSITION, ORDER as the specialization OPEN POSITION, and SALARY RANGE as the specialization of RATE. In addition, partition is also implemented, as found in the object class EMPLOYEE, SUPPLIER, and CUSTOMER; it indicates the fulfillment of the Requirement 5. By fulfilling the four requirements, the conceptual data model of the Ordering and Selection can be completed. Thus, the SM above is considered sufficient for representing the semantics of the Staffing Company that implements ordering and selection.

6.2.3 Implementation of the Requirement 6

The last requirement needs to be fulfilled in order to obtain the semantics of data message is the data message synthesis. This requirement is fulfilled by grouping or classifying the related elements or information items of organization that is considered suitable representing data message. Since the standard for Ordering and Selection defines the messages definition as exhibited in Appendix B, the grouping process for identifying the semantics of data messages will refer to the messages definition.

As mentioned earlier, four types of messages are identified for the Ordering and Selection. Those messages are the `Position`, `PositionStatus`, `Offer`, and `OfferStatus`. The synthesis of the four data messages from the extended OFD will be illustrated in this section.

6.2.3.1 The Position message synthesis

The `Position` message synthesis for the extended OFD is exhibited in Figure 6-12. The `Position` message consists of the POSITION, OPEN POSITION, rfqOrderId, FULFILLED OPEN POSITION, TYPE, STATUS, QUANTITY, CONTRACT TYPE, CLA, FRAMEWORK AGREEMENT, SHIFT TYPE, WORK SITE, REASON OF POSITION, REASON OF CHANGE, VALIDITY, WORK DATE, REQUIRED QUALIFICATION, PREFERRED EMPLOYEE, SUPPLIER, supplierId, RESPONSE TIME, CUSTOMER, customerId, customerSubId, HOURS PER WEEK, DAYS PER WEEK, RATE, MIN SALARY RANGE, and MAX SALARY RANGE object. The Position message is required for the transaction T01 and T03, thus, the result type R01 and R03 are included in the Position message. ORDER object and its id (orderId), which is the specialization of the OPEN POSITION, is also included as part of the Position message. Apparently there is no modification required for the OPL and the derivation rules. The OPL applicable for Position message will be the same OPL as in Table 6-8, and the derivation rule can be found in Rule 6-11.

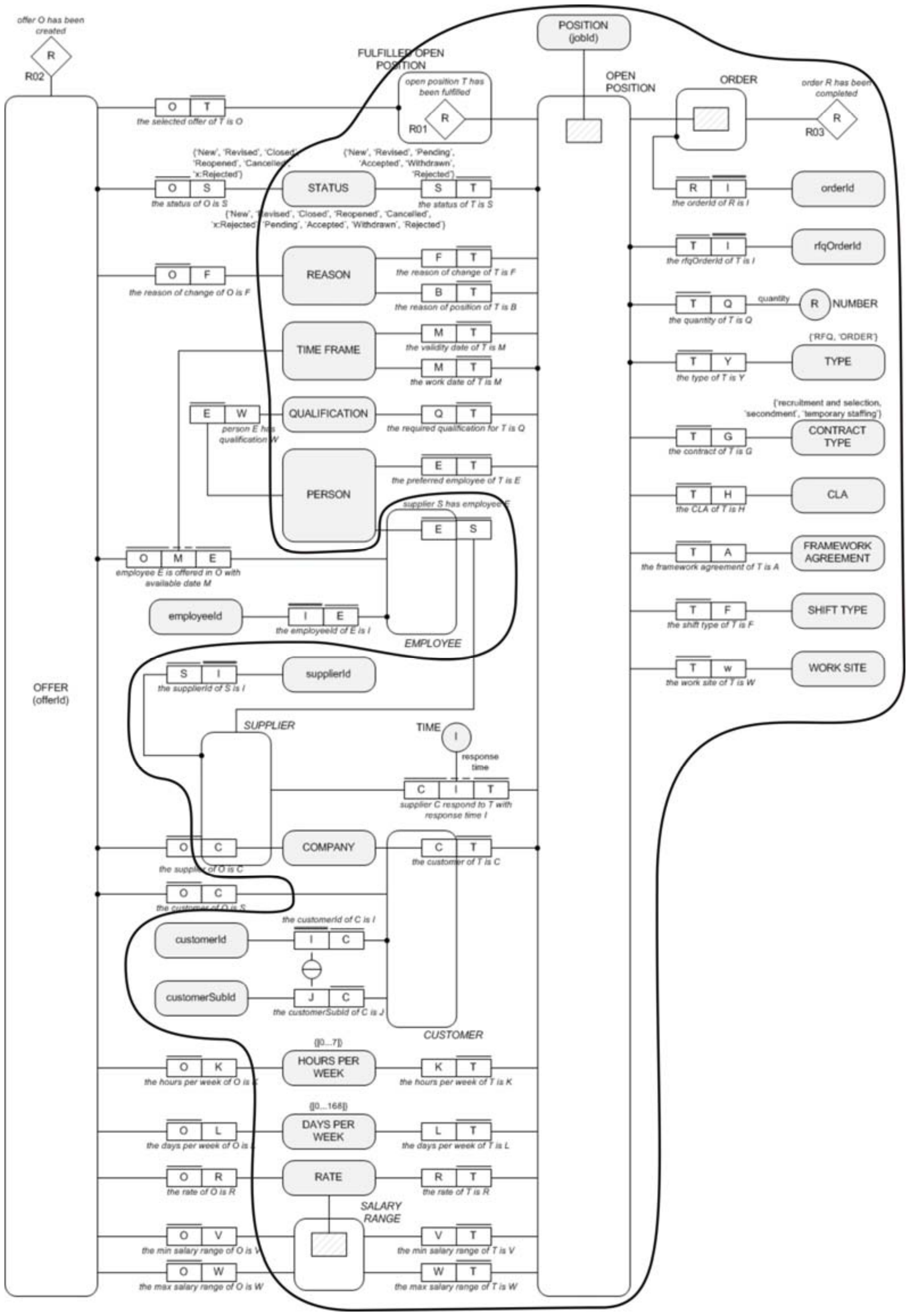


Figure 6-12 Extended OFD of the position message synthesis

6.2.3.2 The PositionStatus Message Synthesis

The `PositionStatus` message is required for the transaction T01. Thus, the result type R01 is contained in the `positionStatus` message. In addition to the `OPEN POSITION` object, the `STATUS` of the `OPEN POSITION` object, as well as the `REASON OF CHANGE` are required for the `PositionStatus` message. The OFD of the `PositionStatus` message is shown in Figure 6-13. No OPL is required to be specified. Nevertheless, the derivation rule of the `OPEN POSITION` is required to be specified, as seen below.

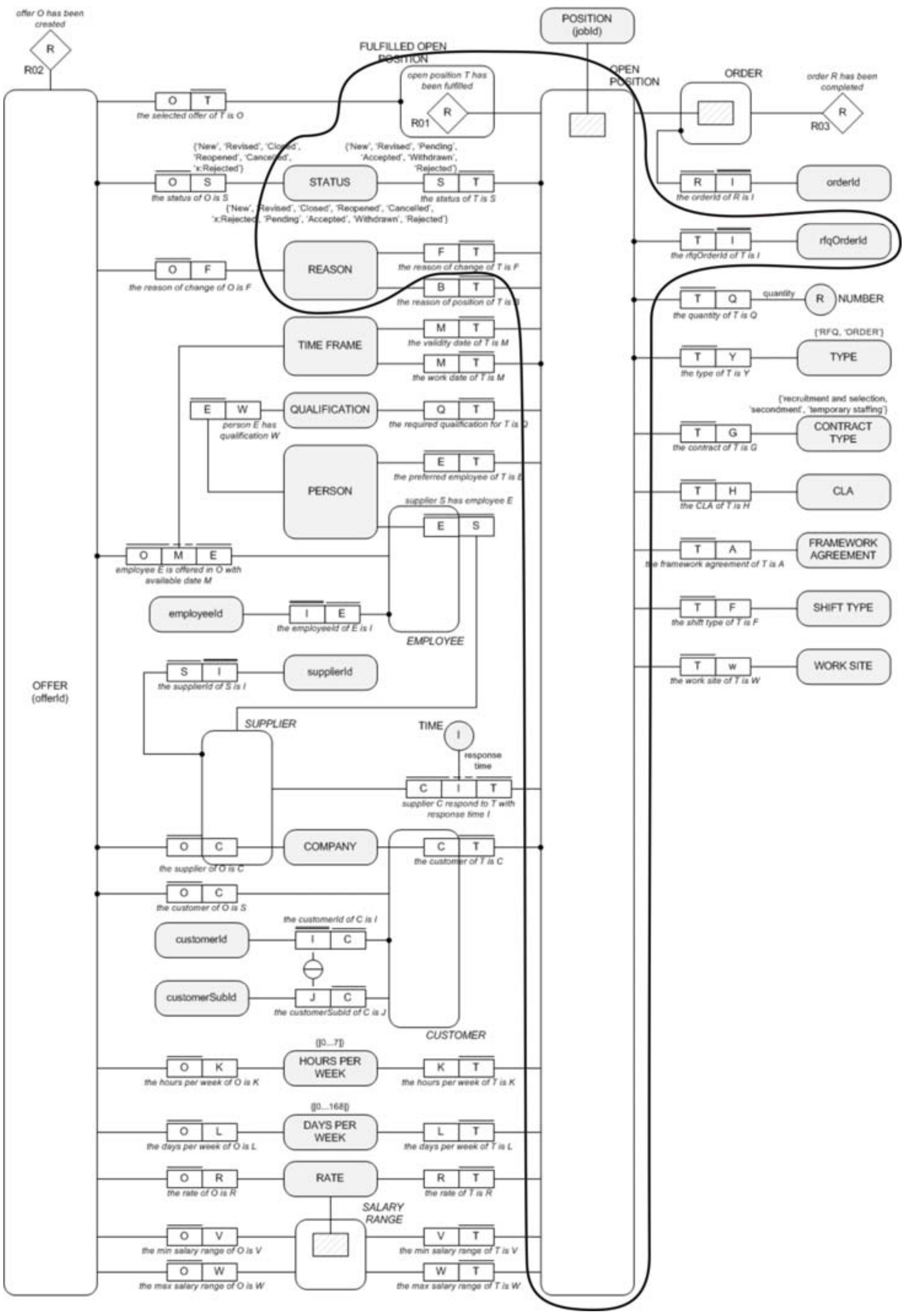


Figure 6-13 Extended OFD of the position status message synthesis

The (updated) derivation rule:

```
* OPEN POSITION(x) = POSITION(x) and (∃y ∈ PERSON | x is the request
  for y to fill in x)
```

6.2.3.3 The Offer Message Synthesis

The `Offer` message consists of the `OFFER`, `FULFILLED OPEN POSITION`, `STATUS`, `REASON OF CHANGE`, `AVAILABLE DATE`, `OFFERED EMPLOYEE QUALIFICATION`, `EMPLOYEE`, `employeeId`, `SUPPLIER`, `supplierId`, `CUSTOMER`, `customerId`, `customerSubId`, `HOURS PER WEEK`, `DAYS PER WEEK`, `RATE`, `MIN SALARY RANGE`, and `MAX SALARY RANGE` object. Since the `Offer` message is required for the transaction T2, the result type R02 is contained as well in the `Offer` message synthesis. R02 is the component of R01; thus, R01 can also be contained in the Offer message, which implies that the `Offer` message is required for T01. The extended OFD of the `Offer` message synthesis is shown in the Figure 6-14. The updated OPL of the Offer message synthesis is in Table 6-9. The derivation rules are listed in Rule 6-12.

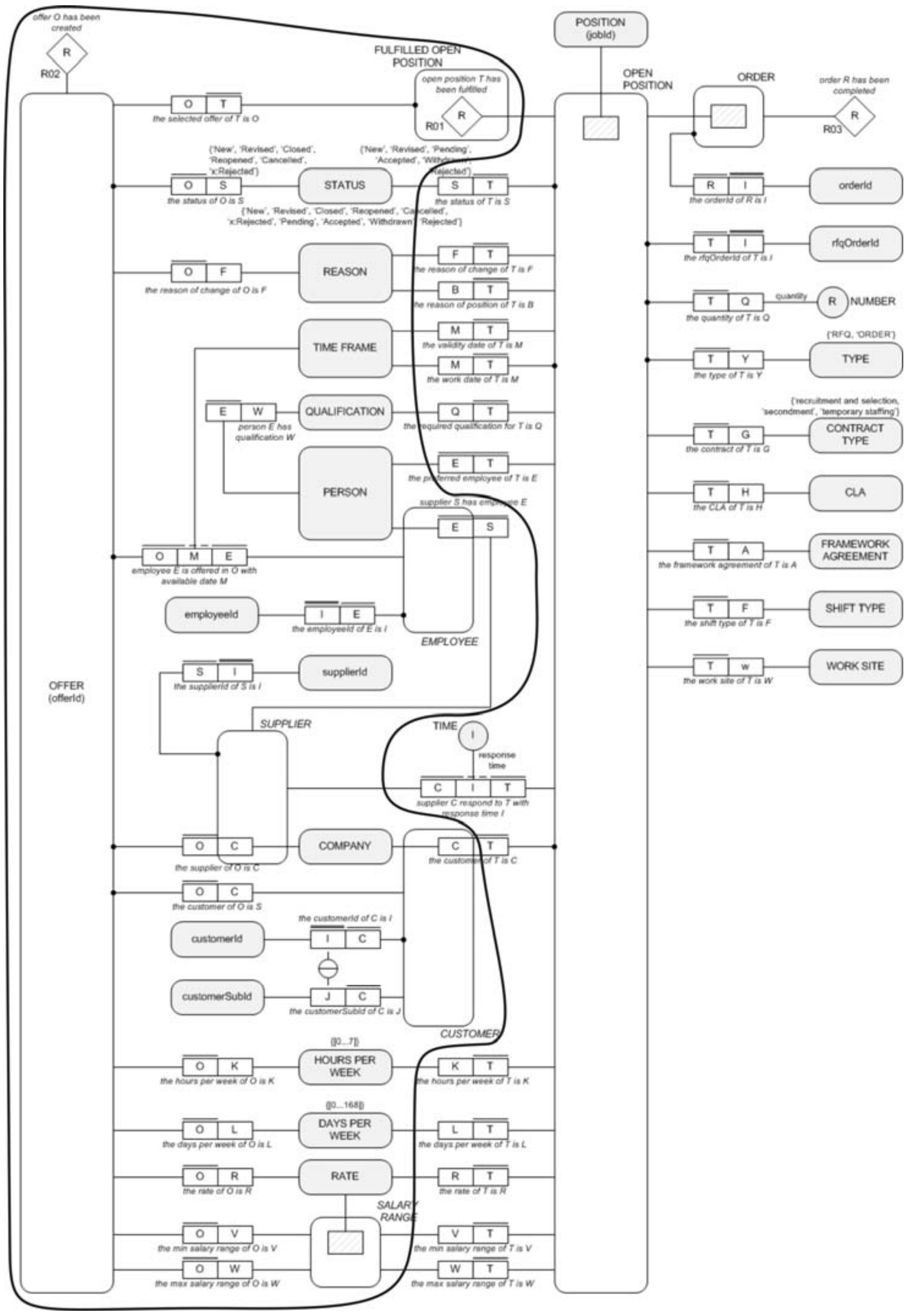


Figure 6-14 Extended OFD of the offer message synthesis

Table 6-9 Updated OPL of the offer message synthesis

Property type	Object class	Scale
startDate (*)	TIME FRAME	JULIAN DATE
endDate (*)	TIME FRAME	JULIAN DATE
jobTitle	POSITION	TEXT
positionDescription	POSITION	TEXT
competence	QUALIFICATION	COMPETENCE
license	QUALIFICATION	LICENSE
education	QUALIFICATION	EDUCATION
contact	COMPANY, PERSON	CONTACT
project	CUSTOMER	PROJECT
department	CUSTOMER	DEPARTMENT
comments	OFFER	TEXT
name	PERSON	TEXT
address	PERSON	ADDRESS
birthDate	PERSON	DATE
sex (*)	PERSON	SEX
employmentHistory	PERSON	EMPLOYMENT HISTORY
identificationDocument	PERSON	IDENTIFICATION DOCUMENT

Rule 6-12 Updated derivation rules of the offer message synthesis

The (updated) derivation rules:

```
 * startDate(TIME FRAME) ≤ endDate(TIME FRAME)
 * SEX has allowed value 'M', 'F'
 * MIN SALARY RANGE ≤ MAX SALARY RANGE
 * SALARY RANGE(x) = RATE(x) and (∃y ∈ PERIOD | x is the amounts paid
   to the employee regularly in every y)
```

6.2.3.4 The OfferStatus Message Synthesis

The `OfferStatus` message synthesis for the basic OFD is exhibited in Figure 6-15. The `OfferStatus` message only concerns the status of the `OFFER` object, thus in addition to the `OFFER` object the `status` object is included in the `OfferStatus` message. In addition, the result types R01 and R02 are contained as well in the `OfferStatus` message, since the `OfferStatus` message is required for transaction T01 and T02. No OPL and derivation rule is specified for the `OfferStatus` message.

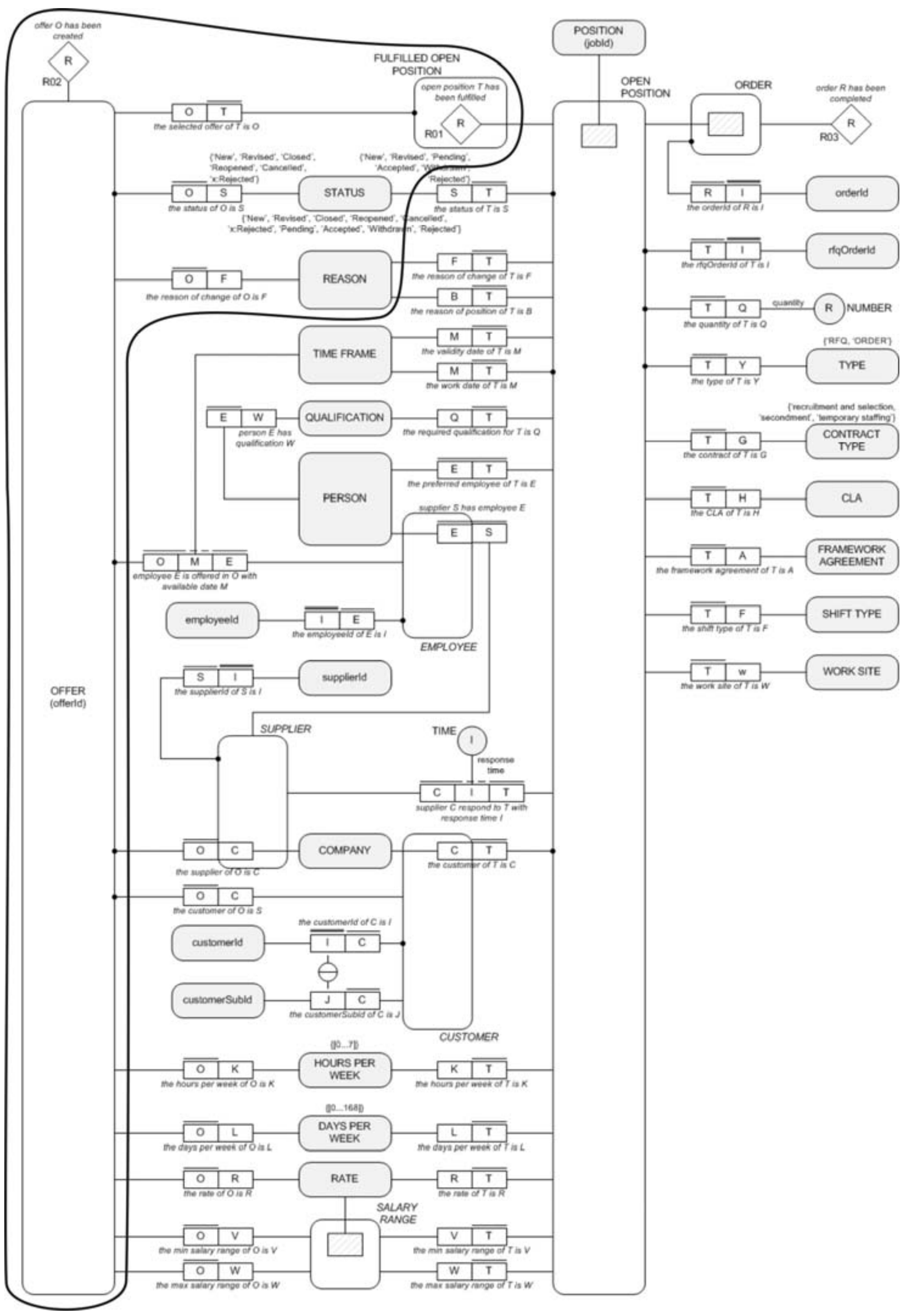

offer O has been created
R
R02
FULFILLED OPEN POSITION
open position T has been fulfilled
R
R01
POSITION (jobId)
OPEN POSITION
ORDER
order R has been completed
R
R03
O T
the selected offer of T is O
{'New', 'Revised', 'Closed', 'Reopened', 'Cancelled', 'x:Rejected'}
{'New', 'Revised', 'Pending', 'Accepted', 'Withdrawn', 'Rejected'}
O S
the status of O is S
STATUS
S T
the status of T is S
{'New', 'Revised', 'Closed', 'Reopened', 'Cancelled', 'x:Rejected', 'Pending', 'Accepted', 'Withdrawn', 'Rejected'}
O F
the reason of change of O is F
REASON
F T
the reason of change of T is F
B T
the reason of position of T is B
TIME FRAME
M T
the validity date of T is M
M T
the work date of T is M
E W
person E has qualification W
QUALIFICATION
Q T
the required qualification for T is Q
PERSON
E T
the preferred employee of T is E
supplier S has employee E
E S
O M E
employee E is offered in O with available date M
employeeId
I E
the employeeId of E is I
EMPLOYEE
S I
the supplierId of S is I
supplierId
OFFER (offerId)
SUPPLIER
TIME
I
response time
C I T
supplier C respond to T with response time I
O C
the supplier of O is C
COMPANY
C T
the customer of T is C
O C
the customer of O is S
the customerId of C is I
customerId
I C
customerSubId
J C
the customerSubId of C is J
CUSTOMER
{[0...7]}
O K
the hours per week of O is K
HOURS PER WEEK
K T
the hours per week of T is K
{[0...168]}
O L
the days per week of O is L
DAYS PER WEEK
L T
the days per week of T is L
O R
the rate of O is R
RATE
R T
the rate of T is R
SALARY RANGE
O V
the min salary range of O is V
V T
the min salary range of T is V
O W
the max salary range of O is W
W T
the max salary range of T is W
R I
the orderId of R is I
orderId
T I
the rfqOrderId of T is I
rfqOrderId
T Q
the quantity of T is Q
quantity
R
NUMBER
{'RFQ, 'ORDER'}
T Y
the type of T is Y
TYPE
{'recruitment and selection, 'secondment', 'temporary staffing'}
T G
the contract of T is G
CONTRACT TYPE
T H
the CLA of T is H
CLA
T A
the framework agreement of T is A
FRAMEWORK AGREEMENT
T F
the shift type of T is F
SHIFT TYPE
T w
the work site of T is W
WORK SITE

Figure 6-15 Extended OFD of the offer status message synthesis

6.3 Conclusion

As demonstrated above, by using DEMO and complying with the requirements for specifying the semantics of data message, the semantics of the data messages of the Ordering and Selection can be specified. Apparently, during the whole process, the description of the case is always necessary to be taken into account.

As illustrated by the case study, the specification of the semantics of data message is represented by the SM. In interpreting the semantics representation, one needs to take into consideration the OFD, OPL, as well as the derivation rules of the SM.

Apparently, the granularity of the entities specified in the conceptual data model depends on details of information identified from operational level. In case less information is identified, fewer entities might be specified. Therefore, it is necessary to further investigate whether different level of granularity affects the level of the understanding one can obtain from the semantics representation.

Thus, by applying DEMO methodology to the case and complying with the six requirements, it is confirmed that the DEMO methodology can be used to construct the specification of the semantics of data message. It is because ontology captures useful semantics of the domain as such the terms and concepts of interest, their meanings, relationships between them, and characteristics of the domain [Patel and Sheth, 2001], which makes it suitable as conceptual data models. Mena et al. stated that use of domain specific ontologies is an appealing approach to allow users to express information (requests) at a higher level of abstraction as compared to keyword based access, in addition, it provides the ability to view an information source at the level of its relevant semantic concepts [Mena et al., 1998]. This concludes the usability of DEMO in specifying the semantics of data message. In addition, it has been confirmed as well that the constraints specification derived from the ORM are smoothly combined with the OFD. The specification of the semantics of the data messages of the Ordering and Selection as defined above represents the answer for the third research question, which is the final outcome of this master thesis project.

Chapter 7
Conclusions and Recommendations

In this chapter the conclusions and recommendations for future work are explained. The conclusions can be found in section 7.1, and the recommendations for future work can be found in section 7.2.

7.1 Conclusions

Based on the requirements analysis and conceptual data modeling, the requirements for specifying the semantics of data message are formulated, it consists of:
Requirement 1: *The identification of organization's requirements*
Requirement 2: *The identification and classification of the elements of organization*
Requirement 3: *The specifications of the relationships between objects*
Requirement 4: *The specification of (additional) constraints*
Requirement 5: *The specification of the available abstractions between objects*
Requirement 6: *The data message synthesis*

The first requirement is derived from the requirements analysis. While, the 2nd, 3rd, 4th, and 5th requirement are formulated based on the conceptual data modeling. Additionally, the last requirement (6th) signifies the synthesis of data message from the produced conceptual data model that is created based on the previous requirements. Overall, all of the requirements represent the aspects required to create conceptual data models with an additional step to specify the semantics of the data of interest that exists in the conceptual data models. By creating the (complete) conceptual data model of organization on the first place, we can understand the context or the domain of data specified in the conceptual data model.

To validate the appropriateness of the requirements, comparison is performed against the CSDP. The CSDP is used to validate some of the requirements those are related to the creation of conceptual schemas. From the validation of the requirement 2, 3, 4 and 5 against the CSDP, it is understood that the four requirements are applicable for creating conceptual schemas. However, the CSDP could not be used for validating the requirement 1 and 6, since it does not contain any step that is relevant to both requirements.

DEMO has been selected as the methodology for specifying the semantics of data message. It is a methodology to develop an enterprise ontology that provides all essential information necessary for the design of the supporting information systems, and at a level of abstraction that makes it also understandable for business people [Albani and Dietz, 2008]. Based on the earlier analysis performed to DEMO, only the SM of DEMO is considered suitable for representing conceptual data models. As stated by Dumay, the SM of DEMO is a conceptual schema of the things and facts that appear to be relevant, where the related information of the organization is modeled. However, since the SM is built based on the other aspect models, they are required to be specified as well. The sequence of the modeling process starts from the ATD and TRT of the IAM, followed by the PSD of the PD, and then continued by the ARS of the AM. After that the OFD and OPL of the SM will be specified based on the AM. This sequence demonstrates that the modeling process in DEMO from the requirements specification to produce the conceptual data model requires a significant amount of time.

The *distinction axiom* has been introduced in the DEMO chapter. The *forma* ability indicates the necessity to express information in a particular language or code schema that is understood by both actors; it represents *syntactic* understanding [Hardjosumarto, 2008]. The *informa* ability denotes that both actor should semantically be in agreement with each other and share the same thought, which means both actors can interpret in the same way; it is called *intellectual* understanding [Hardjosumarto, 2008]. The *performa* ability concerns the creation of new information and knowledge through communication. This holds exposing and evoking commitment and indicates *social* understanding between the involved actors [Hardjosumarto, 2008].

The syntactic understanding is required for performing the informa ability and the intellectual understanding is required for the performa ability. The syntactic and intellectual understanding represents the conditions required for effective communication; while, a service execution can be considered representing the performa ability. As explained in chapter 1, in effective communication there is a correlation between what the sender is thinking about and what the receiver is thinking about. In order to achieve this, the involved actors need to communicate in the same language, in which the selected notation should be understandable and unambiguous, so confusion and error can be avoided [Winbow, 2002]. Thus, we can say that DEMO supports the bringing about effective communication, which is required for a successful service execution, as explicitly specified in the distinction axiom. DEMO itself is considered having a convenient representation; it is specified in a representation that is easy to understand, even for business people.

The possibility of using DEMO for specifying the semantics of data message has been investigated by examining DEMO against the requirements for specifying the semantics of data message. Yet, the SM of DEMO still lacks of the specifications for formulating constraints. Thus, the Requirement 4 is considered only partially fulfilled by DEMO, while the other requirements can be fully satisfied. Due to this reason, several constraint specifications in the ORM are used to extend the SM. By extending DEMO with the ORM; it denotes DEMO's ability to be combined with other methodologies. So finally DEMO can completely fulfill the requirements for specifying the semantics of data message.

The technique of using DEMO for specifying the semantics of data messages has been demonstrated in Chapter 6. In the case study, by implementing the aspect models of DEMO and complying with the requirements for specifying the semantics of data message, the semantics of the data messages of the case study are produced. The semantics of the data messages are represented by the SM that consists of an OFD, which exhibits the particular part of the conceptual schema that represents the semantics of the data message. In addition, it may also consist of an OPL and derivation rules that contain only the information items required for representing the semantics of data message. It is confirmed that ontology can be used as conceptual data model by detailing a certain concern, that is the constraint aspect. Overall, the semantics of data message can be derived from the enterprise ontology.

Several advantages of using DEMO as the methodology for specifying the semantics of data message can be identified as follow.

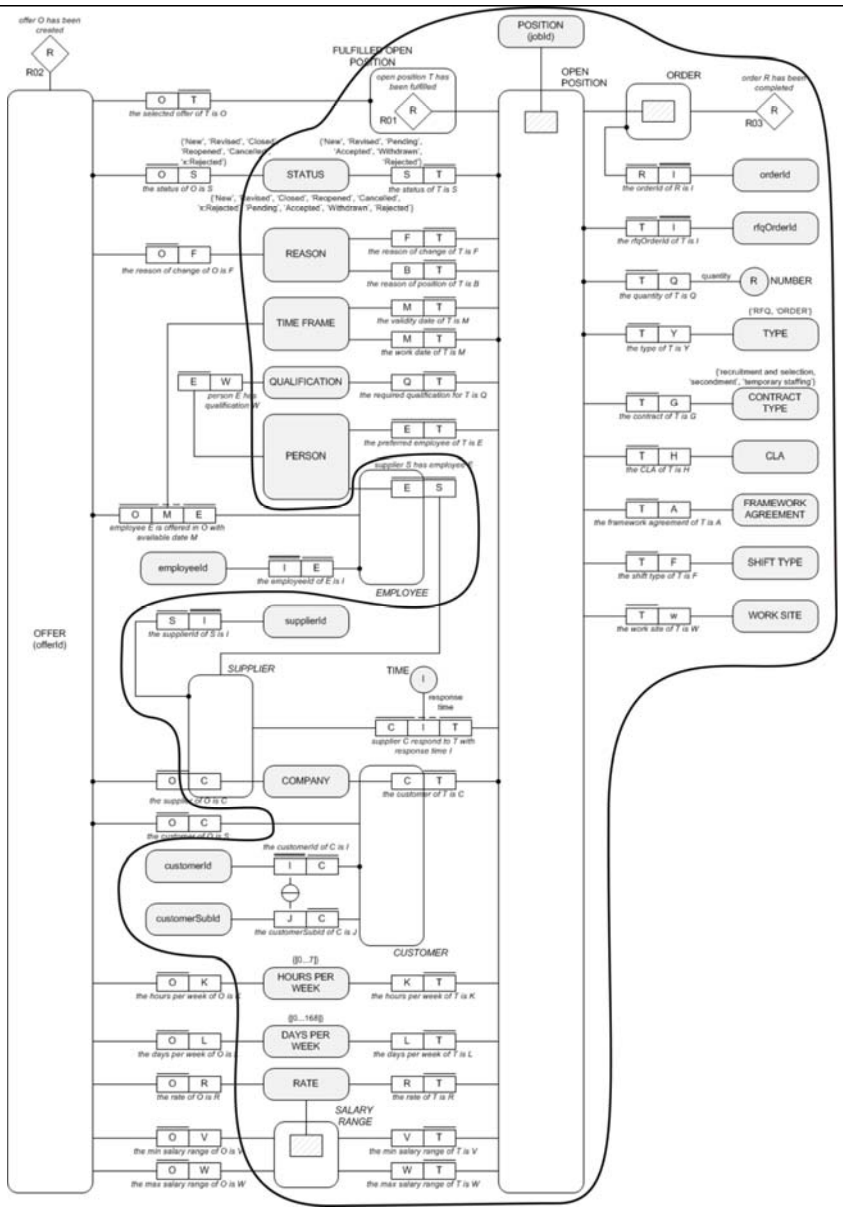
- The ATD of DEMO helps the modeler in defining the boundary of the organization. By focusing only on the aspects within the boundary of the organization, it reduces the complexity in creating conceptual data models by taking into account only the relevant aspects.
- DEMO is considered having a convenient representation. DEMO is represented in compact and understandable forms, as well as the design process is straightforward. Only the essential information is specified, particularly the SM that makes it understandable also for business people.
- The sequence of DEMO aspect models facilitates the construction of proper conceptual data models. Since several aspect models are created in DEMO, it is considered that each aspect model can be used to verify the other aspect models. In addition, by adhering to the sequence of the modeling process, consistency between the aspect models can be maintained. Thus, we can verify the conceptual data model (SM) by checking its consistency with the other aspect models.
- DEMO can be combined with other modeling language. As illustrated above, to completely satisfy all of the requirements for specifying the semantics of data message, the SM of DEMO is extended by using the ORM.

In Chapter 1, Hardjosumarto's Generic Service Specification Framework has been introduced. As identified in chapter 1, the specification of the semantics of the data messages of a service is defined in the Function Description of the Terminology for Service Function Information. To have an idea how the semantics of data messages are specified in a service specification, the specification of the data message of the case study will be provided below.

The transaction T01 will be used an illustration; this transaction will be regarded as a single service named `OpenPositionFulfillment`. In the Table 7-1 below, part of the service specification related to data message will be exhibited. It consists of the specification of the Function Description of the Service Function Information. In addition, the Terminology for the Service Function Information will also be specified. The semantics of the input and output of the service `OpenPositionFulfillment` will be specified in the terminology. The input is the `Position` message and the output is the `Offer` message. The green text shows the Service Function Information section, and the blue text shows the terminology section. In the terminology section, the complete specification of the semantics of the input (`Position`) and output (`Offer`) message is exhibited. It consists of the OFD, the relevant property types listed in the OPL, and the relevant (derivation) rules that are applicable for the `Position` and `Offer` message.

Table 7-1 Part of the OpenPositionFulfillment service specification

Service Function Information:
Function Description:
Function: OpenPositionFulfillment Input: Position Output: Offer
Terminology for Service Function Information:
Function Description:
Input: Position
The OFD:



The OPL:

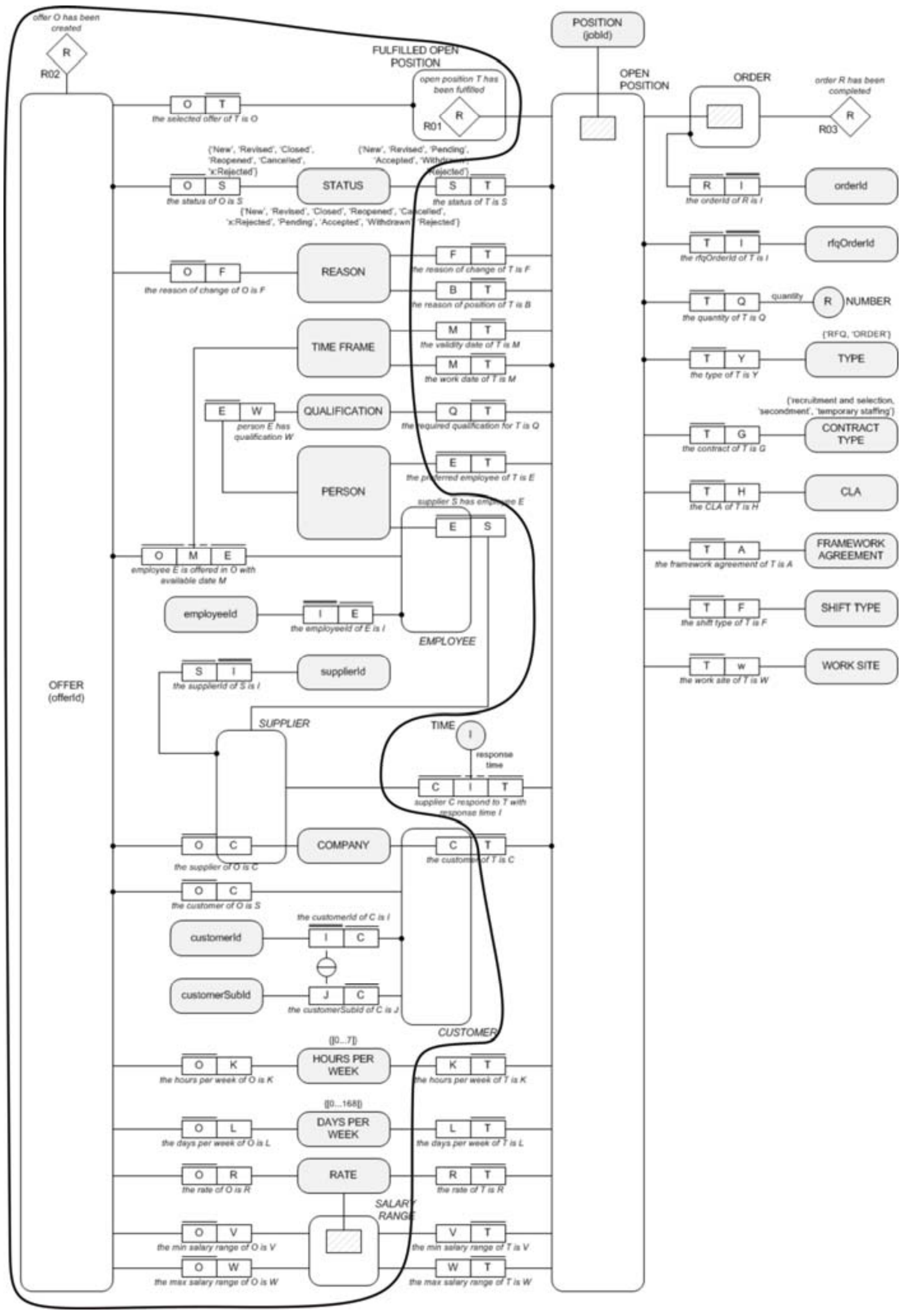
Property type	Object class	Scale
startDate (*)	TIME FRAME	JULIAN DATE
endDate (*)	TIME FRAME	JULIAN DATE
jobTitle	POSITION	TEXT
positionDescription	POSITION	TEXT
competence	QUALIFICATION	COMPETENCE
license	QUALIFICATION	LICENSE
education	QUALIFICATION	EDUCATION
contact	COMPANY, PERSON	CONTACT
project	CUSTOMER	PROJECT
department	CUSTOMER	DEPARTMENT
multiVendorDistribution	OPEN POSITION	BOOLEAN
comments	OPEN POSITION, OFFER	TEXT
name	PERSON	TEXT
address	PERSON	ADDRESS
birthDate	PERSON	DATE
sex (*)	PERSON	SEX
employmentHistory	PERSON	EMPLOYMENT HISTORY
identificationDocument	PERSON	IDENTIFICATION DOCUMENT

The derivation rules:

```
* QUANTITY > 0
* OPEN POSITION(x) = POSITION(x) and (∃y ∈ PERSON | x is the request
  for y to fill in x)
* ORDER(x) = OPEN POSITION(x) and (∃y ∈ TYPE | y of x is 'ORDER')
* startDate(TIME FRAME) ≤ endDate(TIME FRAME)
* SEX has allowed value 'M', 'F'
* RESPONSE TIME ≤ endDate(VALIDITY)
* startDate(AVAILABLE DATE) ≤ startDate(WORK DATE)
* endDate(AVAILABLE DATE) ≥ endDate(WORK DATE)
* MIN SALARY RANGE ≤ MAX SALARY RANGE
* SALARY RANGE(x) = RATE(x) and (∃y ∈ PERIOD | x is the amounts paid
  to the employee regularly in every y)
```

```
Output: Offer
```

The OFD:



The OPL:

Property type	Object class	Scale
startDate (*)	TIME FRAME	JULIAN DATE
endDate (*)	TIME FRAME	JULIAN DATE
jobTitle	POSITION	TEXT
positionDescription	POSITION	TEXT
competence	QUALIFICATION	COMPETENCE
license	QUALIFICATION	LICENSE
education	QUALIFICATION	EDUCATION
contact	COMPANY, PERSON	CONTACT
project	CUSTOMER	PROJECT
department	CUSTOMER	DEPARTMENT
comments	OFFER	TEXT
name	PERSON	TEXT
address	PERSON	ADDRESS
birthDate	PERSON	DATE
sex (*)	PERSON	SEX
employmentHistory	PERSON	EMPLOYMENT HISTORY
identificationDocument	PERSON	IDENTIFICATION DOCUMENT

The derivation rules:

```
* startDate(TIME FRAME) ≤ endDate(TIME FRAME)
* SEX has allowed value 'M', 'F'
* MIN SALARY RANGE ≤ MAX SALARY RANGE
* SALARY RANGE(x) = RATE(x) and (∃y ∈ PERIOD | x is the amounts paid
  to the employee regularly in every y)
```

As illustrated above, the specification of the semantics of the `Position` and `Offer` message are represented in graphical representations, i.e., the OFD. Even though it is considered as a convenient representation, graphical representation can only be understood by human. Thus, in case one needs the conceptual schema to be understood by machine, another representation which is machine understandable is required. Additionally, in order to support automatic service invocation the service specification itself should be specified in a machine understandable specification, since the current one, as shown in the example is still specified in a natural language. Finally, it is concluded that DEMO can be used to specify the semantics of data message; however, it is only understandable for human.

7.2 Recommendations

Based on the analysis performed above, several recommendations for future work are identified as follow.

First, the requirements for specifying the semantics of data messages are formulated based on the analysis performed to data modeling approach. Thus, research on the other approach to formulate the requirements for specifying the semantics of data message can be performed. In addition, a complete validation can also be performed to the produced requirements.

Secondly, only one case study is used to demonstrate the construction of the semantics specification of data message in DEMO. It is possible that the methodology is not applicable to certain cases. Therefore, the specification of the semantics of data message needs to be implemented to more cases. It is because more implementation can help in understanding the usefulness of the methodology in practice. In addition, by having more cases analyzed, we can ensure whether the essential elements (as identified in the SM) are considered sufficient to represent the semantics of data.

Thirdly, the representation of the semantics of data message consists of three main parts: the OFD, OPL, and derivation rules. The specification of the three parts tends to require big space. Therefore, further research to evaluate and improve the representation of the semantics specification is required.

Fourthly, the process of creating the conceptual schema is performed manually starting from the requirement analysis to the synthesis of data message. Obviously manual process like this requires a significant amount of time. Therefore, it would be more convenient to have an automated tool for the process. If possible, to have a tool to create a service specification, in which the specification of the semantics of data message will be one of the supported functions in the tool. For that reason, research on the possibility of creating automatic tool for generating service specification, including the specification of the semantics data message can be performed.

Fifthly, besides the DEMO methodology, the possibility of using other methodology to specify the semantics of data message should be investigated, as well as the possibility of extending DEMO with other methodology. Thus, other methodology can be investigated for the specification of the semantics of data message.

Sixthly, earlier we have identified that the semantics of data message specified in the SM of DEMO can only be understood by human. In case, the specification is required to be understood by machine, another representation that is machine understandable is required. Therefore, further research needs to be performed to create a specification of the semantics of data message that is machine understandable. Additionally, the possibility to transform DEMO model to a machine understandable specification can also be investigated.

Abbreviations

AM	Action Model
ABD	Actor Bank Diagram
ARS	Action Rule Specifications
ATD	Actor Transaction Diagram
B2B	Business to Business
BB	Black-box
BCT	Bank Contents Table
C-acts	Coordination acts
C-bank	Coordination bank
C-facts	Coordination facts
C-results	Coordination results
C-world	Coordination world
CA	Composite Actor
CLA	Collective Labor Agreement
CM	Constructional Model
CSDP	Conceptual Schema Design Procedure
DB	Database
DBMS	Database Management System
DEMO	Design and Engineering Methodology for Organizations
EEMCS	Electrical Engineering, Mathematics, and Computer Science
ER	Entity Relationship
ERD	Entity Relationship Diagram
HTTP	Hypertext Transfer Protocol
IAM	Interaction Model
ICT	Information and Communication Technology
ID	Identification
IP	Internet Protocol
ISM	Interstriction Model
IUT	Information Use Table
OASIS	Organization for the Advancement of Structured Information Standards
OCD	Organization Construction Diagram
OCL	Object Constraints Language
OFD	Object Fact Diagram
OPL	Object Property List
ORM	Object Role Modeling
OWL	Web Ontology Language
P-acts	Production acts
P-bank	Production bank
P-facts	Production facts

P-results	Production results
P-world	Production world
PM	Process Model
PSD	Process Structure Diagram
PSI	Performance in Social Interaction
RDF	Resource Description Framework
RFQ	Request for Quotation
SETU	Stichting Elektronische Transacties Uitzendbranche
SM	State Model
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TPM	Technology Policy and Management
TRT	Transaction Result Table
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WB	White-box
WOSL	World Ontology Specification Language
WS	Web Services
XML	Extensible Markup Language
XSD	XML Schema Definition

References

[Albani and Dietz, 2008] Antonia Albani, Jan L.G. Dietz (2008). Benefits of Enterprise Ontology for the Development of ICT-Based Value Networks. Delft University of Technology. The Netherlands.

[Batini et al., 1992] Carlo Batini, Stefano Ceri, Shamkant B. Navathe (1992). Conceptual Database Design. An Entity-Relationship Approach. The Benjamin/Cummings Publishing Company, Inc.

[Cooper and Qin, 2006] Richard Cooper, Zhenzhou Qin (2006). A Graphical Data Modeling Program with Constraints Specification and Management. Department of Computing Science. University Glasgow.

[Dietz, 2006] Prof.Dr.ir. Jan L.G. Dietz (2006). Enterprise Ontology: Theory and Methodology. Springer.

[Dietz et al., 2007] Prof.Dr.ir. Jan L.G. Dietz, Dr.Dipl-Ing. Antonia Albani, Dr. ir. Jan A.P. Hoogenvorst (2007). Lecture Sheets 6.1 IN4153 – Enterprise Architecture and Web Services. Delft University of Technology. The Netherlands.

[Dumay, 2004] Mark Dumay (2004). Demo or Practice: Critical Analysis of the Language/Action Perspective. Delft University of Technology. The Netherlands.

[Enting et al., 2008] Henk Enting, Dennis Krukkert, Jasper Roes (15 September 2008). SETU Standard – Standard for Ordering and Selection 1.1. SETU.

[Halpin, 1999] Terry Halpin (1999). UML data models from an ORM perspective: Part 7. Available: http://www.orm.net/pdf/orm-jcm7.pdf.

[Halpin, 2001] Terry Halpin (2001). Object-Role Modeling: an Overview. Available: http://www.orm.net/pdf/ORMwhitePaper.pdf.

[Halpin, 2005] Terry Halpin (2005). ORM 2 Graphical Notation. Tehnical Report ORM 2-01, September 2005. Neumont University. Available: http://www.orm.net/pdf/ORM2_TechReport1.pdf.

[Halpin, 2009] Terry Halpin (2009). Object-Role Modeling version 2 (ORM 2). Available: http://www.orm.net/pdf/EncDBS.pdf.

[Hardjosumarto, 2008] An Enterprise Ontology base Approach to Service Specification (2008). Master Thesis Project. Computer Science. Delft University of Technology and Ordina. The Netherlands.

[Hay, 2007] David C. Hay (2007). Data Structure: Data Modeling or XML?. Published: 10 July 2007. Available: http://www.tdan.com/view-articles/5538.

[Hofman, 2008] W.J. Hofman (2008). Service Engineering Workbench – Requirements and standards support. TNO.

[Infogoal, 2009] Infogoal.com (2009). The Data Management Bookstore. Data Modeling Resource Center. Available: http://infogoal.com/dmc/dmcdmd.htm. [Accessed: 17 October 2009].

[Koc, 2001] The Entity-Relationship Model. MGIS 641. Koc University.

[Martin et al., 2004] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, Katia Sycara (2004). OWL-S: Semantic Markup for Web Services. Version: http://www.w3c.org/Submission/ 2004/SUBM-OWL-S-20041122.W3C.

[Mena et al., 1998] E. Mena, V. Kashyap, A. Illarramendi, A. Sheth (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure.

[Meersman et al., 2002] Robert Meersman, Peter Spyns, and Mustafa Jarrar (2002). Data Modelling versus Ontology Engineering. Vrije Universiteit Brussel – STARLab. Brussel, Belgium.

[Meersman, 2001] Robert Meersman (2001). Ontologies and Databases: More than a Fleeting Resemblance. Vrije Universiteit Brussel – STARLab. Brussel, Belgium.

[Murphy, 2009] Mark Murphy (2009). Object-Role Modeling – Part 4 – Ring Constraints and Subtypes. Available: http://blog.starbaseinc.com/blog/business-it-outsourcing/0/0/object-role-modeling-part-4-ring-constraints-and-subtypes. [Accessed: 14 November 2009].

[Papazoglou and Heuvel, 2007] Mike P. Papazoglou and Willem-Jan Heuvel (2007). Service oriented architectures: approaches, technologies and research issues. The VLDB Journal, 16(3):389–415, 2007.

[Patel and Sheth, 2001] Shuchi Patel, Amit Sheth (2001). Planning and Optimizing Semantic Information Requests Using Domain Modeling and Resource Characteristics. Department of Computer Science. University of Georgia. Athens.

[Scheiter, 2006] Torben Schreiter (2006). Semantic Web Services – Fundamentals of Service-Oriented Engineering. Universitat Postdam.

[Setiawati, 2008] Agustina Linda Setiawati (September 2008). Research Assignment – Evaluation on Available Semantic Web Services Standards for Service Specification. Department Information Architecture. Delft University of Technology. The Netherlands.

[Smith et al., 2004] Michael K. Smith, Chris Welty, Deborah L. McGuinness (2004). OWL Web Ontology Language – Guide. W3C Recommendation 10 February 2004.

[ter Bekke, 1992] J. H. ter Bekke (1992). Semantic Data Modeling. Prentice Hall.

[Teorey et al, 2009] Toby J. Teorey, Stephen Buxton, Lowell Fryman, Ralf Hartmut Guting, Terry Halpin, Jan L. Harrington, William H. Inmon, Sam S. Lightstone, Jim Melton, Tony Morgan, Thomas P. Nadeau, Bonnie O'Neil, Elizabeth O'Neil, Patrick O'Neil, Markus Schneider, Graeme Simsion, Graham Witt (2009). Database Design. Know It All. Morgan Kauffmann.

[W.J. Hofman, 2008] W.J. Hofman (2008). Service Engineering Workbench – Overall and First Graduation Project. TNO.

[Winbow, 2002] A. Winbow (2002). The Importance of Effective Communication. International Seminar on Maritime English. STCW and Human Element Section Internal Maritime Organization. Turkey.

[Ws-DIAMOND, 2007] The Ws-DIAMOND Team (2007). WS-DIAMOND. Web Services – DIAgnosability, MONitoring, and Diagnosis. USA.

Appendix A
ORM graphic symbols

Table A-1 Main ORM symbols description [Halpin, 2009]

No	Symbol Name	Description
1.	Entity type	Entity type is known as the non-lexical object type, which equals to concept. It is depicted as a named, soft rectangle, or alternatively an ellipse or hard rectangle. E.g., Person.
2.	Value type	Value type is known as the lexical object type, which equals to term. The shape has dashed lines. E.g., Person name.
3.	Reference scheme	Each entity tpe has a reference scheme, indicating how each instance may be mapped via predicates to a combination of one or more values. Injective (1:1 into) reference schemes mapping entities (e.g., countries) to single values (e.g., country codes) may be abbreviated as in symbol 3 by displaying the reference mode in parentheses, e.g., Country(.code). The reference mode indicates how values relate to the entities. Values are constants with a known denotation, so require no reference scheme.
4.	Independent object type	The exclamation mark in symbol 4 declares that an object type is independent (instances may exist without participating in any elementary facts).
5.	Shadowed object type	Object types displayed in multiple places are shadowed
6.	Unary predicate	e.g., … smokes
7.	Binary predicate	e.g., … loves …. The R/S sign represents the predicate statement, in which forward and inverse readings are separated by slash.
8.	Binary predicate	The arrow tip indicates the other reading direction of the predicate
9.	Ternary predicate	
10.	Duplicate binary predicate	Duplicate predicate shapes are shadowed.
11.	Role names	Roles may be given role names, it is optional. Role name is displayed in square brackets.
12.	Derived fact type	An asterisk indicates that the fact type is derived from

No	Symbol Name	Description
		one or more other fact type.
13.	Derived and stored fact type	If the fact type is derived and stored, a double asterisk is used.
14.	Semi derived fact type	Fact types that are semi-derived are marked "$^{+}$".
15.	Internal uniqueness constraints	They are depicted as bars over one or more roles in a predicate, declare that instances for that role (combination) in the fact type population must be unique.
16.	Internal uniqueness constraints	If the constrained roles are not contiguous, a dotted line separates the constrained roles. For an elementary n-ary association, each internal uniqueness constraint in ORM must span at least n-1 roles.
17.	Preferred uniqueness constraints	A predicate may have many uniqueness constraints, at most one of which may be declared preferred by a double-bar.
18.	External uniqueness constraint	It is shown as a circled uniqueness bar; it may be applied to two or more roles from different predicates by connecting to them with dotted lines. This indicates that instances of the role combination in the join of those predicates are unique.
19.	Preferred external uniqueness constraints	They are depicted by a circled double-bar.
20.	Objectified predicate (nested predicate)	One may objectify relationship; make an object out of it so it can play roles. Graphically, the objectified predicate is enclosed in a soft rectangle, with its name in quotes as shown in symbol 20.
21.	Connection	Roles are connected to their players by a line segment.
22.	Mandatory role constraint	It declares that every instance in the population of the role's object type must play that role. This is shown as a large dot placed at the object type.
23.	Mandatory role constraint	Mandatory role constraint that is placed at the role end. It is similar with dot placed at the object type.
24.	Inclusive-or (disjunctive mandatory)	This constraint that is applied to two or more roles indicates that all instances of the object type population must play at least one of those roles. This is shown by connecting the roles by dotted lines to a circled dot as seen in symbol 24.
25.	Value constraints	Value constraints are specified to restrict the population of an object type or role
26.	Subset constraint	It is represented as a dotted arrow with a circled subset symbol. It restricts the population of the first sequence to be a subset of the second. It is part of set comparison constraints.
27.	Equality constraint	It is depicted as a dotted line with a circled "=" symbol. It indicates the population must be equal. It may be applied between two or more sequence. It is part of set comparison constraints.
28.	Exclusion constraint	It is depicted as a circled "X". It indicates the population must mutually exclusive. It may be applied between two or more sequence. It is part of set comparison constraints.
29.	Exclusive-or	Combining an inclusive-or and exclusion constraint

No	Symbol Name	Description
	constraint	yields an exclusive-or constraint.
30.	Subtype	A solid arrow from one object type to another indicates that the first is a (proper) subtype of the other. E.g., Woman is subtype of Person.
31.	Frequency constraint	This constraint is applied to a role sequence. This indicates that instances that play those roles must do solely *exactly n* times, *at least n* times, *at most n* times, or *at least n and at most m* times.
32.	Value-comparison constraints	The arrow shows the direction in which to apply the circled operator between two instances of the same type, e.g., **For each** Employee, hiredate > birthdate
33.	Ring constraints	They may apply to a pair of compatible roles. Read the stmbols left to right and top row first, these indicate that the binary relation formed by the role population must respectively be irreflexive, asymmetric, antisymmetric, symmetric, intransitive, acyclic, intransitive and acyclic, or intransitive and asymmetric.
34.	Deontic symbol for uniqueness constraints	Obligatory but can be violated uniqueness constraints
35.	Deontic symbol for mandatory constraints	Obligatory but can be violated mandatory constraints
36.	Deontic symbol for set-comparison constraints	Obligatory but can be violated set-comparison constraints
37.	Deontic symbol for frequency constraints	Obligatory but can be violated frequency constraints
38.	Deontic symbol for ring constraints	Obligatory but can be violated ring constraints

Appendix B
SETU Standards for Ordering and Selection

B.1 Events Description

Table B-1 describes the transitions as presented in the state transition diagrams. The first column describes the event that triggers the transition, the second column indicates the source of the messages (consumer or supplier), and the third column describes what the trigger is for the event and the expected response.

Table B-1 States and events of the Position and Offer objects [Enting et al., 2008]

Event	Source	Description
`CreatePosition`	Consumer	The consumer has an open position and creates a message to indicate this to the supplier. The supplier is expected to respond with an offer.
`UpdatePosition`	Consumer	The open position at the consumer changes and the consumer sends a message to update the information at the supplier. The supplier is expected to respond with an offer of an updated offer.
`ClosePosition`	Consumer	The position at the consumer is fulfilled, or does not need to be fulfilled anymore; the consumer sends a close message to the supplier. The supplier is not expected to respond. The ClosePosition message only indicates that the supplier is not expected to respond anymore, the position is not archived. Archiving a position is done with the ArchivePosition message. The position can be reopened after sending the ClosePosition message.
`ReopenPosition`	Consumer	The position was closed, but needs to be fulfilled again; the supplier is notified of this change. The supplier is expected to respond with an offer.
`ArchivePosition`	Consumer	The position is not open anymore and the position is archived. The supplier is notified of this change and is not expected to respond. This message terminates the position; the position can not be reopened. If the position becomes available again a new CreatePosition message has to be sent.
`RejectPosition`	Supplier	The supplier has received a position and rejects this position. The consumer is not expected to respond.

Event	Source	Description
`CreateOffer`	Supplier	The supplier has received a position and responds to this with an offer. The consumer is expected to respond with a reservation, acceptance or rejection.
`UpdateOffer`	Supplier	The supplier has received an updated position, or the information in the offer has changed, the supplier sends an updated offer to the consumer. The consumer is expected to respond with a reservation, acceptance, or rejection
`ReserveOffer`	Consumer	The consumer has received an offer, and places a reservation on the offer. The supplier is not expected to respond.
`AcceptOffer`	Consumer	The consumer has received an offer, may have placed a reservation, and accepts the offer. The supplier is not expected to respond.
`WithdrawOffer`	Supplier	The supplier withdraws the offer because the person offered is not available anymore. The consumer is not expected to respond.
`RejectOffer`	Consumer	The consumer rejects the offer and notifies the supplier of this rejection. The supplier is expected to respond with an update or new offer.
`CreateOrder`	Consumer	The consumer creates an order and sends this order to the supplier.

B.2 Position and PositionStatus Messages Definition

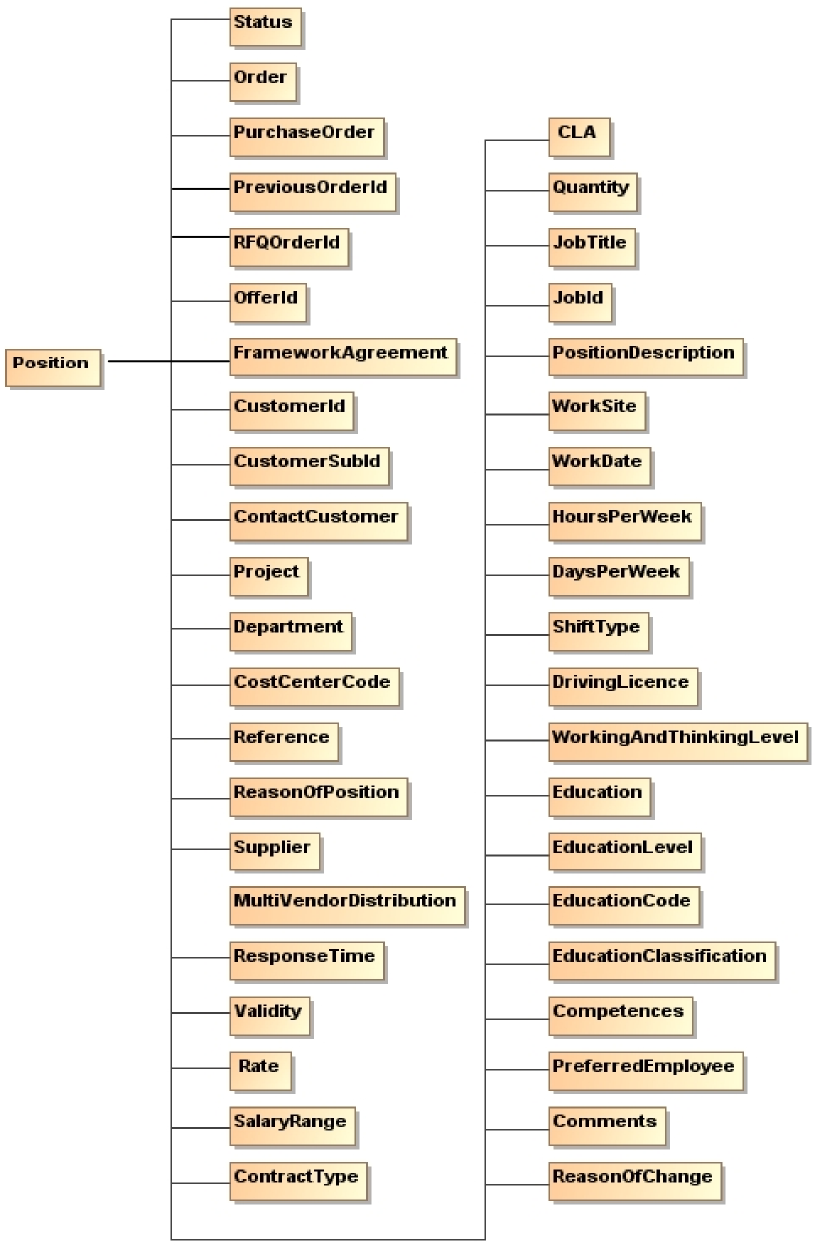


Figure B-1 Position message definition [Enting et al., 2008]

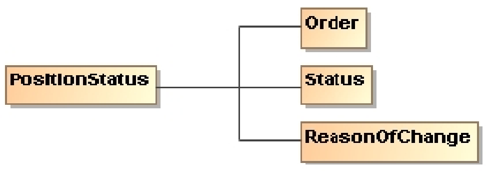


Figure B-2 PositionStatus message definition [Enting et al., 2008]

B.3 Elements Position Message

Table B-2 Elements Object Position [Enting et al., 2008]

Attribute	Mandatory	Description
`Status`	Mandatory	The status of the position.
`Order`	Mandatory	The order number of the position.
`Purchase Order`	Optional	The purchase order number of the position.
`PreviousOrderId`	Optional	Reference to the previous order in case the Position message is used as an order.
`RFQOrderId`	Optional	Reference to the position in case the Position message is used as an order.
`OfferId`	Optional	Reference to the offer in case the Position message is used as an order.
`FrameworkAgreement`	Optional	Reference to a framework agreement.
`CustomerId`	Mandatory	The identifier of the consumer.
`CustomerSubId`	Optional	The sub identifier of the consumer.
`ContactCustomer`	Optional	The name of the contact at the consumer.
`Project`	Optional	The project code of the project at the consumer.
`Department`	Optional	The department name of the department at the consumer.
`CostCenterCode`	Optional	The cost center code of the position.
`Reference`	Optional	A reference number for the position.
`ReasonOfPosition`	Optional	The reason for the open position.
`Supplier`	Optional	The supplier the position is sent to.
`MultiVendorDistribution`	Optional	Information about whether the position is sent to multiple staffing companies.
`ResponseTime`	Optional	The required response time of the supplier.
`Validity`	Optional	The validity of the position.
`Rate`	Optional	The required rate.
`SalaryRange`	Optional	The salary range of the position.
`ContractType`	Optional	The type of contract of the open position.
`CLA`	Optional	The CLA that is applicable to the position.
`Quantity`	Mandatory	The number of open positions.
`JobTitle`	Optional	The title of the job described in the position.
`JobId`	Optional	The identifier of the job described in the position.
`PositionDescription`	Optional	A description of the position.
`WorkSite`	Optional	The site the position is for.
`WorkDate`	Mandatory	The start and end date of the position.
`HoursPerWeek`	Optional	The number of hours per week.
`DaysPerWeek`	Optional	The number of day per week.
`ShiftType`	Optional	The type of shift.
`DrivingLicense`	Optional	The required driving licence.

Attribute	Mandatory	Description
`WorkingAndThinkingLevel`	Optional	The required working and thinking level for the position.
`Education`	Optional	The type of education required.
`EducationLevel`	Optional	The education level required.
`EducationCode`	Optional	The code of the education required.
`EducationClassification`	Optional	The classification of the education required.
`Competences`	Optional	The required competences.
`PreferredEmployee`	Optional	The name and/or employee number of the preferred employee.
`Comments`	Optional	Comments about the open position.
`ReasonOfChange`	Optional	The reason the position has changed.

B.4 Offer and OfferStatus Messages Definition

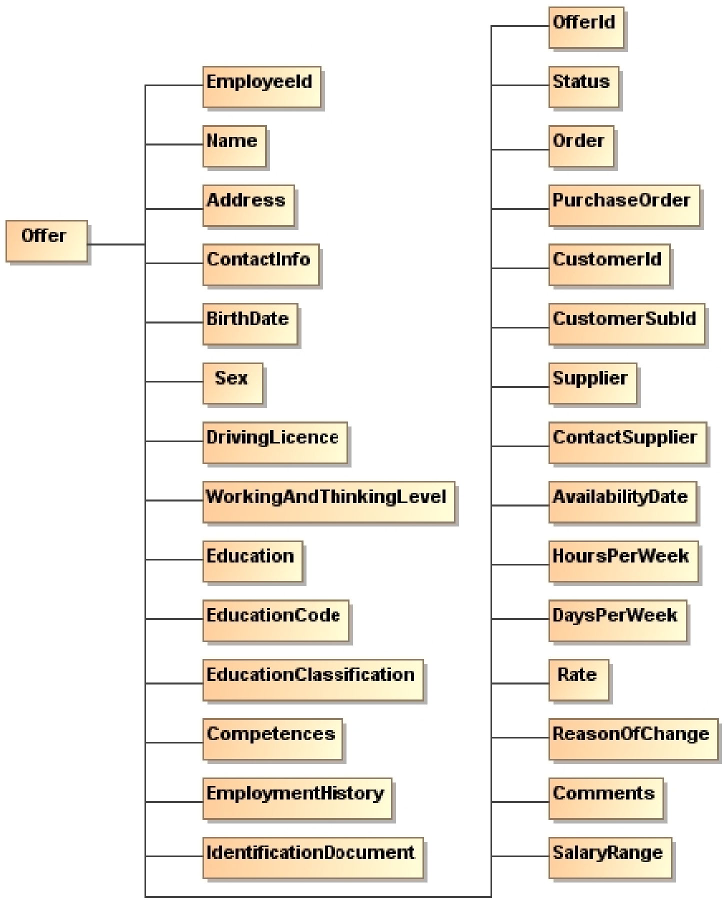


Figure B-3 Offer message definition [Enting et al., 2008]

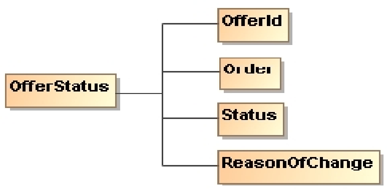


Figure B-4 OfferStatus message definition [Enting et al., 2008]

B.5 Elements Offer Message

Table B-3 Elements Object Offer [Enting et al., 2008]

Attribute	Mandatory	Description
`EmployeeId`	Mandatory	The identified of the employee
`Name`	Optional	The name of the employee
`Address`	Optional	The address of the employee
`ContactInfo`	Optional	Contact information of the employee
`Birthdate`	Optional	The birth date of the employee
`Sex`	Optional	The sex of the employee
`DrivingLicense`	Optional	The driving license the employee has
`WorkingAndThinkingLevel`	Optional	The working and thinking level of the employee
`Education`	Optional	The education the employee has completed
`EducationCode`	Optional	The code of the education of the employee has completed
`EducationClassification`	Optional	The classification of the education the employee has completed
`Competences`	Optional	The competences of the employee
`EmploymentHistory`	Optional	The employment history of the employee
`IdentificationDocument`	Optional	Information about the identification document of the employee
`OfferID`	Mandatory	The unique number of the Offer
`Status`	Mandatory	The status of the offer
`Order`	Optional	The order number associated to the offer
`PurchaseOrder`	Optional	The purchase order number associated to the offer
`CustomerId`	Mandatory	The identifier of the cunsumer
`CustomerSubId`	Optional	The sub identified of the consumer
`Supplier`	Mandatory	The identifier of the supplier
`ContactSupplier`	Optional	The name of the contact person at the supplier
`AvailableDate`	Optional	The start and end date the person offered is available
`HoursPerWeek`	Optional	The number of hours per week offered
`DaysPerWeek`	Optional	The number of days per week offered
`Rate`	Optional	The rate offered
`SalaryRange`	Optional	The salary range offered
`ReasonOfChange`	Optional	The reason of change of the offer
`Comments`	Optional	Comments about the offer

Appendix C
Implementation of the Step 1 and Step 2 of the Requirements Analysis Method of the Case Study

The step 1 (*The Performa-Informa-Forma Analysis*) is performed by coloring the appropriate parts of the descriptions: red for Performa items, green for Informa items, and blue for Forma items. In addition, the step 2 will be performed by adding a small box "[" and "]", or disk "(" and ")", or diamond "<" and ">" over the pieces of text that are marked red. These shapes, respectively, indicate an actor role, a C-act/result, or a P-act/result. The implementation of step 1 and step 2 is shown below.

The SETU Standard for Ordering and Selection deals with electronically sending information related to <ordering> and <selecting> a temporary worker for an open position. It represents the first step on hiring a temporary worker. As shown in Figure C-1, the involved actors in the process are the [Staffing Company] and [Staffing Customer]; the parties may take different roles as listed in Table C-1.

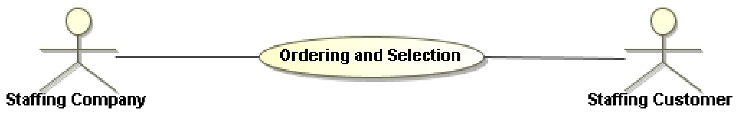


Figure C-1 Parties for ordering and selection [Enting et al., 2008]

A Staffing Customer can notify a Staffing Company for an open position. This indicates that the [Staffing Customer] has (requested) the [Staffing Company] to <provide> employee(s) to fill in the open position. Afterward, a Staffing Company can <provide> an offer as the response to the request of the Staffing Customer, or even (reject) it if the Staffing Company is unable to fulfill the request. Upon receiving an offer from a Staffing Company, the [Staffing Customer] can respond with a (reservation), (acceptance) or (rejection). In addition, an order of a temporary worker can also be sent by the Staffing Customer to the Staffing Company. In receiving an order from a Service Company, the [Service Provider] has to <fulfill> it; a declination must not be performed.

Table C-1 Parties Roles for ordering and selection [Enting et al., 2008]

Party	Role	Description
Staffing	Supplier	The Staffing Company acts as the supplier of (temporary)

Company		workers. After receiving an open position, it starts <looking> for a suitable worker and <offers> his worker to the consumer. The Staffing Company may also receive an order from a procurement system.
	Consumer	In some cases the Staffing Company acts as a consumer of (temporary) workers. This may occur if the Staffing Company has to supply a worker that is not enlisted at the Staffing Company and the Staffing Company <orders> a (temporary) worker to another staffing company.
Staffing Customer	Consumer	The Staffing Customer acts as a consumer of (temporary) worker. After sending an open position to one or more suppliers, it waits for the suppliers to <offer> temporary worker. The staffing customer can also place an order using a procurement system.

The complete events, supported in the Standard for Ordering and Selection, are mentioned below[4].

1. `CreatePosition`: the consumer has an open position and creates a message to indicate this to the supplier]. The [supplier] is expected to <respond> with an offer.
2. `UpdatePosition`: the open position at the consumer changes and the consumer sends a message to update the information at the supplier. The [supplier] is expected to <respond> with an offer of an updated offer.
3. `CreateOrder`: the [consumer] <creates> an order and sends this order to the supplier.
4. `ClosePosition`: the position at the [consumer] is <fulfilled>, or does not need to be fulfilled anymore; the consumer sends a close message to the supplier. The supplier is not expected to respond. The `ClosePosition` message only indicates that the [supplier] is not expected to <respond> anymore, the position is not archived. The position can be reopened after sending the `ClosePosition` message.
5. `ReopenPosition`: the position was closed, but needs to be <fulfilled> again; the supplier is notified of this change. The [supplier] is expected to <respond> with an offer.
6. `ArchivePosition`: the position is not open anymore and the position is (archived). The supplier is notified of this change and is not expected to respond. This message terminates the position; the position can not be reopened. If the position becomes available again a new `CreatePosition` message has to be sent.
7. `RejectPosition`: the [supplier] has received a position and (rejects) this position. The consumer is not expected to respond.
8. `CreateOffer`: the [supplier] has received a position and <responds> to this with an offer. The consumer is expected to respond with a (reservation), (acceptance) or (rejection).
9. `UpdateOffer`: the supplier has received an updated position, or the information in the offer has changed, the supplier sends an updated offer to the consumer. The consumer is expected to respond with a (reservation), (acceptance) or (rejection).
10. `ReserveOffer`: the consumer has received an offer, and places a (reservation) on the offer. The supplier is not expected to respond.
11. `AcceptOffer`: the consumer has received an offer, may have placed a (reservation), and (accepts) the offer. The supplier is not expected to respond.

[4] The complete description of the events can be found in Appendix B.1

12. `WithdrawOffer`: the supplier (withdraws) the offer because the person offered is not available anymore. The consumer is not expected to respond.
13. `RejectOffer`: the consumer (rejects) the offer and notifies the supplier of this rejection. The supplier is expected to <respond> with an update or new offer.

For each event, a message is sent from the Staffing Company to the Staffing Customer, and vice versa. The events and the respective message are listed in Table C-2. All together, four types of message are specified in the Standard for Ordering and Selection: `Position`, `PositionStatus`, `Offer`, and `OfferStatus`. The definitions and description of the messages are exhibited in Appendix B (B.2, B.3, B.4, and B.5).

Table C-2 Message per event [Enting et al., 2008]

No	Event	Message	Status
1.	`CreatePosition`	Position	New
2.	`UpdatePosition`	Position	Revised
3.	`CreateOrder`	Position	New
4.	`ClosePosition`	PositionStatus	Closed
5.	`ReopenPosition`	PositionStatus	Reopened
6.	`ArchivePosition`	PositionStatus	Cancelled
7.	`RejectPosition`	PositionStatus	x:Rejected
8.	`CreateOffer`	Offer	New
9.	`UpdateOffer`	Offer	Revised
10.	`ReserveOffer`	OfferStatus	Pending
11.	`AcceptOffer`	OfferStatus	Accepted
12.	`WithdrawOffer`	OfferStatus	Withdrawn
13.	`RejectOffer`	OfferStatus	Rejected

Appendix D
The Areas of Concern of the Generic Service Specification Framework

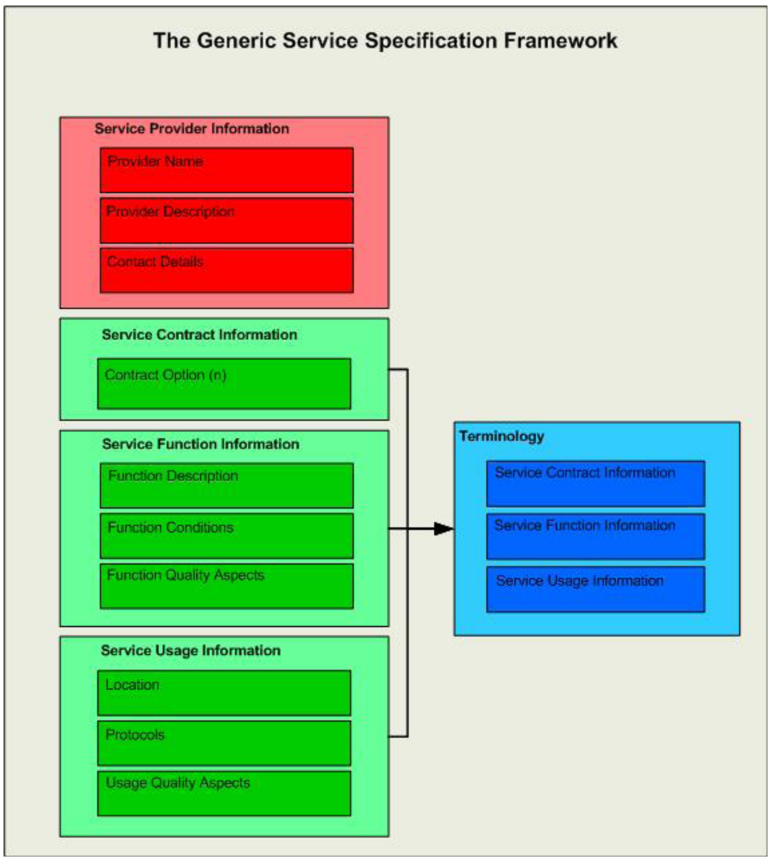

The Generic Service Specification Framework
Service Provider Information
Provider Name
Provider Description
Contact Details
Service Contract Information
Contract Option (n)
Service Function Information
Function Description
Function Conditions
Function Quality Aspects
Service Usage Information
Location
Protocols
Usage Quality Aspects
Terminology
Service Contract Information
Service Function Information
Service Usage Information

Figure D-1 the Generic Service Specification Framework [Hardjosumarto, 2008]

Areas of concern are the main part of Generic Service Specification Framework. It specifies the main aspects required to be specified in the service specification. As shown in Figure D-1, Generic Service Specification Framework comprises five areas of concern, i.e., *Service Provider Information*, *Service Contract Information*, *Service Function Information*, *Service Usage Information*, and *Terminology*. Each concern will be explained in the following sections.

D.1 Service Provider Information

The Service Provider Information describes the service providing party. By providing this information, the service consuming party able to contact the service providing party anytime, especially when problems occured during service invocation. Information specified in the Service Provider Information consists of:

1. *Provider Name*
2. *Provider Description*. It may contain the service providing party background information and the domain wherein the organization operates.
3. *Contact Details*. It mentions the contact person information such as name, phone number, and (email) address.

D.2 Service Contract Information

The Service Contract Information is of interest for management stakeholders. It specifies one or more contract options that can be choose by stakeholders. It consists of:

1. *Service Level*. It denotes the quality level of delivered service. It is based on the quality aspects that are defined in the Service Function Information and Service Usage Information. The service providing party can define different service levels; that depends on the various needs of distinct stakeholders. Usually, it is determined on the basis of service's performance and availability.

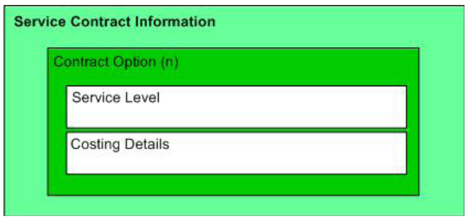


Figure D-2 Service Contract Information [Hardjosumarto, 2008]

2. *Costing detail*. It exposes the price and charge approach of the price. Service providing party could set up different service levels to be provided from which each of these level costs differently.

The Service Contract Information is depicted in Figure D-2.

D.3 Service Function Information

The Service Function Information comprises information that clarifies what kind of service is provided. This information is of interest for the management stakeholders and service consuming parties. Management stakeholders need to know whether the service provides the functionality that meets business requirements. On the other hand, service consuming parties are interested whether the service supports them in performing their activities. Based on the Enterprise Ontology, Service Function Information describes the production act of service and all other aspects that are involved in the production world. As shown in Figure D-3, Service Function Information consists of:

1. *Function Description*. It contains elements that describe the function of providing service component, which is determined by P-acts and can be described in the terms of the relationships between input and output variables. The Function Description specifies the following information:

a) *Function*. It specifies the P-act of service.
b) *Type*. It defines the type of service. There are two general types of service: business and ICT service. Each type of service can be distinguished on the basis of their production type, which is either *datalogical*, *infological*, and *ontological*. Thus, there are six different types of services can be defined.
c) *Supported optional service calls*. It indicates whether the providing service component supports the performance of a *reject-act* or a *cancel-act* as a service call.
d) *Input*. It states the data should be handed over prior to service execution.
e) *Output*. It comprises the data result from the service execution.

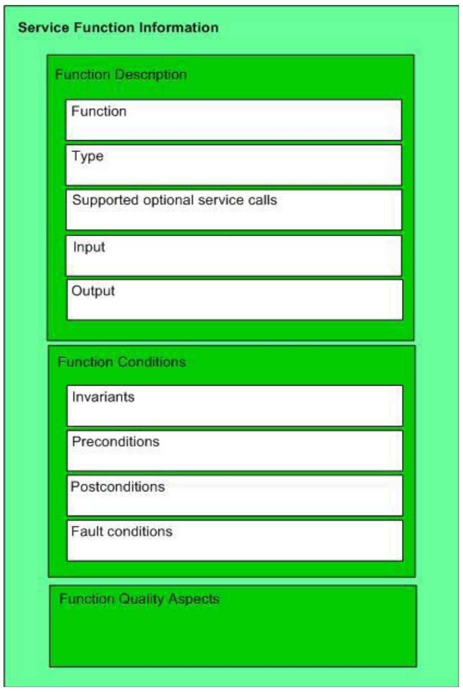


Figure D-3 Service Function Information [Hardjosumarto, 2008]

2. *Function Conditions*. It specifies information to which conditions the input and output should hold to enable successful service execution. It consists of the following aspects:
   a) *Invariants* are conditions, which are preconditions and postconditions as well, and should be complied with throughout the service execution.
   b) *Preconditions* states P-facts as conditions that should always hold prior to the execution of service.
   c) *Postconditions* are P-facts exist after service execution
   d) *Fault conditions* are faults that may rise subsequent to service execution, which should be accompanied with the messages that clarify the stakeholders how to handle the faults.

3. *Function Quality Aspects*. They indicate the quality of the production act of a service. Several aspects that can be specified are *suitability*, *accuracy*, *maturity*, *fault tolerance*, *recoverability*, *time behavior*, *changeability*, *stability*, *testability* and *reusability*.

D.4 Service Usage Information

The Service Usage Information clarifies how the service consuming party can successfully make use a service as described in the Service Function Information. It focuses on how the service consuming party can communicate with the providing service component. The information that is specified in Service Usage Information consists of:

1. *Location*. This aspect depends on the type of the service. If it is an ICT service, location may refer to, for instance, an URL as the endpoint of the service. In the terms of business service, it can denote the physical location of the service, such as room BB.111 at the TNO ICT Delft.
2. *Protocols*. Protocol is defined as the rules governing the syntax, semantics, and synchronization of communication. It determines the successful communication between the consuming service component and the providing service component. It is a way for sharing thoughts such as speaking, listening, and reading by using a common language. For ICT services, it may have value such as TCP/IP, HTTP, and SOAP. Regarding business services, protocol can be for instance the specific rules or manners within an enterprise.
3. *Usage Quality Aspects*. They denote the quality of the communication between the consuming service component and providing service component. This type of quality aspects consist of *interoperability*, *compliance*, *security*, *maturity*, *fault tolerance*, *recoverability*, *time behavior*, *chargeability*, *stability*, *testability*, and *availability*.

D.5 Terminology

The terminology in Generic Service Specification Framework describes the terms used in service specification. It is provided to ensure the syntactic and semantics understanding between the service providing party and service consuming party. Syntactic understanding is enabled by clarifying the form wherein the specification is presented. Semantic understanding can be ensured by using the terminology used by the service consuming party. Thus, service providing party should realize to whom the service specification is intended. Three types of terminology can be specified:

- Terminology for Service Contract Information. Since Service Contract Information provides information on different contract options to enter into an agreement with the service providing party regarding the use of the service, it can be expected that it will mainly comprise of juridical terms. The information only needs to be provided for the management stakeholders.
- Terminology for Service Function Information. Service function information is intended for the management stakeholders and service consuming parties. Thus, it should be described in such a way that is understandable to all types of stakeholders. This indicates the information specified in Service Function Information does not concern whether the service is realized and implemented using human beings or ICT systems. Thus, Service Function Information does not contain technical terms in any case. Graphical model might be incorporated as well.
- Terminology for Service Usage Information. When using a service, it is necessary to know whether a service is provided by human beings or ICT systems. Regarding ICT services, the information can be described using technical terms, which clearly indicates ICT systems as the service providers.