Object-Role Modeling: Validation of a database design methodology in the context of an EHR system

Design of a database blueprint

Peter Martena

UMCG, Programma Nieuw EPD
RUG, Master Technology & Operations Management

Groningen, januari 2015

Studentenbureau UMCG

Universitair Medisch Centrum Groningen

Object-Role Modeling: Validation of a database design methodology in the context of an EHR system

Design of a database blueprint

Groningen, januari 2015

Auteur	Peter Martena
Studentnummer	S2324040
Afstudeerscriptie in het kader van	FEB Technology & Operations Management Rijksuniversiteit Groningen
Opdrachtgever	Ivonne Lesman-Leegte Nieuw EPD, UMCG
Begeleider onderwijsinstelling	Dr. H. Balsters FEB Rijksuniversiteit Groningen
Begeleider UMCG	Ivonne Lesman-Leegte Nieuw EPD, UMCG

Trefw ORM, database design, eMeasure, DCM, BPMN

Preface

This thesis is the result of the final research project of the MSc Technology and Operations Management program at the University of Groningen. Although the final research project was very challenging at times, it was an enormous learning experience in the field of healthcare operations, database modeling and design science. I would first like to thank all of the people at the LTHN that supported this project. Their enthusiasm, knowledge and insights have been crucial for the completion of this thesis.

I would also like to thank my fellow students Rick Beukeboom and Pim van de Laar for their support, input, cooperation and discussions. Finally, I would like to say a special thanks to Dr. Herman Balsters for his guidance, feedback and support during this research and a thank you to the co-assessor Dr. Jos Bokhorst for assessing both my research proposal and master's thesis.

Table of contents

Summary

A Large Teaching Hospital in the Netherlands (LTHN) has started a project with an ICT company for the design and implementation of an Electronic Health Record system (EHR) in order to create relatively higher levels of standardization and quality. The EHR focusses on creating applications that are linked to a central database; the LTHN wants to design a database that includes the building blocks for the design of these applications. The goals of this thesis are to convert the BPMN models to an ontology, create a database blueprint for the newly designed processes and validate the BPMN-ORM methodology. The original 14 Fact-Type Identification questions of BPMN-ORM methodology are reduced to 12 questions, which is achieved by combining various questions in a more efficient way. This paper proposes to add an additional question to the methodology that addresses the validation of the created models.

1. Introduction

Most of us are aware that healthcare costs are rising every year and that people, on average, are living longer. On the other hand, hospitals are constantly pressured by governments, managers and society to lower costs and improve efficiencies (Okma & Crivelli, 2013). However, it is imperative that the relatively high quality standard of hospitals is not affected by these cost reducing- and efficiency improving projects (Vanberkel, Boucherie, Hans, Hurink, & Litvak, 2010). The current set of standards used in hospitals are according to Health Level 7 (HL-7); which support the administrative and clinical processes in hospitals (Jaffe, Hammond, Quinn & Dolin, 2009). Troseth (2010) claims that standardization of these processes helps to reduce costs whilst maintaining (or even improving) quality. She further states that many healthcare organizations will start to focus more and more on the standardization of procedures, processes, documentation and technology. Approximately a decade ago, Van Ginniken (2002) stated that even doctors and clinicians, who usually do not like change, are becoming aware that there is a need for computerized standardization. So why is there still a relatively low level of computerized standardization? Van Ginniken (2002) stated that many projects are too much based on technological advancement instead of fulfilling the end users' needs and that this caused a lot of IT-projects to fail in the healthcare sector.

A Large Teaching Hospital in the Netherlands (LTHN) has started a project with an ICT company for the design and implementation of an Electronic Health Record system (EHR) in order to create relatively higher levels of standardization and quality. One of the focus points of the EHR is to design a process for the creation of Detailed Clinical Models (DCM) and eMeasures that should be linked to a database. However, there are a lot of applications over numerous disciplines within the healthcare sector. This increases the level of complexity for the creation and design of these DCM's and eMeasures. In order to deal with this relatively high level of complexity, the LTHN wants to design a database that includes the building blocks for the design of DCM's and eMeasures. These building blocks are relatively small parts of the DCM's and eMeasures and contain (medical) information. Examples of these building blocks are 'Check value sets' or 'Determine code system'. The objective of this thesis is to design a sociotechnical system (relatively high end user involvement with technology) for the design of DCM's and eMeasures. This sociotechnical system should include specifications validated by the end users and serve as a blueprint for the ICT company. The ICT company will use the blueprints to create the information systems. It is important that these systems are based on the needs of the end users (e.g. researchers) in order for these systems to be considered successful.

The LTHN states that the current (undesired) situation also includes the fact that there is almost no structure in the currently used files/data structure. The LTHN also claims that researchers currently need to search for the required data in multiple sources like the data warehouse, Word files and/or Excel files and that the required data is often unavailable. These issues have led to a new desired situation in which an EHR is used that provides the required data in a structured, standardized and more efficient way.

This project consists of two main parts. This first part consists of an information analysis regarding the stakeholders, work flows, creating a library of building blocks, creating and designing Business Process Modeling and Notation (BPMN) models and validating the methods used. This part is done by Beukeboom (2015) and Van de Laar (2015) and will serve as an input for this thesis.

The second part focuses on converting the BPMN models to an ontology and creating a database blueprint for the processes. Ontologies are used for information retrieval from big data, databases or processes. Akmal, Sing and Batres (2014) state that ontologies are frameworks that consist of classes and relationships that facilitate a common understanding between stakeholders. Object-Role Modeling (ORM) will be used to create the database blueprints. These blueprints are then validated by the end user and, if required, adjusted to the end user's needs.

The managerial relevance, or societal relevance in this case, is to create a standardized information system that helps to reduce costs and to support doctors and researchers at standard tasks and operations. The scientific relevance of this research is to validate the BPMN-ORM methodology for designing the database blueprint. In other words, checking the methodology for mistakes, difficulties or potentially simplifying the step-by-step approach in the context of an EHR system. The methodology should also be made generalizable so that it can be used in other hospitals. In conclusion:

"*In the new EHR system we build a sociotechnical system with specifications validated by the end users to attain an information system in order to support the end user when needed.*"

Based on the problem context, the following research question for this thesis is formulated:

"How can an end user validated database blueprint be derived from the BPMN models within the context of an EHR system at the LTHN?"

The sub-questions to answer the main research questions are discussed on the next page.

These are the sub-questions formulated for this thesis.

1. *"How can we convert end user validated BPMN models to an end user validated database blueprint?"*
2. *"How can the current database design method be improved and generalized?"*

Chapter 2 of this thesis discusses the theoretical framework. The methodology is discussed in chapter 3. Chapter 4, 5 and 6 are respectively the results, discussion and conclusions.

2. Theoretical background

This chapter discusses the theoretical background of the research. Design science, BPMN, ontologies and ORM are discussed.

2.1 Design science

In order to gain a deeper understanding of the problem, the first step is to determine the type of scientific problem of this research within the information systems discipline. The work of Hevner, March, Park & Ram (2004) describes two main types of scientific problems within this discipline: the behavioral science problem and the design science problem. So what are behavioral science and design science problems? According to Hevner et al. (2004) behavioral science is the development/validation of human and/or organizational behavior. Design science focusses on the creation of artifacts in order to transform the current (undesired) situation into a new desired situation (Hevner et al., 2004). Design science deals with practical-knowledge problems, whereas behavioral science deals with pure science problems (Wieringa, 2007). When comparing both types of problems to the LTHN case it becomes clear that this is a design science problem; the LTHN aims to create an artifact (database) in order to move to a new desired situation in which the new information systems are used.

The next step to gain a more clear understanding of the problem is to determine if this particular problem concerns a so called type 1 or type 2 problem. Type 1 problems aim to improve an existing context (e.g. EHR system) which leads to a new theory or results. The other type of design science problems, the type 2 problems, focus on the experimentation of building a system and validating this system and its design principles with the end users or experts in the field (Balsters, 2014). Since this project first experiments with building a database blueprint and then validating the design and system, this problem is defined as a type 2 problem.

2.2 BPMN

One of the main focus points of the first part of the project is to develop and design a BPMN model for the creation of a DCM and an eMeasure. These BPMN models serve as an input for the design phase of the project. But what is BPMN? BPMN, or Business Process Modeling and Notation, is a language for process modeling and was officially released in February, 2006 (Recker, 2010). This language makes it possible to map processes and increase the level of standardization in organizations.

So why use BPMN and not a different program for modeling and visualizing the processes? Recker (2010) states that, although there are relatively many alternatives to graphical modeling of business processes, BPMN has basically become the standard for graphical process modeling. This statement is validated in the paper of Chinosi and Trombetta (2012), who state that BPMN is now an internationally accepted (ISO) standard for (graphical) modeling of business processes. According to Balsters (2014), BPMN is widely accepted and makes the project and process modeling easier to understand for end users and IT.

The BPMN basic elements are flow objects, data, connections, pools, lanes and artifacts (Balsters, 2014). The flow objects, tasks, gateways and events are shown in figure 2.1. Data (e.g. paper or envelope) can be either input or output of an activity. Connections are ways of connecting objects in BPMN. Pools can contain multiple lanes and are rectangular. A pool can be an organization, company or division. Swimlanes represent the different stakeholders that are directly involved with the process. The lanes are parts of the pools and are used for organizing objects. Artifacts are text annotations that provide additional information regarding objects (Balsters, 2014). For a more detailed description regarding the basic elements of BPMN please refer to the papers of Chinosi and Trombetta (2012) or Dijkman, Dumas and Ouyang (2008).

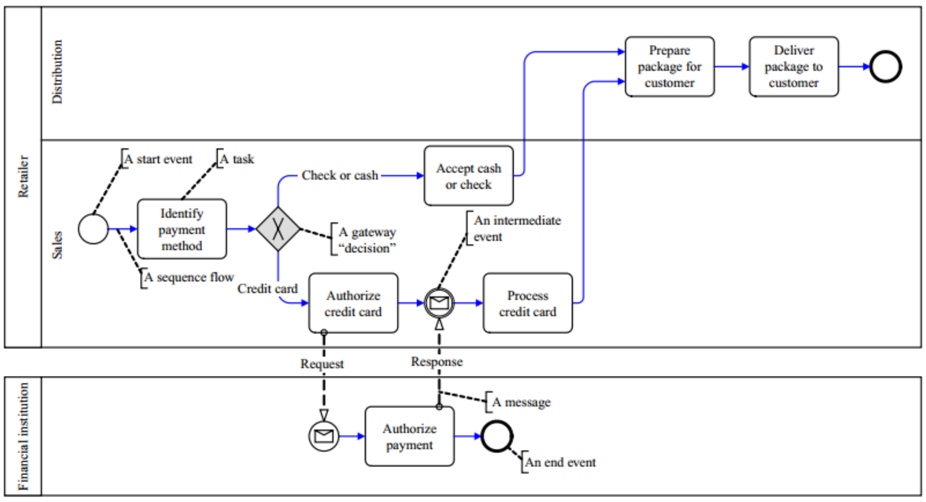


Figure 2.1: BPMN representation of a payment process (Recker, 2010).

Most of the relevant data regarding the DCM's and eMeasures are currently documented in Microsoft Word files. These will be used to create BPMN models and will serve as input for the creation of the ontology and database.

2.3 Ontology and database modeling

The BPMN models need to be converted to an ontology. An ontology is a framework that consists of classes and relationships and facilitates a common understanding between stakeholders (Akmal et al., 2014). Noy and McGuinness (2001) state that an ontology defines a vocabulary for a domain so that end users and domain experts can share information. This vocabulary of a domain includes definitions of concepts and their relationships. According to Noy and McGuinness (2001), there are several reasons for developing an ontology:

- To facilitate and share a common understanding of information in a domain;
- To facilitate the possibility to reuse knowledge within the domain;
- To analyze the knowledge within this domain;
- To make a distinction between operational and domain knowledge;
- The possibility to explicitly define and state assumptions within the domain.

The LTHN wants to create a library of building blocks and a framework for the data models. In order to create this framework, the BPMN models of the first part of this project are considered as the 'business domain'. A business domain is called in technical terms 'Universe of Discourse' (Halpin and Morgan, 2008). The Universe of Discourse (UoD) is a part of the world that we would like to understand and model.

Halpin and Morgan (2008) state that understanding the UoD is one of the most important phases in modeling and that there should be an extensive collaboration between the modeler and the domain experts.

The ontology will be used as a knowledge base to develop the database blueprint for the BPMN models. So now we know what an ontology is, but what is exactly included in an ontology? Noy and McGuinness (2001) state an ontology consists of:

- Classes (definition of concepts in a UoD, can include sub-classes);
- Slots (features/attributes of the concepts, also called roles);
- Facets (restrictions of slots);

Knowledge bases are created by classes of an ontology by defining the individual instances of these classes (Noy and McGuinness, 2001). They also state in their paper that there is not a scientifically correct methodology for developing an ontology, but do state that ontologies are often developed according to these steps:

1. Defining the classes within the domain/ontology;
2. Hierarchically order the (sub)classes;
3. Defining the slots and their allowed values;
4. Entering in the values (of the instances) in the defined slots.

The ontology will then be used to model the business domain. Sub-chapters 2.3.1 to 2.3.3 discuss three modeling approaches: Entity-Relationship Modeling, Object-Role Modeling and Unified Modeling Language.

2.3.1 Entity-Relationship Modeling (ERM)

Although introduced in the mid 70's, ERM is still the most used method for data modeling. ERM models the world as entities that have attributes and relationships (Halpin and Morgan, 2008). Figure 2.2 shows an example of an ERM diagram.

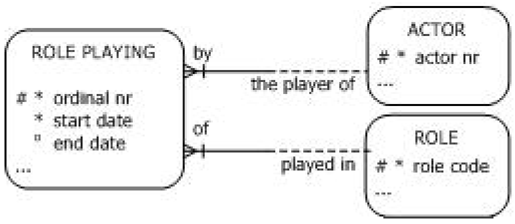


Figure 2.2: ERM diagram.

The entities in the diagram are represented by the rectangles with the rounded edges (Actor, role and role playing). The relationships between the entities are represented by the lines between entities. Each part of the relationship can be identified by the broken and solid line between identities. The so called 'crow's foot' that attaches the relationship to the Role playing entity means that multiple entities (e.g. actors) can be associated with the role playing entity.

Halpin and Morgan (2008) state that it is not easy to move from a data case to an ERM diagram, because of the complexity to model constraints and attributes. They also state in their book that ERM diagrams are relatively hard to validate and lack a high level of readability.

2.3.2 Object-Role Modeling (ORM)

Object-Role Modeling (ORM) is a *fact based modeling approach*. This means that ORM views the world as objects and roles (relationship parts) and focuses on modeling facts in a relatively easy understandable language for end users and domain experts (Halpin and Morgan, 2008). Balsters (2013) states that ORM is attribute free, in contrast to ERM. Figure 2.3 shows an example of an ORM diagram. The figure is pretty straightforward and requires relatively little technical knowledge of the end user, which makes it easier to understand and validate than ERM.

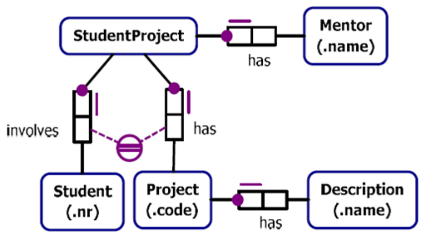


Figure 2.3: Object-Role model example (Balsters, 2014).

Let us briefly elaborate on the ORM diagram of figure 2.3. The entities are shown as rectangles with rounded edges (e.g. StudentProject, Mentor). This is the same as in the ERM diagrams discussed in 2.3.1. The identification of the entities are displayed in parenthesis, also called the reference mode (Balsters, 2014). For example, a student can be identified by his or her student number (.nr) and a mentor by his or her name (.name). The entities are connected by predicates (relationships) which are the boxes between the entities. A relationship can consist of multiple boxes; each box represents an entities' role. Figure 2.4 shows various predicate types.

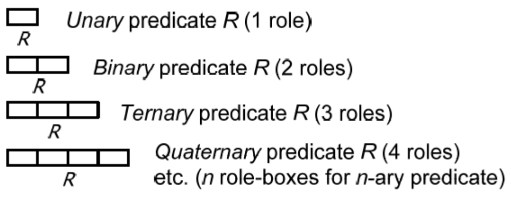

Unary predicate *R* (1 role)
Binary predicate *R* (2 roles)
Ternary predicate *R* (3 roles)
Quaternary predicate *R* (4 roles)
etc. (*n* role-boxes for *n*-ary predicate)
R
R
R
R

Figure 2.4: Various types of predicates (Balsters, 2014).

The predicates in figure 2.3 include a purple dot; this means that this is *mandatory* in order to continue the relationship/process. The binary predicates in figure 2.3 also include a purple line above certain role boxes. These are called *uniqueness constraints*; the entry for that box needs to be unique and does not allow for duplicates (Halpin and Morgan, 2008). An example regarding the uniqueness constraint is that each student project has at most one project (.code).

Halpin and Morgan (2008) state that ORM advantages over ERM are its simplicity, stability and ease of validation with the end users. A disadvantage of ORM is that it often leads to larger diagrams. NORMA (a plugin for Microsoft Visual Studio) overcomes this disadvantage by breaking up the large diagrams in relatively smaller parts. Because the ORM diagrams are relatively easier to understand and validate than ERM, ORM will be used for this project. The BPMN models are converted to ORM diagrams. The ORM diagrams can also be converted to a Unified Modeling Language (see 2.3.3), which is more understandable for programmers.

2.3.3 Unified Modeling Language (UML)

UML enables the ORM diagrams to be more specified for programming purposes of the database and is for programmers easier to read than ORM (Halpin and Morgan, 2008). This is an advantage for IT, however validating UML diagrams with the end user is relatively difficult. ORM offers this advantage, but is for (IT) programmers less suitable because it is on a more conceptual level. ORM offers the opportunity to validate the designs with the end users, where UML is in general easier to understand for programmers. Figure 2.5 shows an example of a UML class diagram.

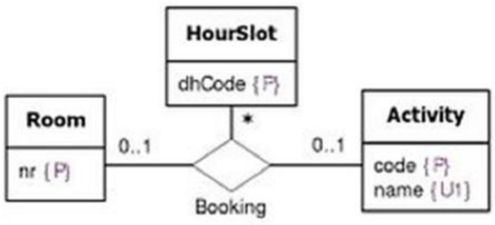


Figure 2.5: UML class diagram by Halpin and Morgan (2008).

One of the limitations of UML is the capability of verbalization of facts which makes validation with the end user more difficult. It is therefore important that the BPMN models are validated with the end users because they serve as input for the ORM diagrams and UML.

2.4 Database modeling and roadmap

This sub-chapter consists of two parts: database modeling and the roadmap. The first part discusses the database modeling, from the BPMN models to the database blueprint. The second part explains the roadmap of Balsters (2013). This roadmap is a general step-by-step process which will serve as a guideline for modeling the ORM diagrams.

2.4.1 Database modeling

This paragraph will explain how to move from the BPMN models to the database blueprint. The BPMN-ORM methodology of Balsters (2014) is used in order to design the database blueprint from the BPMN models. This methodology helps to convert BPMN models into a relatively easy understandable language, even for non-technical people (Balsters, 2013). Figure 2.6 shows an example of a BPMN model created with the Bizagi tool (www.bizagi.com).

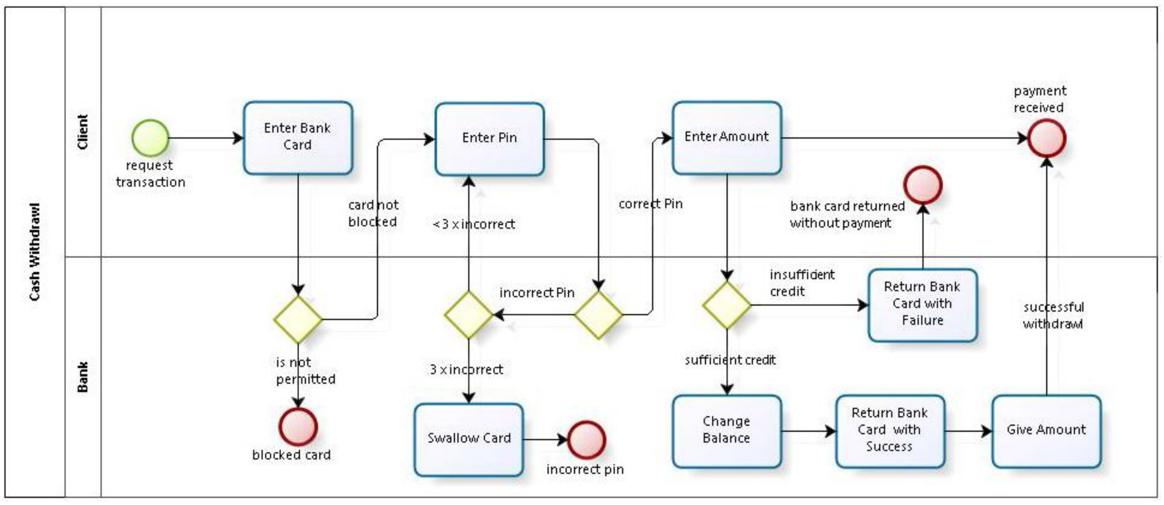


Figure 2.6: BPMN model of a bank transaction (Balsters, 2014).

For explanation purposes, we will now zoom in on how to model the ‘Enter Bank Card’ task (Figure 2.7) into an Object-Role Model (Figure 2.8).

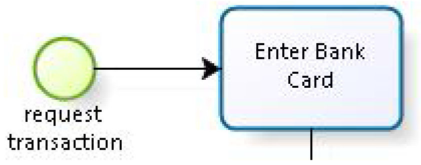


Figure 2.7: The 'Enter Bank Card' task.

The tasks in BPMN models are phrased as Verb-Noun (Example: Enter-Bank Card). The verb refers to the action (e.g. enter) and the noun refers to a person, action, item, etc. This is a common method for describing actions in BPMN. The ORM notation is different than the BPMN phrases; ORM uses a Noun-Verb. The verb within the ORM phrase is objectified; this means that it is an event. An event is the occurrence of an activity at a particular moment in time (e.g. 'Entry'). Converting the BPMN phrase of 'Enter Bank Card' to an ORM phrase yields 'Bank Card Entry'; the Bank Card is the noun and Entry is the objectified verb (Balsters, 2014).

Balsters (2014) developed a methodology to construct the data models from BPMN models (BPMN-ORM methodology). As mentioned in chapter 1, the scientific relevance of this thesis is to validate this methodology. This methodology contains 14 questions and are shown in table 2.1. The questions in this table are called 'Fact-Type Identification' or FTI questions.

1	What is the event we are addressing?
2	Which stakeholders are involved?
3	What are the stakeholder goals?
4	What are the CSF's for each stakeholder goal in the context of this event?
5	Which objects are involved in the event as participants?
6	Which fact types are these participants engaged in?
7	Which constraints pertain to these fact types?
8	How do we identify the participants?
9	Which fact types is the event engaging in?
10	Which constraints pertain to these fact types?
11	How do we identify the event?
12	What are the input events for our particular event?
13	What do we have as output values of the event?
14	What are the associated business rules for these outputs?

Table 2.1: Fact-Type Identification questions (Balsters, 2014).

The data models can be built by using this methodology. Answering these 14 questions yields figure 2.8 for the 'Bank Card Entry'. This is modeled with the NORMA tool in Visual studio.

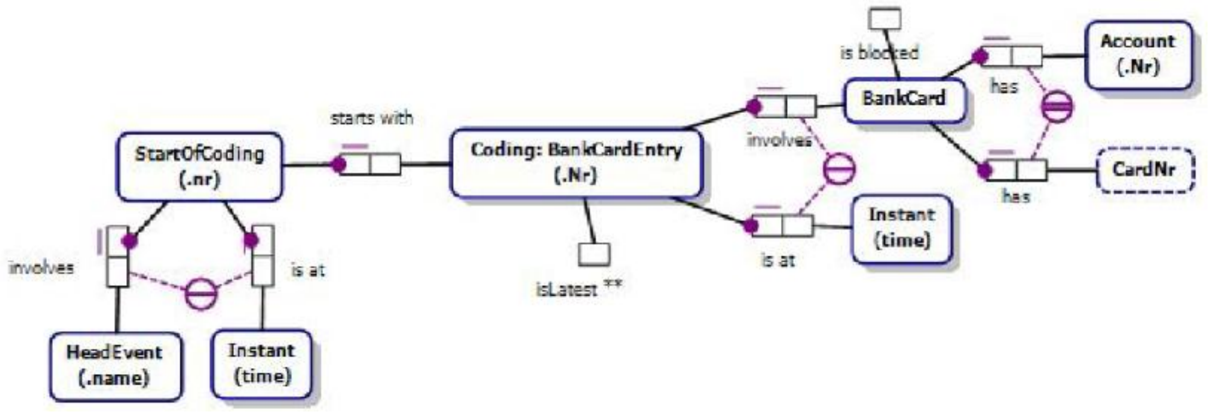


Figure 2.8: Example of ORM of 'Enter Bank Card' (Balsters, 2014).

Using the NORMA tool on ORM yields a part of the database blueprint for the Bank Card transaction (Figure 2.9). NORMA is a tool that helps to automatically link all data models with each other in order to create the database blueprint.

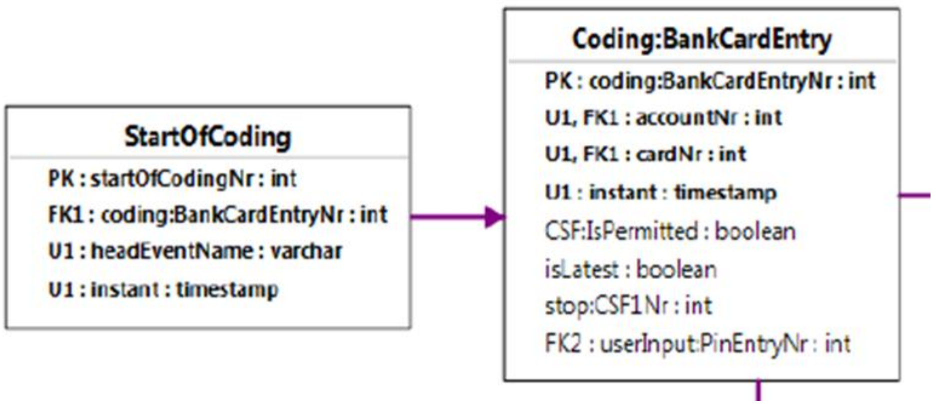
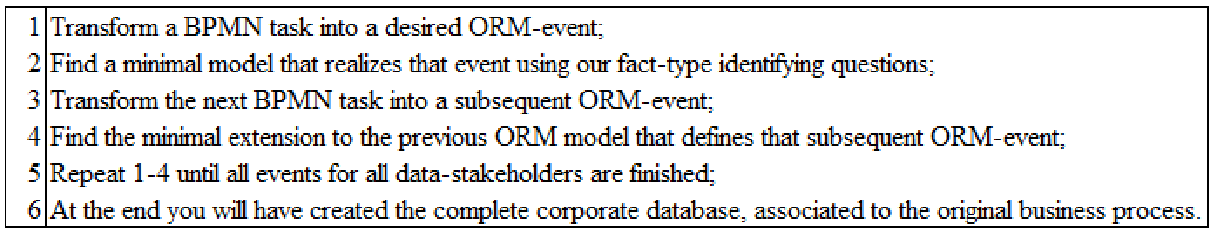

StartOfCoding
PK : startOfCodingNr : int
FK1 : coding:BankCardEntryNr : int
U1 : headEventName : varchar
U1 : instant : timestamp
Coding:BankCardEntry
PK : coding:BankCardEntryNr : int
U1, FK1 : accountNr : int
U1, FK1 : cardNr : int
U1 : instant : timestamp
CSF:IsPermitted : boolean
isLatest : boolean
stop:CSF1Nr : int
FK2 : userInput:PinEntryNr : int

Figure 2.9: Example of (partial) database blueprint for 'Bank Card Entry' (Balsters, 2014).

Appendix 1 shows an example of a complete database blueprint for a bank transaction (Balsters, 2014) and serves as an indication of what the database blueprint for this thesis should look like.

2.4.2 Roadmap

This paragraph discusses the roadmap of Balsters (2013) for modeling ORM diagrams. This roadmap consists of six steps and will be used as a guideline for this thesis project. The six questions are shown in table 2.2 below.



1	Transform a BPMN task into a desired ORM-event;
2	Find a minimal model that realizes that event using our fact-type identifying questions;
3	Transform the next BPMN task into a subsequent ORM-event;
4	Find the minimal extension to the previous ORM model that defines that subsequent ORM-event;
5	Repeat 1-4 until all events for all data-stakeholders are finished;
6	At the end you will have created the complete corporate database, associated to the original business process.

Table 2.2: Roadmap of Balsters (2013).

After completing the step-by-step process of the roadmap, the database blueprint is created and has to be validated by the end users at the LTHN.

3. Methodology

Chapter 3 discusses the phases and scope of the project, the methodology and validation of various steps. This chapter is divided into two main parts; the first part focuses on the overall project's methodology and the second part focuses on the thesis specific methodology.

3.1 The overall project

The project consists of two main parts. The first part focuses on the Design problem phase, Diagnosis/Analysis phase and the validation of the (BPMN) models. After the BPMN models are validated, they serve as an input for the start of the second part of the project; the Solution design phase. The solution design phase for this project entails the creation of an ontology for the BPMN models. The final part of the second phase is the validation of the database by presenting the database blueprints to the end users. If the database blueprints are not validated by the end users, or if they do not meet the requirements, they are adjusted based on the needs and requirements of the end users. If the end users do not accept the new solution, the project will be considered as a failure. This is the reason why this phase is very important for this project.

3.2 Overall research framework

As defined in chapter 2, this thesis is about a design science problem; the LTHN aims to create an artifact (database) in order to move from the current (undesirable) situation to a new desired situation. Wieringa (2007) states that design science deals with practical-knowledge problems; moving from an old or current situation to a new situation.

Van Strien (1997) acknowledges that there is a gap between science and practice. In order to minimize this gap, he developed the regulative cycle in 1997. The regulative cycle includes the following five phases:

1. Design problem;
2. Diagnosis/Analysis;
3. Design solution;
4. Implementation;
5. Validation.

Each of these phases are discussed in more detail in paragraph 3.2. Figure 3.1 shows the regulative cycle of Van Strien (1997).

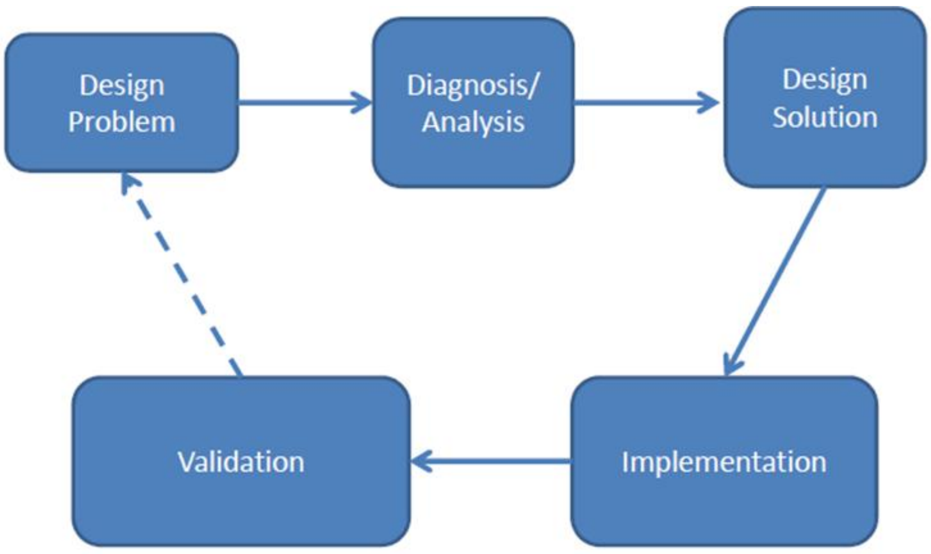


Figure 3.1: The regulative cycle of Van Strien (1997).

Based on the regulative cycle of Van Strien (1997), Balsters (2014) developed phase expansions for each phase. These phase expansions consist of multiple focus points and questions that should be answered in order to continue to the next phase. The phase expansions are:

1. Design problem. This phase focuses on analyzing the context of the problem. The main goal of this phase is to identify the stakeholders, their goals and the critical success factors (CSF's) of these goals.
2. Diagnosis/Analysis. The second phase of the regulative cycle focuses on identifying the causes for potential difficulties that occur when resolving the CSF's. It also deals with testing these causes by checking the quality attributes of the CSF's (e.g. how expensive, fast and safe does the solution has to be?). The last part of this phase expansion is identifying a potential order dependency of the CSF's (e.g. which attributes are more important than others).
3. Design solution. Phase 3 of the regulative cycle aims at identifying if there are alternative solutions available, if old solutions can be used and reinvented to a new solution or if a new solution has to be built from scratch.
4. Implementation. The realization of an artifact is considered to be as implementation. This phase is done by creating the database blueprints.
5. Validation. Validation of the database blueprints is done by presenting these to the end user. If the end user states that that something is missing (e.g. process step, stakeholder and check) then the regulative cycle starts again until the end user validates the design of the database blueprints.

Because our project is divided into two parts (first part being Design problem and Diagnosis/Analysis phase, second part being the Design solution phase, Implementation and Validation phases) there is an additional validation between these phases. Including the additional validation between the Diagnosis/Analysis phase and Design solution phase ensures that part two has validated input. The scope of the second part of the project, the Design solution, will only include the validated input of the BPMN models created by Beukeboom (2015) and Van de Laar (2015).

The Implementation and Validation phases focus on creating the database blueprints. These database blueprints will be validated by the end users at the LTHN and adjusted if needed.

3.3 Thesis specific methodology

This sub-chapter discusses the thesis specific methodology. The research structure is based on the cycle of Wieringa and Heerkens (2007) and consists of the following five phases:

1. Research problem identification (Chapter 1);
2. Research design (Chapter 2 and 3);
3. Research design validation (Chapter 2 and 3);
4. Execute research (Chapter 4);
5. Evaluate results (Chapter 5).

The following four steps for executing this specific research are formulated below, based on the literature review (chapter 2) and Fischer (2014).

1. Understand the Universe of Discourse (UoD);
2. Design ORM diagrams based on preliminary BPMN models;
3. Designing final ORM diagrams based on the end user validated BPMN models;
4. Critically review (the Fact-Type Identification questions of) the BPMN-ORM methodology.

3.3.1 Understand the Universe of Discourse (UoD)

The first step is getting familiar with the context and understanding the Universe of Discourse (UoD). In order to create an understanding of the UoD, the interviews held Beukeboom and Van de Laar with the stakeholders and domain experts for designing the BPMN models are visited.

3.3.2 Design ORM diagrams based on preliminary BPMN models

This step starts with designing an ORM diagram for the BPMN model of the general process. This BPMN model is a representation of the process on the highest level and contains several nested processes (sub-processes). After the first BPMN model is transformed into an ORM diagram, the nested processes are transformed into ORM diagrams and gradually incorporating them in the general process. The transformation from BPMN to ORM is done by using the roadmap (table 2.2) and is an iterative process. This means that if the BPMN models are updated with new information, the ORM diagrams are updated as well. These steps are repeated until every BPMN model for every process and sub-process is completed. Because of the relatively small time frame for this thesis, the ORM-Logic driven English (OLE) is not included. These are rules that accompany ORM diagrams (Balsters, 2013), for example:

T(Activity1 → Activity2) =
T(Activity1) **is followed by (at most one)** T(Activity2)

Building these rules is, due to the relative complexity and time needed for learning OLE, left outside the scope of this thesis. This thesis does however use the available OLE from Balsters (2014) as a guideline, without actually building these rules. These rules should be built in a later stadium for the actual implementation of this project.

3.3.3 Designing final ORM diagrams based on the end user validated BPMN models

The ORM diagrams are finalized based on the validated BPMN models. The general process and the nested processes are linked to each other in order to create the database blueprint.

3.3.4 Critically review the BPMN-ORM methodology

This step focuses on reviewing the BPMN-ORM methodology (14 FTI questions) of Balsters (2014) and runs parallel with the steps discussed in paragraphs 3.3.2 and 3.3.3. Input of Beukeboom (2015) and Van de Laar (2015) is also used for reviewing the methodology. Questions for reviewing the methodology are:

- Are questions formulated clearly and correctly?
- Is the order of questions correct?
- Are there questions that can be combined?
- Are there questions that need to be included?
- Are there questions that can be excluded?

After the ORM diagrams are finalized and the database blueprints are created the artifact is validated with the end user.

3.4 Data collection

The data collection of the first part starts with analyzing and collecting the Word documents that contain the data. The complete data overview is created with the help of the Bizagi website (www.bizagi.com) in order to build the BPMN models. Semi-structured interviews are conducted with the stakeholders in order to gain a deeper understanding of the data, processes, goals and CSF's. The validation of the BPMN models is done by meeting with the end users and stakeholders. The data gained from the interviews is primary data; these data is collected by the semi-structured interviews done by Van de Laar (2015) and Beukeboom (2015).

Data collection for the second part consists of collecting of the validated BPMN models of the first part of the project (secondary data). Semi-structured interviews with experts are held in order to correctly convert these data to the ontology and database blueprints (primary data). The validation of the database blueprints will be done by using semi-structured validation sessions with end users and stakeholders and presenting them the actual database blueprints (validation forms). Flaws in the database blueprints are adjusted until they are validated.

3.5 Validity and reliability

Karlsson (2009) states that the following tests must be addressed for any academic research that aims to be valid and reliable: construct validity, internal validity, external validity and reliability. The construct validity test is met by letting the interviewees review the drafts and conclusions. The drafts and conclusions are altered if interviewees disagree with them. This repeats until the interviewee states that they are correct and valid. The internal validity test is met by matching the outcomes of interviews and interviewees to identify possible causal relationship between them. External validity will be tested by looking whether or not the findings can be made generalizable (for other hospitals). Reliability of the overall research project is guaranteed by using a structured research approach and storing the results of the validation forms (sessions).

4. Results

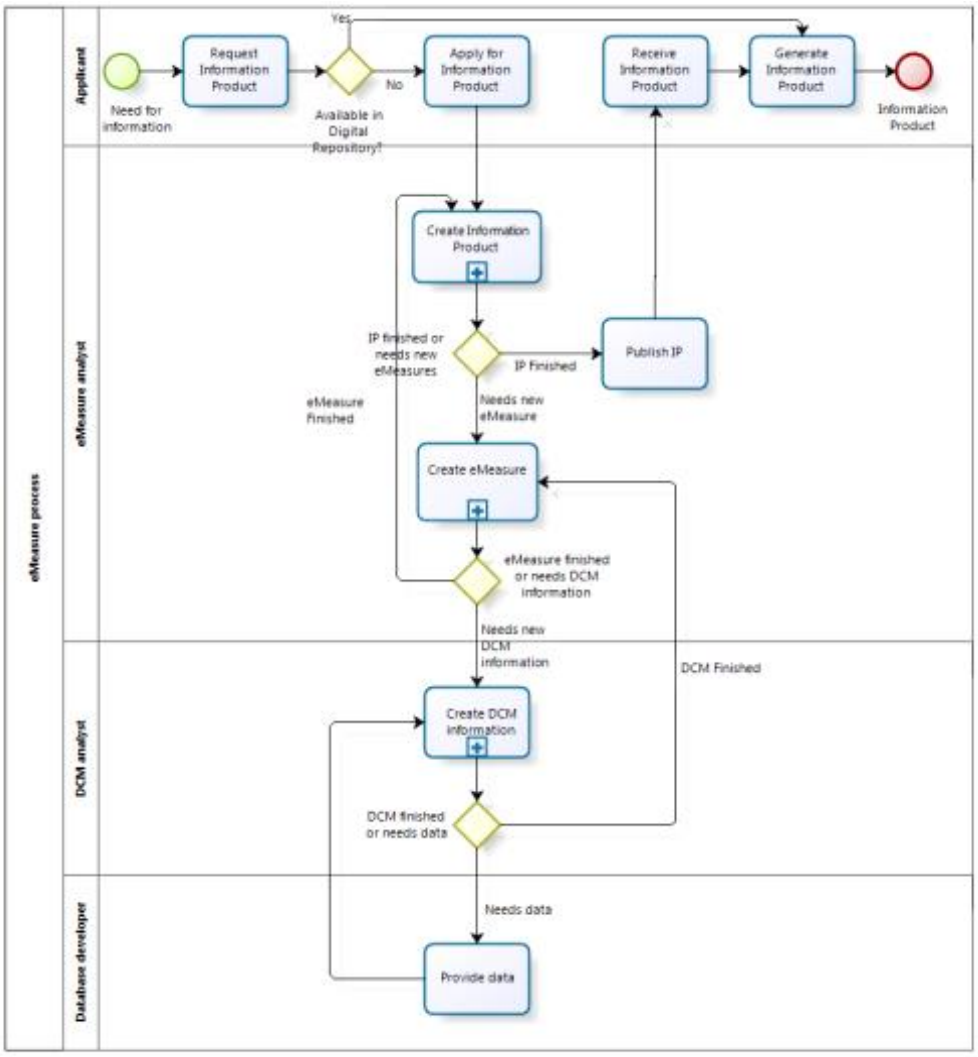
This chapter discusses the results of the various ORM diagrams that were created based on the BPMN-ORM methodology and will provide answers to the following sub-questions:

1. *"How can we convert end user validated BPMN models to an end user validated database blueprint?"*
2. *"How can the current database design method be improved and generalized?"*

This chapter is structured in the same way as the thesis specific methodology (chapter 3.3). Sub-chapters 4.1-4.3 provide an answer on sub-question 1; sub-chapter 4.4 will answer sub-question 2.

4.1 Understand the Universe of Discourse (UoD)

The UoD is shown in figure 4.1 and is a general overview of the process in BPMN. This model is based on the interviews with stakeholders and provides an understanding of the UoD.



eMeasure process
Applicant
eMeasure analyst
DCM analyst
Database developer
Need for information
Request Information Product
Available in Digital Repository?
Yes
No
Apply for Information Product
Receive Information Product
Generate Information Product
Information Product
Create Information Product
IP finished or needs new eMeasures
IP Finished
Publish IP
Needs new eMeasure
eMeasure Finished
Create eMeasure
eMeasure finished or needs DCM information
Needs new DCM information
DCM Finished
Create DCM information
DCM finished or needs data
Needs data
Provide data

Figure 4.1: BPMN model of the Universe of Discourse (UoD).

Figure 4.1 shows three nested processes (sub-processes); create Information Product (IP), create eMeasure and create DCM. Please refer for a more detailed explanation on information products, eMeasures and DCM's to the theses of Beukeboom (2015) and Van de Laar (2015). The UoD also shows the various stakeholders and activities that are relevant for this process. The creation of ORM diagrams starts with the general process BPMN model. As the project continuous, the nested processes are modeled into ORM and will (partially) replace the diagram of the UoD.

4.2 Design ORM diagrams based on preliminary BPMN models

This step started with designing an ORM diagram based on the general process BPMN model (top down approach was used for ORM modeling). The ORM diagrams for the nested processes are designed when the first part of the project progressed and are gradually incorporated in the general process ORM diagram. Because this is an iterative process, the intermediate ORM diagrams are not included. Only the final ORM diagrams are included in this thesis. These final ORM diagrams are discussed in the next paragraph and included in appendix 4.

4.3 Designing final ORM diagrams based on the end user validated BPMN models

As discussed in sub-chapter 4.2 a top down modeling approach is used for this thesis. This sub-chapter uses a relatively small part of the 'Create DCM' nested process that was modeled, based on the 14 FTI-questions and roadmap of Balsters (2013). The ORM diagrams are modeled based on the following assumptions:

1. All requests (e.g. information product, eMeasure) are single requests, there are no combined or multiple requests at the same instant (time);
2. All checks and instants (timestamps) are mandatory.

It must be noted that completely new processes are designed. This means that not all stakeholders know exactly what needs to be time stamped (logged) and if all checks should be mandatory. Therefore, assumption 2 has been made. Because the new processes are already relatively complex, the ORM diagrams are initially based on single requests. If, for example, an applicant would send a combined request (for two or more information products) the complexity for creating a new process would increase significantly. One information product might already be available, the second one might need a new eMeasure and a third request could need completely new data and/or DCM's. Therefore assumption 1 (treat all requests as single requests) for the creation of a completely new database was made.

4.3.1 Designing the actual ORM diagrams from the BPMN models

We will now zoom in on the 'Dummy-event' called 'Create DCM'. The 'Dummy-events' were used because the newly designed process contained nested processes with unknown structures at a certain point in time during the overall project. In order to cope with this, 'Dummy-events' were initially used (figure 4.1) so that progress of this thesis could be discussed with the supervisor and stakeholders. The 'Dummy-events' made it possible to start the modeling in a top down way, instead of having to wait until all (nested) processes were known and finalized. The ORM diagrams are modeled according the following 14 FTI-questions.

1	What is the event we are addressing?
2	Which stakeholders are involved?
3	What are the stakeholder goals?
4	What are the CSF's for each stakeholder goal in the context of this event?
5	Which objects are involved in the event as participants?
6	Which fact types are these participants engaged in?
7	Which constraints pertain to these fact types?
8	How do we identify the participants?
9	Which fact types is the event engaging in?
10	Which constraints pertain to these fact types?
11	How do we identify the event?
12	What are the input events for our particular event?
13	What do we have as output values of the event?
14	What are the associated business rules for these outputs?

Table 4.1: 14 Fact-Type Identification questions (Balsters, 2014).

It must be noted that there is an overlap regarding these questions with the theses of Van de Laar (2015) and Beukeboom (2015). For example, the questions regarding the stakeholders, their goals, CSF's and in- and outputs can also be found in their theses and BPMN models. This thesis uses those answers as input for modeling the ORM diagrams.

Semi-structured interviews were conducted by Van de Laar (2015) and Beukeboom (2015) in order to answer questions 1-5 and 12-14 and incorporate these into the BPMN models. The stakeholders, goals and CSF's in the BPMN models are then converted to ORM diagrams. The next page shows how these 14 questions are modeled 'Create DCM'.

Each of the following 14 questions are discussed individually and focus on explaining the reader what is precisely done. The focus points for certain questions are marked with blue circles.

Question 1: "What is the event we are addressing?"

As mentioned on the previous page, the event being addressed is 'Create DCM'. This event consists of two main events: 'Make candidate DCM' and 'Make final DCM'. Both of these are nested processes and are shown in figure 4.2. The paragraphs below discuss various parts of 'Make candidate DCM' (ORM event 1).

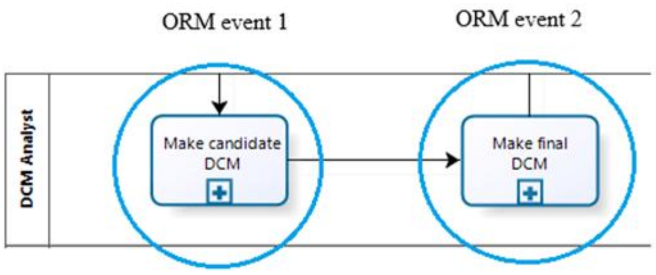


Figure 4.2: ORM event 1 (Make candidate DCM) and ORM event 2 (Make final DCM).

Question 2: "Which stakeholders are involved?"

Swimlanes represent the different stakeholders that are directly involved in the process. Figure 4.3 shows how a stakeholder (DCM Analyst) is converted from BPMN to ORM. This is repeated until all stakeholders of this particular event are converted to ORM.

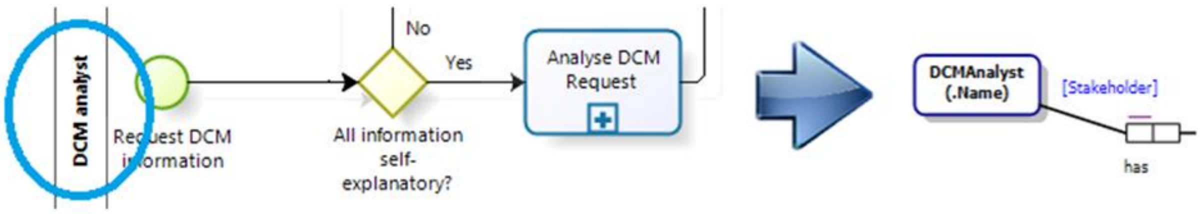


Figure 4.3: Converting a stakeholder from BPMN to ORM.

Question 3: "What are the stakeholder goals?"

The goal of the DCM analyst is to provide the required DCM information to another stakeholder (e.g. eMeasure analyst). This question is also repeated for all stakeholders. For a more detailed overview of the stakeholders, their goals and CSF's for 'Make candidate DCM', please refer to the thesis of Beukeboom (2015).

Question 4: "What are the CSF's for each stakeholder goal in the context of this event?"

A CSF is modeled in BPMN as a diamond (gateway) and modeled to ORM as a unary predicate (figure 2.4). Figure 4.4 shows how these are converted from BPMN to ORM.

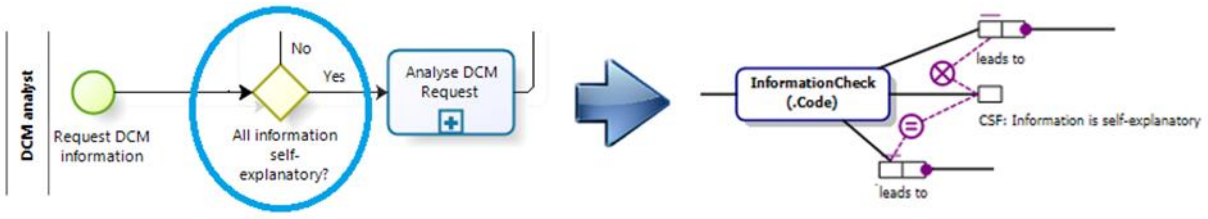


Figure 4.4: Converting a CSF from BPMN to ORM.

The CSF in figure 4.4 is that 'Information needs to be self-explanatory'. The equality constraint or exclusion constraint determines, based on the CSF, the next event (leads to).

Question 5: "Which objects are involved in the event as participants?"

Object-Role Modeling (ORM) is a *fact based modeling approach*. This means that ORM views the world as objects and roles (relationships). The objects are shown as rectangles with rounded edges in ORM. Figure 4.5 shows a number of objects that are involved in the 'Make candidate DCM' process. The identified stakeholder of question 2 is also included in figure 4.5.

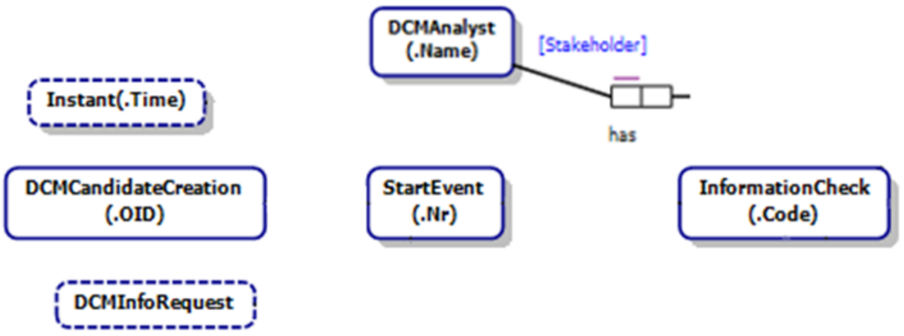


Figure 4.5: Various object examples of 'Make candidate DCM'.

Question 6: "Which fact types are these participants engaged in?"

According to Halpin and Morgan (2008), a fact type is a kind of fact that happens in a business domain. Let us take the objects 'DCMCandidateCreation' and 'StartEvent' as an example. Because the 'DCMCandidateCreation' is a nested process, it has to have its own start event (and stop event). The fact type becomes 'DCMCandidateCreation starts with StartEvent'.

Because the assumption has been made that everything in the newly designed process needs to be logged, the following fact types also apply to 'DCMCandidateCreation' and 'StartEvent': 'DCMCandidateCreation is at Instant(.Time)' and 'StartEvent is at Instant(.Time)'. Every start event and stop event needs to have a number, timestamp and description (Balsters, 2014). Examples of fact types that apply to the stop event in 'DCMCandidateCreation' are: 'StopEvent unnests to FinalDCMCreation' and 'StopEvent is at Instant(.Time)'. For a more detailed overview of all fact types for all processes please refer to appendix 4. Figure 4.8 shows the various fact types that apply to the objects of figure 4.5.

Question 7: "Which constraints pertain to these fact types?"

Halpin and Morgan (2008) state that there are various types of constraints in ORM. The most important constraints that are used during this research are uniqueness constraints, mandatory constraints, equality constraints and exclusion constraints. The purple dot on the roles in figure 4.6 are mandatory constraints. Assumption 2 states that all checks and instants (timestamps) are mandatory (e.g. every Start event must have a logged timestamp) and this is incorporated in all ORM diagrams. Figure 4.8 shows the various constraints that apply to the roles of figure 4.5.

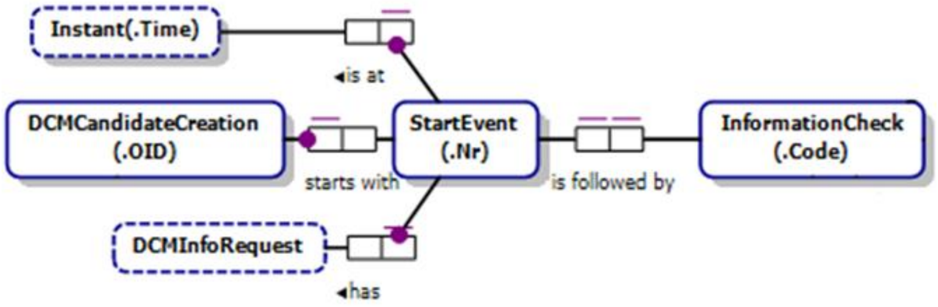


Figure 4.6: Mandatory constraints and uniqueness constraints.

The uniqueness constraints are also included in figure 4.6 and are shown by a relatively thin purple line above a role. For example, each start event is at exactly one instant in time. All build in constraints can be relatively easily checked in NORMA. NORMA includes a Verbalization Browser to check (with the end user) if the uniqueness constraints are applied correctly. Two examples of the Verbalization browser are included:

Example 1: StartEvent is at Instant(.Time).
Each StartEvent is at **exactly one** Instant(.Time).
It is possible that more than one StartEvent is at **the same** Instant(.Time).

Example 2: StartEvent is followed by InformationCheck.
Each StartEvent is followed by **at most one** InformationCheck.
For each InformationCheck, **at most one** StartEvent is followed by **that** InformationCheck.

As mentioned before, all checks are mandatory. However, all checks also have equality and exclusion constraints. Figure 4.7 shows how these constraints are modeled.

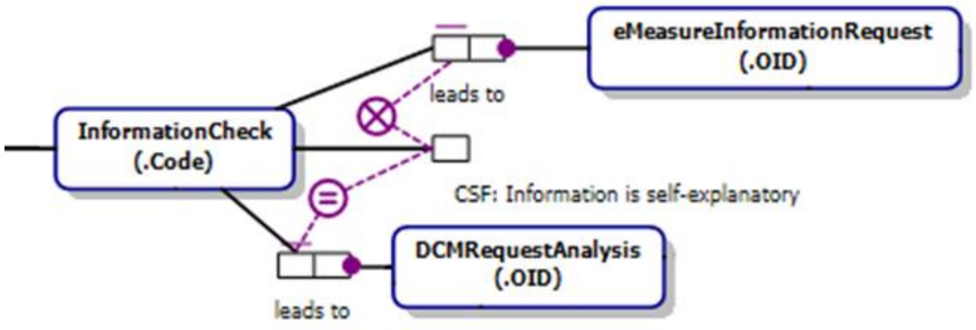

eMeasureInformationRequest (.OID)
leads to
InformationCheck (.Code)
CSF: Information is self-explanatory
DCMRequestAnalysis (.OID)
leads to

Figure 4.7: Equality and uniqueness constraints.

If the information check is met by the CSF: Information is self-explanatory, the process continues to the 'DCMRequestAnalysis'. If the CSF is not met, the process continues to the 'eMeasureInformationRequest'. Please note the mandatory constraints on the roles; this means that every 'eMeasureInformationRequest' and every 'DCMRequestAnalysis' must be preceded by the 'InformationCheck'.

Question 8: "How do we identify the participants?"

As mentioned in paragraph 2.3.2, the identification of the objects are displayed in parenthesis, also called the reference mode (Balsters, 2014). In order to obtain the actual identification of the various events and participants, semi-structured interviews with the technical domain expert (e.g. DCM analyst) were conducted. The results of the semi-structured interviews showed that there are various identification types for various events and participants. Stakeholders were initially identified by a .Name. This was however changed to by a .Nr because it is hospital specific; a name could be used in multiple hospitals or, even worse, the same name could be used in one hospital. The .Name identification was used as a default identification in figures 4.3 – 4.8 and in the final models changed to .Nr for identification.

After meeting with several domain experts it became clear that it was still unknown how Information Products should be identified. Because a new process is being designed, the domain experts did not known what an actual Information Products should look like or what should be included. In order to overcome this issue, the assumption has been made that for the time being, Information Products are given a .Name identification as default. This was agreed upon in the presence of various domain experts (e.g. DCM Analyst and eMeasure Analyst).

According to the eMeasure analyst, eMeasures are identified by Object Identifiers (OID), which consist of a root and a consecutive number: (e.g. 2.16.840.1.113883.2.4.3.8.1000.36:001). The identification of DCM's is according to the DCM analyst also done by using an OID. The DCM 'blood type' uses for example the following OID and consecutive number: 2.16.840.1.113883.2.4.3.8.1000.36:{9146E8C3-A236-4010-A6A2-FF8F9D024EFF}.

Valuesets are also identified by an OID. The 'DrugGebruikCodeLijst', or in English 'DrugUseCodeList' uses the following OID: 2.16.840.1.113883.2.4.3.11.60.39.11.65. The attributes are identified by a (SNOMED-CT) code. Misuses drugs (findings) uses the following (SNOMED-CT) code: 361055000. The instants are identified by a timestamp, which consists of a date (day, month, and year) and time.

In conclusion, stakeholders are identified by .Nr, Information Products are identified by a .Name, eMeasures, DCM's and Valuesets are identified by a .OID, attributes are identified by .Code and instants are identified by .Time (date and time). Figure 4.8 shows the objects with their respective identifications (.OID, .Time, .Code .Name and .Nr).

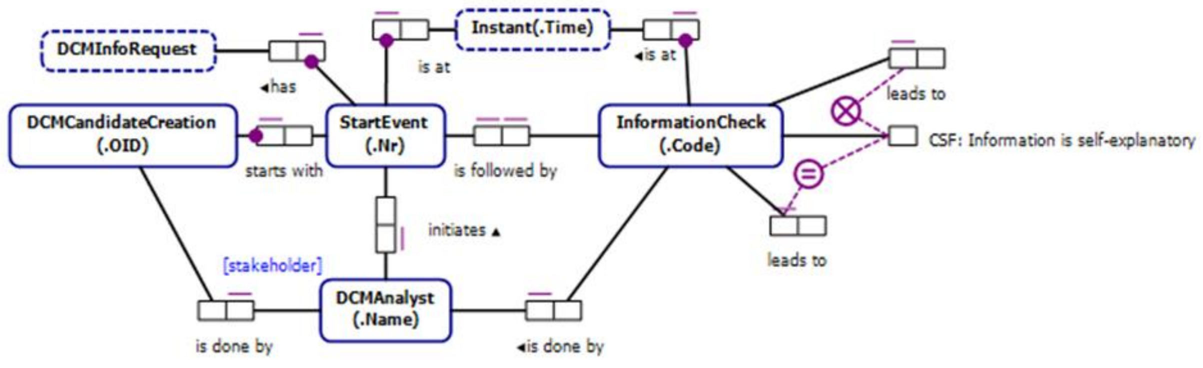


Figure 4.8: 'Make candidate DCM' with fact types, constraints and identifications.

We will now discuss questions 9, 10 and 11:

Question 9: "Which fact types is the event engaged in?"

Question 10: "Which constraints pertain to these fact types?"

Question 11: "How do we identify the event?"

These questions have been combined with questions 6, 7 and 8 for this research. This decision was made because it was more efficient to plan the semi-structured interviews with the stakeholders. Combining the questions gave the opportunity to have one interview rather than two for discussing the fact types, constraints and identifications for both the events and participants. The questions were combined for this research into the following questions:

Question 6 and 9: "Which fact types are the event and participants engaged in?"

Question 7 and 10: "Which constraints pertain to these fact types?"

Question 8 and 11: "How do we identify the event and participants?"

Combining these questions saved a considerable amount of time and number of interviews. The last three questions focus on the input, output and associated business rules for the outputs.

Question 12: "What are the input events for our particular event?"

This question addresses the input events for the particular event. It is also possible that a single event uses multiple input events. The input events for all events were identified by analyzing the BPMN models and looking what the specific required input events are for the particular events. For example, the 'CreateCandidateDCM'-event uses 'DCMInformationRequest' as an input event. This question is also repeated for all events.

Question 13: "What do we have as output values of the event?"

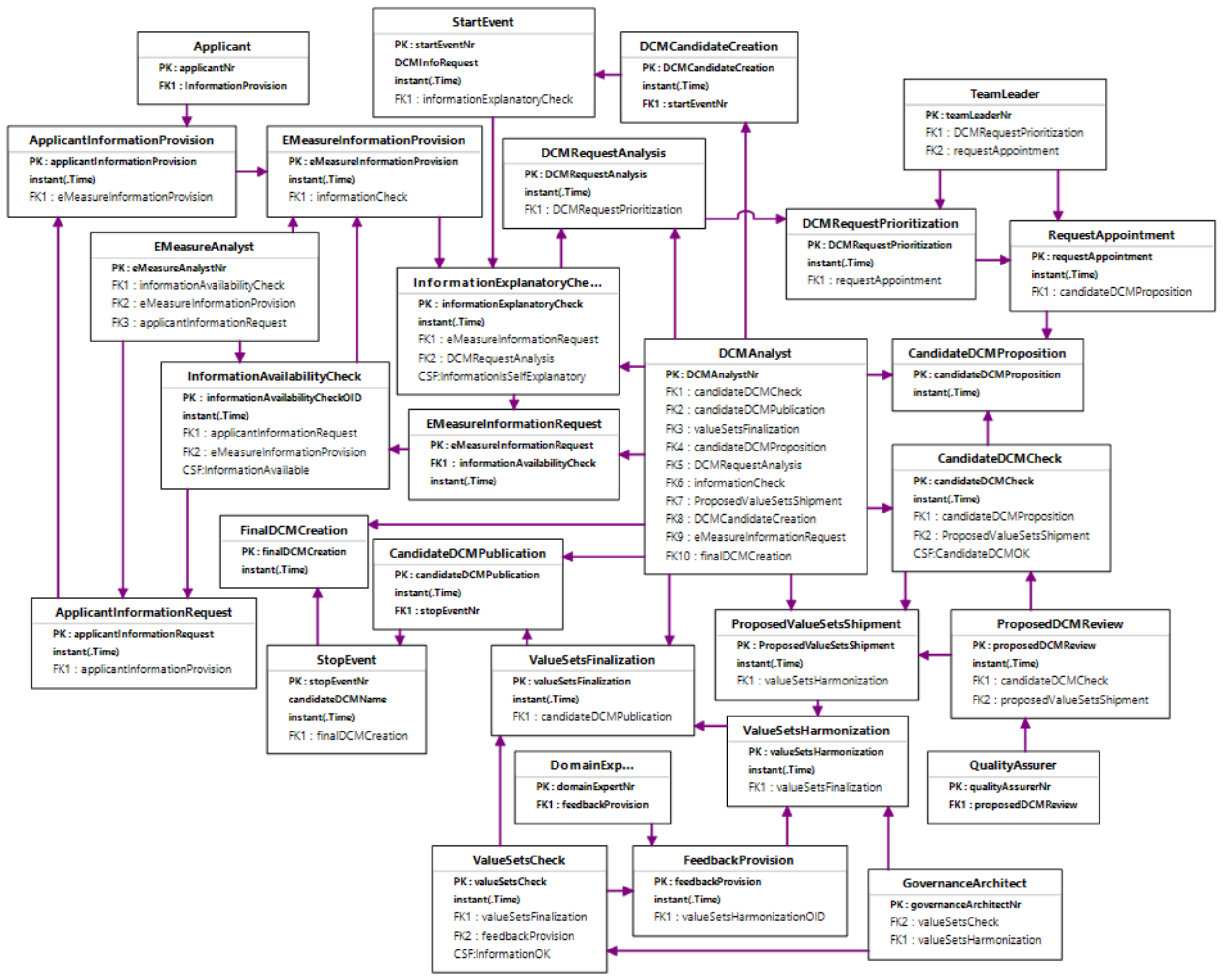
This question addresses the output for the particular event. Output values of an event can for example be a mapped DCM, a published candidate DCM, a published final DCM and etcetera. These output values were identified by Beukeboom (2015) and Van de Laar (2015) and modeled into the BPMN models. The output value of one event determines the next event of the process.

Question 14: "What are the associated business rules for these outputs?"

It must be noted that the overarching project focuses on the design of a completely new process and artifact. The truth is that a lot of factors and business rules are still relatively unknown at this point in time. However, after several meetings it did became clear that the current set of standards used at the LTHN must be according to Health Level 7 (HL-7); which support the processes in hospitals (Jaffe et al., 2009). All information products, eMeasures and DCM's need to meet the HL-7 standards. This not only a standard for the LTHN but an international (accepted) standard for all hospitals.

The second identified business rule is that all of the process steps need to be performed by authorized personnel in order to make sure that all is done according the HL-7 standard. The last identified business rule is that value sets must meet the governance architect's requirements.

The 14 questions are repeated until the database was completed. The database blueprints are combined with the BPMN models into 'Validation forms' (sub-chapter 4.5). Figure 4.9 shows a part of the final database blueprint of 'Create Candidate DCM', which is a nested process. All of the final ORM diagrams with their corresponding database blueprints can be found in appendix 4.



Applicant
PK : applicantNr
FK1 : InformationProvision
StartEvent
PK : startEventNr
DCMInfoRequest
instant(.Time)
FK1 : informationExplanatoryCheck
DCMCandidateCreation
PK : DCMCandidateCreation
instant(.Time)
FK1 : startEventNr
TeamLeader
PK : teamLeaderNr
FK1 : DCMRequestPrioritization
FK2 : requestAppointment
ApplicantInformationProvision
PK : applicantInformationProvision
instant(.Time)
FK1 : eMeasureInformationProvision
EMeasureInformationProvision
PK : eMeasureInformationProvision
instant(.Time)
FK1 : informationCheck
DCMRequestAnalysis
PK : DCMRequestAnalysis
instant(.Time)
FK1 : DCMRequestPrioritization
DCMRequestPrioritization
PK : DCMRequestPrioritization
instant(.Time)
FK1 : requestAppointment
RequestAppointment
PK : requestAppointment
instant(.Time)
FK1 : candidateDCMProposition
EMeasureAnalyst
PK : eMeasureAnalystNr
FK1 : informationAvailabilityCheck
FK2 : eMeasureInformationProvision
FK3 : applicantInformationRequest
InformationExplanatoryChe...
PK : informationExplanatoryCheck
instant(.Time)
FK1 : eMeasureInformationRequest
FK2 : DCMRequestAnalysis
CSF:InformationIsSelfExplanatory
DCMAnalyst
PK : DCMAnalystNr
FK1 : candidateDCMCheck
FK2 : candidateDCMPublication
FK3 : valueSetsFinalization
FK4 : candidateDCMProposition
FK5 : DCMRequestAnalysis
FK6 : informationCheck
FK7 : ProposedValueSetsShipment
FK8 : DCMCandidateCreation
FK9 : eMeasureInformationRequest
FK10 : finalDCMCreation
CandidateDCMProposition
PK : candidateDCMProposition
instant(.Time)
InformationAvailabilityCheck
PK : informationAvailabilityCheckOID
instant(.Time)
FK1 : applicantInformationRequest
FK2 : eMeasureInformationProvision
CSF:InformationAvailable
EMeasureInformationRequest
PK : eMeasureInformationRequest
FK1 : informationAvailabilityCheck
instant(.Time)
CandidateDCMCheck
PK : candidateDCMCheck
instant(.Time)
FK1 : candidateDCMProposition
FK2 : ProposedValueSetsShipment
CSF:CandidateDCMOK
FinalDCMCreation
PK : finalDCMCreation
instant(.Time)
CandidateDCMPublication
PK : candidateDCMPublication
instant(.Time)
FK1 : stopEventNr
ApplicantInformationRequest
PK : applicantInformationRequest
instant(.Time)
FK1 : applicantInformationProvision
ProposedValueSetsShipment
PK : ProposedValueSetsShipment
instant(.Time)
FK1 : valueSetsHarmonization
ProposedDCMReview
PK : proposedDCMReview
instant(.Time)
FK1 : candidateDCMCheck
FK2 : proposedValueSetsShipment
StopEvent
PK : stopEventNr
candidateDCMName
instant(.Time)
FK1 : finalDCMCreation
ValueSetsFinalization
PK : valueSetsFinalization
instant(.Time)
FK1 : candidateDCMPublication
ValueSetsHarmonization
PK : valueSetsHarmonization
instant(.Time)
FK1 : valueSetsFinalization
DomainExp...
PK : domainExpertNr
FK1 : feedbackProvision
QualityAssurer
PK : qualityAssurerNr
FK1 : proposedDCMReview
ValueSetsCheck
PK : valueSetsCheck
instant(.Time)
FK1 : valueSetsFinalization
FK2 : feedbackProvision
CSF:InformationOK
FeedbackProvision
PK : feedbackProvision
instant(.Time)
FK1 : valueSetsHarmonizationOID
GovernanceArchitect
PK : governanceArchitectNr
FK2 : valueSetsCheck
FK1 : valueSetsHarmonization

Figure 4.9: Database blueprint of 'Make Candidate DCM'.

For readability purposes, the names of the primary keys (PK) and foreign keys (FK) are altered a little bit so that they are more understandable for the end user (e.g. ValueSetsCheckOID was changed to ValueSetsCheck) and DisplayDataTypes in the properties window in NORMA was changed from 'True' to 'False' (e.g. FK1: valueSetsCheckOID was changed to FK1: valueSetsCheck).

4.4 Critically review the BPMN-ORM methodology

This step focuses on reviewing the BPMN-ORM methodology (14 FTI questions) of Balsters (2014) and runs parallel with the steps discussed in paragraphs 3.3.2 and 3.3.3. Input of Beukeboom (2015) and Van de Laar (2015) is also used for reviewing the methodology. The methodology is reviewed from both a scientific and 'end user' perspective. Questions for reviewing the BPMN-ORM methodology are:

- Are questions formulated clearly and correctly?
- Is the order of questions correct?
- Are there questions that can be combined?
- Are there questions that need to be included?
- Are there questions that can be excluded?

In regard to the first question (*"Are questions formulated clearly and correctly?"*) the modeler did not have any issues regarding the formulation of the 14 FTI-questions. They are considered to be straightforward and easy to understand for anyone that has (some) knowledge or experience with BPMN and ORM.

The second question (*"Is the order of questions correct?"*) focuses on the fact if the actual models are built in a logical way, based on the 14 FTI-questions. The current methodology starts with identifying the stakeholders for an ORM diagram and the diagram becomes more detailed as more of the FTI-questions are answered in the current particular order. The current order is considered to be correct.

The third question (*"Are there questions that can be combined?"*) is a different story. As stated in sub-chapter 4.3, the following FTI-questions were combined for this research:

- *Question 6 and 9: "Which fact types are the event and participants engaged in?"*
- *Question 7 and 10: "Which constraints pertain to these fact types?"*
- *Question 8 and 11: "How do we identify the event and participants?"*

The first consideration to combine these questions was that it proved to be a more efficient way to plan the semi-structured interviews with the stakeholders. Combining these questions provided the opportunity to have one interview rather than two for discussing the fact types, constraints and identifications for both the events and participants and saved a considerable amount of time and number of interviews. Another benefit is that it is easier for the modeler to model the answers of these combined questions in one go. For example, if questions 7 and 10 are not combined then the answers of question 7 are first modeled which might contain 60-80% of the fact types of a model. The modeler must then answer questions 8 and 9 before reaching question 10 and finish the last % of the fact types. From an end user or modeler perspective, it is a lot more user-friendly to combine the questions and model all of the fact types before moving to another question (e.g. doing constraints first and then back to the fact types again).

After using the BPMN-ORM methodology and critically reviewing the fourth question (*"Are there questions that need to be included?"*) it became clear that it does not explicitly address the validation of the designed ORM diagrams. Before the project was changed (see more on this in chapter 5), the validation of the models became unclear. Therefore this research suggests to include the following question in the BPMN-ORM methodology: *"How can we validate our model?".* There are various ways to validate the ORM diagrams like the NORMA Verbalization browser (sub-chapter 4.3) and User Interface mock ups. This research used validation forms which basically are forms that include the BPMN model and (a part of) the corresponding database blueprint. This technique makes it relatively easy for the domain expert and end user to validate the model/database blueprint. These validation forms are discussed in sub-chapter 4.5 and propose a relatively easy method in order to validate the models.

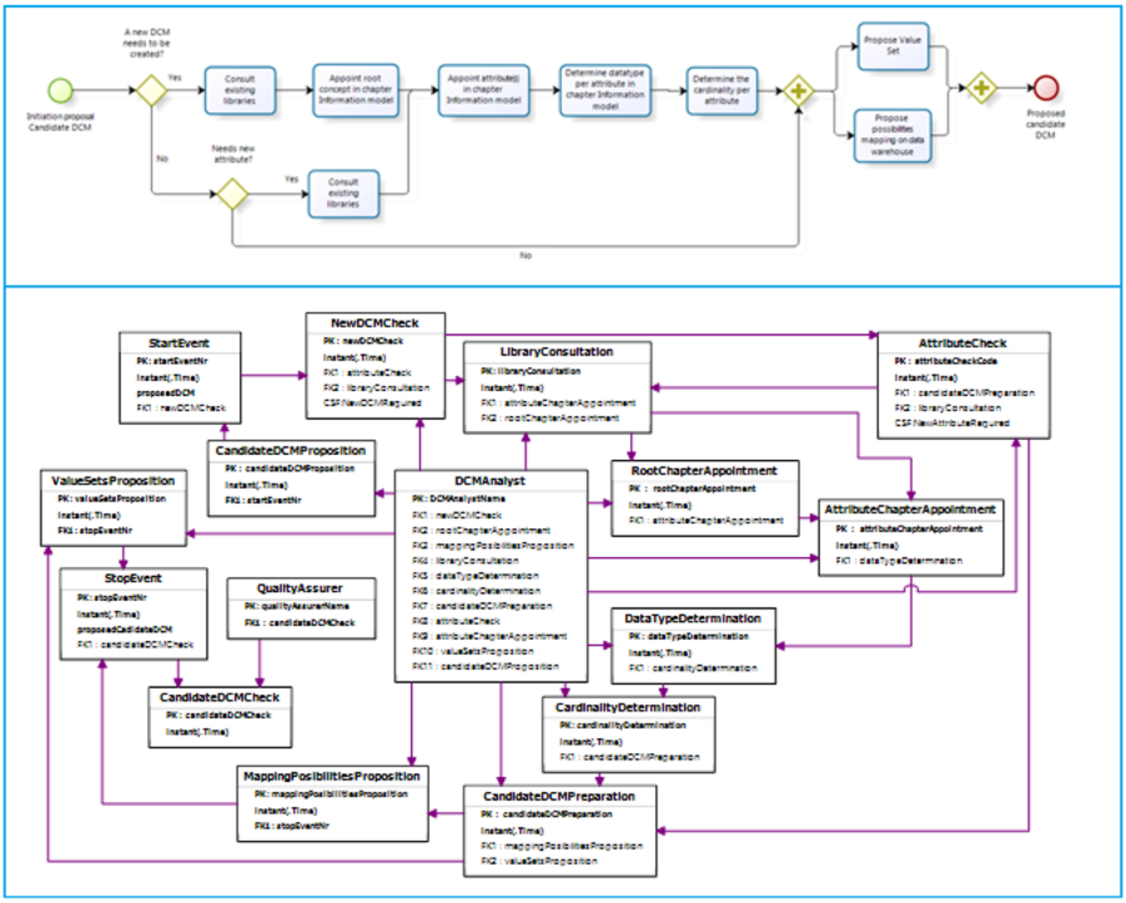
The final question for the critical review of the BPMN-ORM methodology (*"Are there questions that can be excluded?"*) is the result of combining the questions stated on the previous pages. By combining these questions it is possible to remove three questions (9, 10 and 11) and using them in a more effective way. The number of steps in the BPMN-ORM methodology can be reduced from 14 to 12 (including *"How can we validate our model?"*-question). Other questions should not be removed because it will negatively influence the user-friendliness of the methodology. Table 4.2 shows the proposed BPMN-ORM methodology.

1	What is the event we are addressing?
2	Which stakeholders are involved?
3	What are the stakeholder goals?
4	What are the CSF's for each stakeholder goal in the context of this event?
5	Which objects are involved in the event as participants?
6	Which fact types are the event and participants engaged in?
7	Which constraints pertain to these fact types?
8	How do we identify the event and participants?
9	What are the input events for our particular event?
10	What do we have as output values of the event?
11	What are the associated business rules for these outputs?
12	How can we validate our model?

Table 4.2: Proposed 12 Fact-Type Identification questions.

4.5 Validation forms

The database blueprints are validated with the end user and domain expert by showing them so called 'Validation forms' that contain both the BPMN model and the corresponding database blueprint. At the validation session it was explicitly told that, if applicable, database blueprints were nested processes and that these nested processes have their own start and stop events. It was also told that the stop events unnest to the next process step (e.g. 'StopEvent unnests to CandidateDCMCheck'). Figure 4.10 shows a version of the validation form 'Propose Candidate DCM'.



Initiation proposal Candidate DCM
A new DCM needs to be created?
Yes
No
Consult existing libraries
Appoint root concept in chapter Information model
Appoint attribute(s) in chapter Information model
Determine datatype per attribute in chapter Information model
Determine the cardinality per attribute
Propose Value Set
Propose possibilities mapping on data warehouse
Proposed candidate DCM
Needs new attribute?
Yes
Consult existing libraries
No
StartEvent
PK: startEventNr
Instant(.Time)
proposedDCM
FK1 : newDCMCheck
NewDCMCheck
PK : newDCMCheck
Instant(.Time)
FK1 : attributeCheck
FK2 : libraryConsultation
CSF:NewDCMRequired
LibraryConsultation
PK: libraryConsultation
Instant(.Time)
FK1 : attributeChapterAppointment
FK2 : rootChapterAppointment
AttributeCheck
PK : attributeCheckCode
Instant(.Time)
FK1 : candidateDCMPreparation
FK2 : libraryConsultation
CSF:NewAttributeRequired
CandidateDCMProposition
PK : candidateDCMProposition
Instant(.Time)
FK1 : startEventNr
ValueSetsProposition
PK: valueSetsProposition
Instant(.Time)
FK1 : stopEventNr
DCMAnalyst
PK: DCMAnalystName
FK1 : newDCMCheck
FK2 : rootChapterAppointment
FK3 : mappingPosibilitiesProposition
FK4 : libraryConsultation
FK5 : dataTypeDetermination
FK6 : cardinalityDetermination
FK7 : candidateDCMPreparation
FK8 : attributeCheck
FK9 : attributeChapterAppointment
FK10 : valueSetsProposition
FK11 : candidateDCMProposition
RootChapterAppointment
PK : rootChapterAppointment
Instant(.Time)
FK1 : attributeChapterAppointment
AttributeChapterAppointment
PK : attributeChapterAppointment
Instant(.Time)
FK1 : dataTypeDetermination
StopEvent
PK: stopEventNr
Instant(.Time)
proposedCadidateDCM
FK1 : candidateDCMCheck
QualityAssurer
PK: qualityAssurerName
FK1 : candidateDCMCheck
DataTypeDetermination
PK: dataTypeDetermination
Instant(.Time)
FK1 : cardinalityDetermination
CandidateDCMCheck
PK : candidateDCMCheck
Instant(.Time)
CardinalityDetermination
PK: cardinalityDetermination
Instant(.Time)
FK1 : candidateDCMPreparation
MappingPosibilitiesProposition
PK: mappingPosibilitiesProposition
Instant(.Time)
FK1 : stopEventNr
CandidateDCMPreparation
PK : candidateDCMPreparation
Instant(.Time)
FK1 : mappingPosibilitiesProposition
FK2 : valueSetsProposition

Figure 4.10: Validation form of Propose Candidate DCM'.

In order to validate the models, validation sessions were planned with the end users/domain experts in order to discuss all database blueprints. Focus points of these (semi-structured) validation sessions were the process flows, process steps, correctness of stakeholders and readability of the diagrams.

Chapter 5 reflects on how the overarching project and research were conducted. The drawbacks of the project and research are also discussed.

5. Discussion

This chapter discusses the results of chapter 4 and also reflects on how the overarching project and research were conducted. The drawbacks of the project and research are also discussed.

The database blueprints were designed according to a top down approach. The ORM diagrams for the nested processes are designed when the first part of the project progressed and were gradually incorporated in the top level model. 'Dummy-events' were used in ORM because the newly designed process contained nested processes with unknown structures at a certain point in time during the overall project. These 'Dummy-events' made it possible to discuss the progress of this thesis with the supervisor and stakeholders. The 'Dummy-events' made it possible to start the modeling in a top down way, instead of having to wait until all (nested) processes were known and finalized. Once the structures of the nested processes were known, they replaced the 'Dummy-events' in ORM. Literature did not provide a solution on how to deal with nested processes with unknown structures in ORM; only on how to deal with nested processes that include known structures. Another possibility to overcome this issue is to create an option in NORMA that is similar to the nested processes in BPMN. BPMN has the option to include a nested process and to zoom in on its respective structure by simply clicking on it. It is then possible to build the structure of the nested process and automatically link these.

When looking at the original BPMN-ORM methodology, it is easy to see that there are a couple of relatively similar questions in the methodology (e.g. questions 6 and 9, 7 and 10, 8 and 11). Combining these questions proved to be a more efficient way to plan the interviews and to design the ORM diagrams. From an end user or modeler perspective, it is a lot more user-friendly to combine the questions and model all of the identifications, constraints or fact types before moving to another question (e.g. doing fact types from question 7 first, move to constraints of question 8 and then at question 10 back to the fact types again). Both efficiency and user-friendliness (modeler) are considered to be great advantages that can be achieved by combining these questions into the proposed questions of chapter 4.4.

One of the major practical difficulties is that validation of end user validated BPMN models took relatively a lot of time, which affected the time to complete the ORM diagrams. Because the overarching project focused on the design of a complete new process, BPMN models were often changed which resulted in a relatively lot of rework for the ORM diagrams and corresponding database blueprints. It is recommend for similar future projects to include tollgates so that certain process steps can start at a predefined moment in time.

Another practical difficulty that also was identified by Hoekstra (2014) is that there is currently no official 'rule' for the scope of an ORM diagram (e.g. which BPMN parts should be included in one ORM diagram). This thesis suggests to introduce a 'rule of thumb' for the scope of a single ORM diagram. The 'rule of thumb' used for this project is that the scope of a single ORM diagram should still fit on one page. Since there is currently a lack of an official rule stated in literature; this 'rule of thumb' is a fairly easy to apply rule for (new) ORM modelers.

A third, relatively small, practical difficulty was the identification of information products. Because a new process has been designed, nobody at the LTHN knew how the information products should be identified. After meeting with the domain experts, the assumption has been made that information products are identified by a .Name. As a result of the first meetings with the stakeholders, the initial project was changed. The use of User Interface mock ups for the validation of the models as described in the proposal was completely scrapped. Instead the database blueprints were combined with BPMN models into 'validation forms', which proved a relatively easy and quick method to validate the models.

Validation is not explicitly included in the original BPMN-ORM methodology, but there are a number of methods possible (e.g. NORMA Verbalization browser, User Interface mock ups). The type of validation method might depend on the type of process that needs to be designed. This research uses validation forms that contain both the BPMN models and (parts of) the corresponding database blueprint (depending on the model's complexity). Because the original BPMN-ORM methodology does not include a validation question and the fact that there are various validation methods available, an extra question was added to the methodology. The ORM diagrams were changed according to the feedback received based on the validation forms. For validation purposes, the blueprints were made more clear (DisplayDataTypes "false") and names are made more clear (e.g. FK2:FinalDCMCheckOID to FK2: FinalDCMCheck). By using BPMN end user validated models and end user validation of the database blueprints, all database blueprints are considered to be validated.

The validation forms contained database blueprints with 'Dummy-events'. The final blueprints include nested process with known structures. End users were explicitly told that, if applicable, database blueprints were nested processes and that these nested processes have their own start and stop events. The end users were also told that the stop events unnest to the next process step (e.g. 'StopEvent unnests to CandidateDCMCheck'). Every (nested) structure was validated individually so that it was more understandable for the end user.

6. Conclusions

The overarching project consists of two main parts. The first part focuses on the creation and validation of the BPMN models for the design of the new process. After the BPMN models are validated, they serve as an input for the start of the second part of the project. The second part of this project entails the creation of an ontology for the BPMN models and the validation of the database by presenting the validation forms to the end users and domain experts.

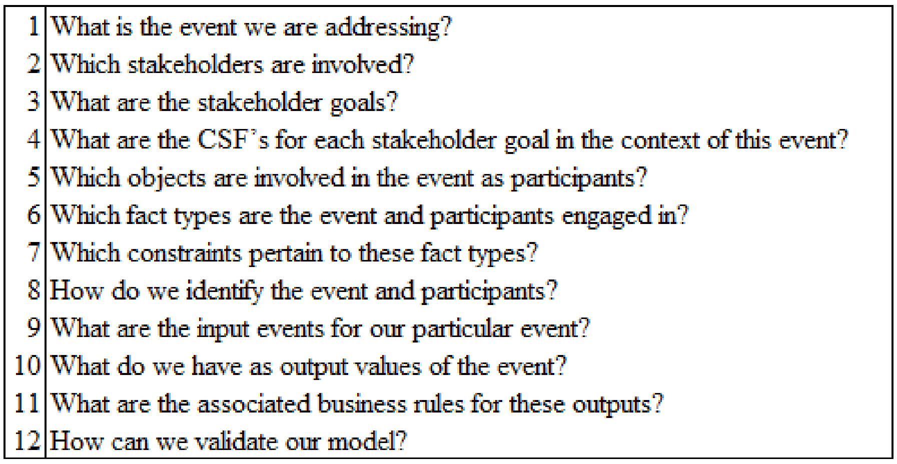
This particular part of the overarching project focused on the second part of the overarching project. The scientific relevance of this thesis is the validation of the BPMN-ORM methodology. This thesis shows how data can be derived from BPMN models and answers the main research question:

"How can an end user validated database blueprint be derived from the BPMN models within the context of an EHR system at the LTHN?"

The main research question consists of the two following sub-questions:

1. *"How can we convert end user validated BPMN models to an end user validated database blueprint?"*
2. *"How can the current database design method be improved and generalized?"*

After critically reviewing the 14 FTI-questions, research showed that relatively small changes should be made to the BPMN-ORM methodology which answers both sub-questions. This leads to the following proposed BPMN-ORM methodology:



1	What is the event we are addressing?
2	Which stakeholders are involved?
3	What are the stakeholder goals?
4	What are the CSF's for each stakeholder goal in the context of this event?
5	Which objects are involved in the event as participants?
6	Which fact types are the event and participants engaged in?
7	Which constraints pertain to these fact types?
8	How do we identify the event and participants?
9	What are the input events for our particular event?
10	What do we have as output values of the event?
11	What are the associated business rules for these outputs?
12	How can we validate our model?

Table 6.1: Proposed BPMN-ORM methodology.

The proposed changes from the original BPMN-ORM methodology are to combine various questions and to add in a new question regarding the validation of the designed models.

6.1 Research limitations

The limitations of this research will discussed individually. The first limitation is that the research was done at one of the biggest university hospitals in the Netherlands with lots of processes and a high level of environmental complexity. This research focused on converting the BPMN models for the creation of information products, eMeasures and DCM's to ORM diagrams. The results of this research apply to other teaching hospitals in the Netherlands. Smaller, non-university hospitals often have less processes and a less complex environment. This means that the results of this research also apply to the smaller, non-university hospitals if they are authorized to create information products, eMeasures and DCM's. The results of this research also apply to BPMN models in general; other processes within the LTHN (e.g. requests for financial information products) are not studied but can also modeled in BPMN. This is considered to be the second limitation; this particular research only included the BPMN models of Beukeboom (2015) and Van de Laar (2015).

6.2 Further work

The ORM diagrams of this thesis were validated by using validation forms that contain the BPMN model and (parts of) the corresponding database blueprint. The first recommendation for further work is to develop User Interface mock ups to validate whether or not the right data can be extracted relatively easy from the database blueprints (from an end user's perspective). The User Interface mock ups can consist of sketches on A4 paper that show the potential interface screens that include both the sequence of the data models and the data required by the end user.

A second recommendation for further work is to validate the proposed BPMN-ORM methodology in a complete different setting (e.g. nonhospital setting) or in a different setting at the LTHN (e.g. dental procedures, exploratory surgery procedures or financial procedures). It is expected that testing the proposed BPMN-ORM methodology in a different setting will improve its applicability.

A final, more pragmatic suggestion for further work is to look at the opportunity to implement nested processes with unknown structures in NORMA. A possibility to overcome this issue (including nested processes with unknown structures without using 'Dummy-events') is to create an option in NORMA that is similar to the nested processes in BPMN. BPMN has the option to include a nested process and to zoom in on its respective structure by simply clicking on it. It is then possible to build the structure of the nested process and to reduce the level of complexity for the modeler.

References

Akmal, S., Shih, L. & Batres, R (2014). Ontology-based similarity for product information retrieval. ***Computers in Industry***, 65(1): 91-107.

Balsters, H. (2013). Mapping BPMN process models to data models in ORM, ***Lecture Notes in Computer Science***, 1841.

Balsters, H. (2014). Abstract data from process. 1-13.

Balsters, H., 2013b. Mapping BPMN process models to ORM data models, ***Lecture Notes in Computer Science***, 8186.

Balsters, H. (2014). Design science; BankTransSept21. 1-36.

Balsters, H. (2014). Design science; Lecture slides in Research methods. 1-21.

Balsters, H. (2014). Inleiding informatiesystemen; Lecture slides of lectures 1-9; 1-202.

Beukeboom, R. T. (2015). Design of the process of developing DCM's with regard to eMeasures. University of Groningen. The Netherlands.

Castells, P., Fernández, M. & Vallet, D. (2007). An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. ***IEEE Transactions on Knowledge & Data Engineering***, 19(2): 261-272.

Chinosi, M., & Trombetta, A. (2012). BPMN: An introduction to the standard. ***Computer standards & interfaces***, 34(1): 124-134.

Dijkman, R. M., Dumas, M. & Ouyang, C. (2008). Semantics and analysis of business process models in BPMN. ***Information and Software Technology***, 50(12): 1281–1294.

Fischer, R. A. (2014). Validation of a Process-driven Database Design Method for an Electronic Health Record-system: from process models to data models. University of Groningen. The Netherlands.
Halpin, T. & Morgan, T., 2008. Information Modeling and Relational Databases 2nd edition, Morgan Kaufmann Publishers. Burlington, United States of America.

Hevner, A. R., March, S. T., Park, J., Ram, S. (2004). Design science in information systems research. ***MIS Quarterly***, 28(1): 75-105.

Jaffe, C., Hammond, W. E., Quinn, J. & Dolin, R. H. (2009). Healthcare IT standards and the standards development process: lessons learned from Health Level 7. ***Intel Technology Journal***, 13(3): 58-79.

Kabel, S., Hoog, R. de, Wielinga, B. J. & Anjewierden, A. (2004). The Added Value of Task and Ontology-Based Markup for Information Retrieval. ***Journal of the American Society for Information Science & Technology***, 55(4): 348-362.

Karlsson, C. (2009). Researching Operations Management, Routledge: New York.

Noy, N., & McGuinness, D. (2001). Ontology development 101: A guide to creating your first ontology. ***Development***, (32): 1-25.

Okma, K. G. H., & Crivelli, L. (2013). Swiss and Dutch "consumer-driven health care": ideal
model or reality? ***Health policy***, 109(2): 105–12.

Recker, J. (2010). Opportunities and constraints: the current struggle with BPMN. ***Business Process Management Journal***, 16(1): 181-201.

Strien, P. J. van. (1997). Towards a Methodology of Psychological Practice: The Regulative Cycle. ***Theory & Psychology***, 7(5): 683-700.

Troseth, M. (2010). Standardization will be key. ***Health Management Technology***, 31(1): 8-18.

Vanberkel, P. T., Boucherie, R. J., Hans, E. W., Hurink, J. L., & Litvak, N. (2010). Efficiency
evaluation for pooling resources in health care. ***Or Spectrum***, 34(2): 371–390.

Van de Laar, P. J. C. (2015). Creating a general method for eMeasure development. University of Groningen. The Netherlands.

Van Ginniken, A. M. (2002). The computerized patient record: balancing effort and benefit. ***International Journal of Medical Informatics***, 65, 97–119.

Wieringa, R. (2007). Writing a report about design research. University of Twente. The Netherlands.

Wieringa, R. & Heerkens, H. (2007). Designing requirements engineering research. ***Comparative Evaluation in Requirements Engineering***, 36–48.

www.bizagi.com, accessed on 19 September 19, 2014.

Appendix 1: Database blueprint for a bank transaction

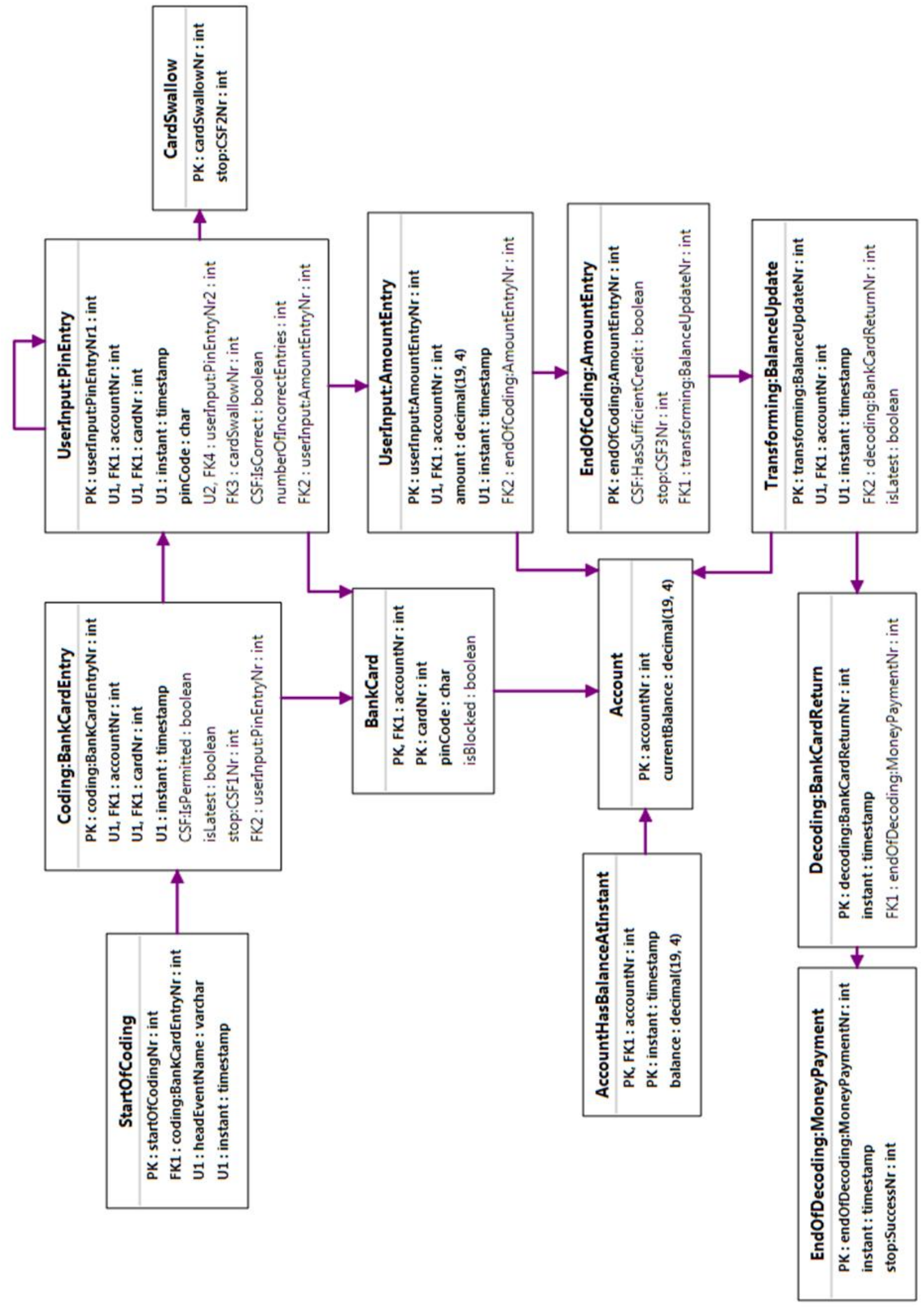

CardSwallow
PK : cardSwallowNr : int
stop:CSF2Nr : int
UserInput:PinEntry
PK : userInput:PinEntryNr1 : int
U1, FK1 : accountNr : int
U1, FK1 : cardNr : int
U1 : instant : timestamp
pinCode : char
U2, FK4 : userInput:PinEntryNr2 : int
FK3 : cardSwallowNr : int
CSF:IsCorrect : boolean
numberOfIncorrectEntries : int
FK2 : userInput:AmountEntryNr : int
UserInput:AmountEntry
PK : userInput:AmountEntryNr : int
U1, FK1 : accountNr : int
amount : decimal(19, 4)
U1 : instant : timestamp
FK2 : endOfCoding:AmountEntryNr : int
EndOfCoding:AmountEntry
PK : endOfCoding:AmountEntryNr : int
CSF:HasSufficientCredit : boolean
stop:CSF3Nr : int
FK1 : transforming:BalanceUpdateNr : int
Transforming:BalanceUpdate
PK : transforming:BalanceUpdateNr : int
U1, FK1 : accountNr : int
U1 : instant : timestamp
FK2 : decoding:BankCardReturnNr : int
isLatest : boolean
Coding:BankCardEntry
PK : coding:BankCardEntryNr : int
U1, FK1 : accountNr : int
U1, FK1 : cardNr : int
U1 : instant : timestamp
CSF:IsPermitted : boolean
isLatest : boolean
stop:CSF1Nr : int
FK2 : userInput:PinEntryNr : int
BankCard
PK, FK1 : accountNr : int
PK : cardNr : int
pinCode : char
isBlocked : boolean
Account
PK : accountNr : int
currentBalance : decimal(19, 4)
Decoding:BankCardReturn
PK : decoding:BankCardReturnNr : int
instant : timestamp
FK1 : endOfDecoding:MoneyPaymentNr : int
StartOfCoding
PK : startOfCodingNr : int
FK1 : coding:BankCardEntryNr : int
U1 : headEventName : varchar
U1 : instant : timestamp
AccountHasBalanceAtInstant
PK, FK1 : accountNr : int
PK : instant : timestamp
balance : decimal(19, 4)
EndOfDecoding:MoneyPayment
PK : endOfDecoding:MoneyPaymentNr : int
instant : timestamp
stop:SuccessNr : int

Figure 1: Database blueprint for a bank transaction (Balsters, 2014).

Appendix 2: The BPMN models

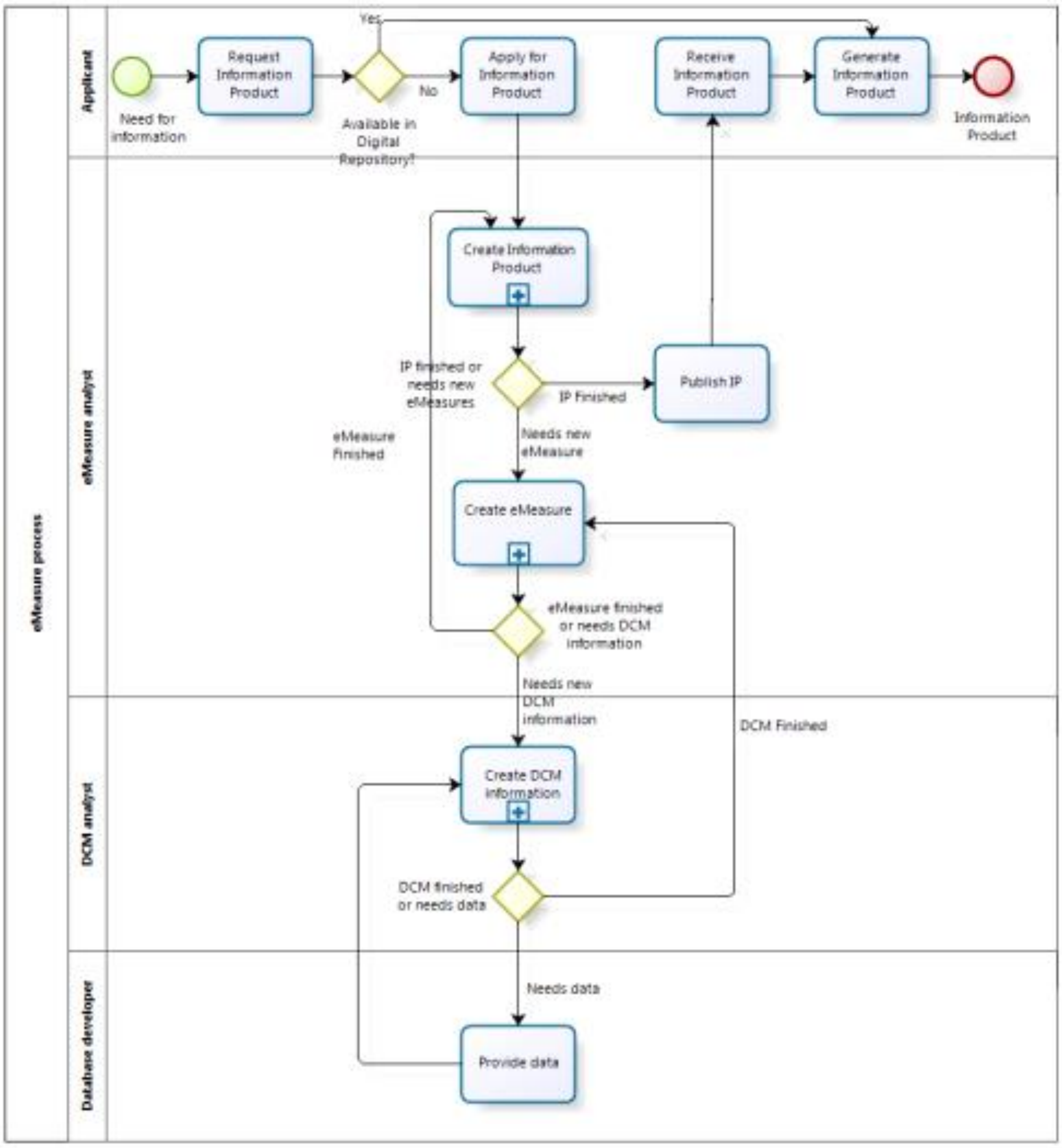
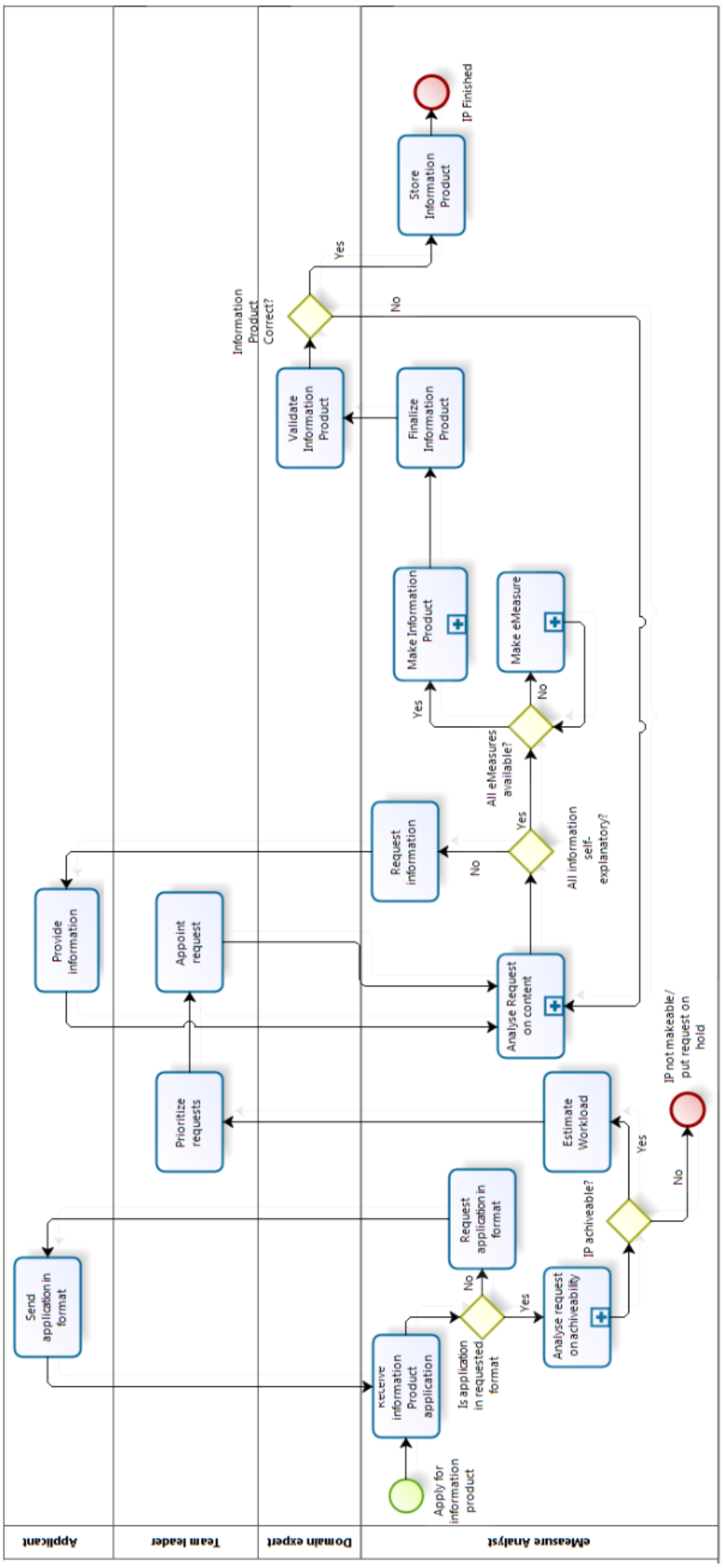

eMeasure process
Applicant
eMeasure analyst
DCM analyst
Database developer
Need for information
Request Information Product
Available in Digital Repository?
Yes
No
Apply for Information Product
Receive Information Product
Generate Information Product
Information Product
Create Information Product
IP finished or needs new eMeasures
IP Finished
Publish IP
Needs new eMeasure
eMeasure Finished
Create eMeasure
eMeasure finished or needs DCM information
Needs new DCM information
DCM Finished
Create DCM information
DCM finished or needs data
Needs data
Provide data

Figure 2: BPMN model of the Universe of Discourse (Van de Laar & Beukeboom, 2015).

BPMN models of Van de Laar (2015)



Applicant
Team leader
Domain expert
eMeasure Analyst
Apply for information product
Receive information Product application
Is application in requested format
No
Yes
Request application in format
Send application in format
Analyse request on achiveability
IP achiveable?
No
IP not makeable/ put request on hold
Yes
Estimate Workload
Prioritize requests
Appoint request
Analyse Request on content
Provide information
All information self-explanatory?
No
Request information
Yes
All eMeasures available?
No
Make eMeasure
Yes
Make Information Product
Finalize Information Product
Validate Information Product
Information Product Correct?
No
Yes
Store Information Product
IP Finished

Figure 3: BPMN model of Create Information Product.

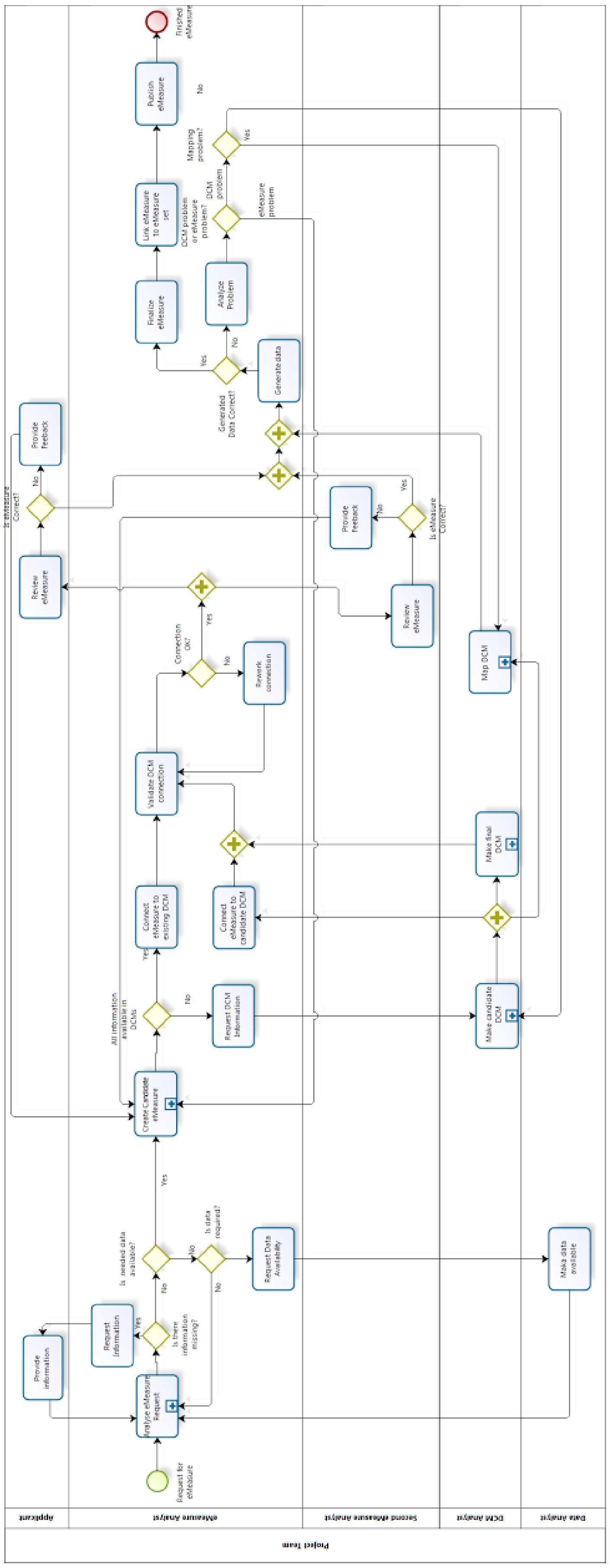


Figure 4: BPMN model of Create eMeasure.

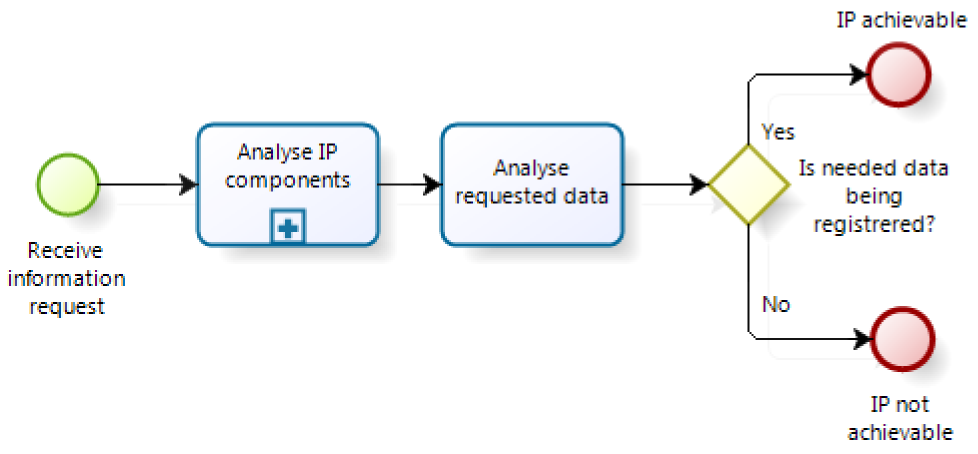


Figure 5: BPMN model of Analyze Request on Achievability.

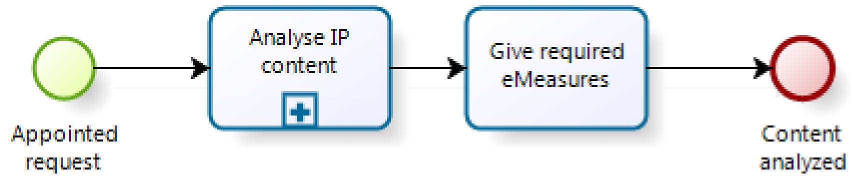


Figure 6: BPMN model of Analyze Request on Content.

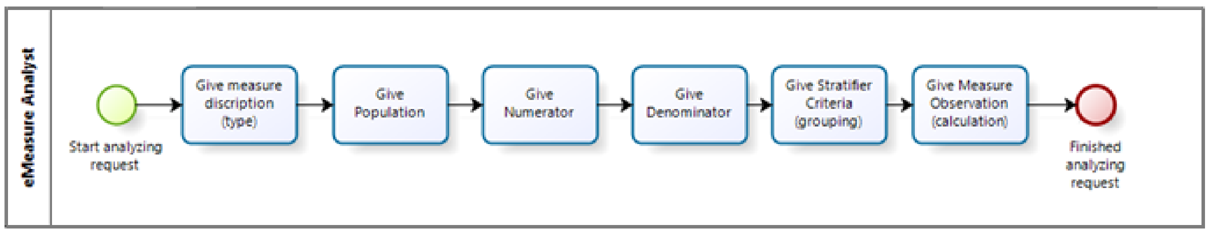

eMeasure Analyst
Start analyzing request
Give measure discription (type)
Give Population
Give Numerator
Give Denominator
Give Stratifier Criteria (grouping)
Give Measure Observation (calculation)
Finished analyzing request

Figure 7: BPMN model of Analyze eMeasure Request.

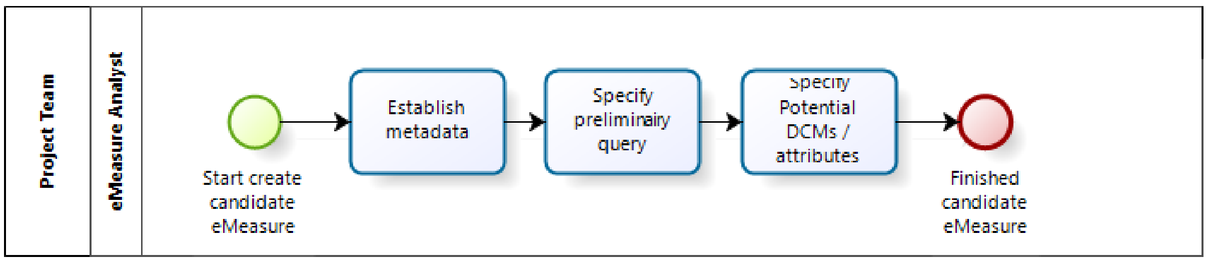

Project Team
eMeasure Analyst
Start create candidate eMeasure
Establish metadata
Specify preliminairy query
Specify Potential DCMs / attributes
Finished candidate eMeasure

Figure 8: BPMN model of Create Candidate eMeasure.

BPMN models of Beukeboom (2015)

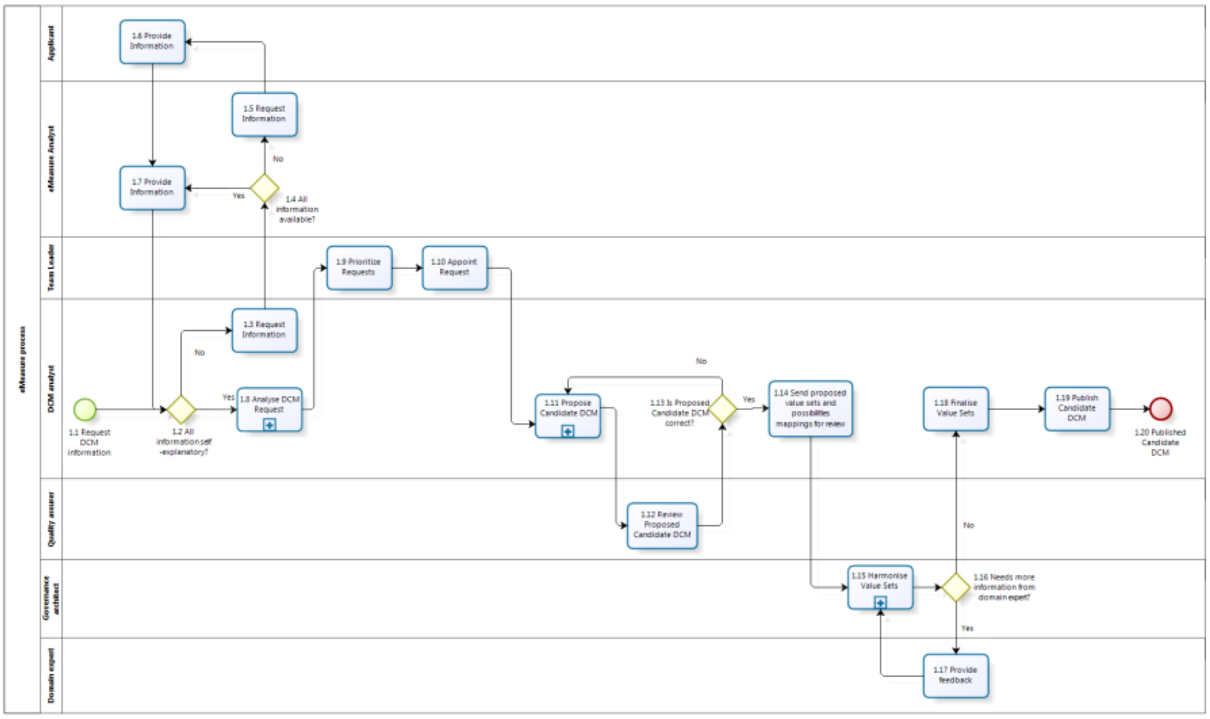

eMeasure process
Applicant
eMeasure Analyst
Team Leader
DCM analyst
Quality assurer
Governance architect
Domain expert
1.6 Provide Information
1.5 Request Information
No
1.7 Provide Information
Yes
1.4 All information available?
1.9 Prioritize Requests
1.10 Appoint Request
1.3 Request Information
No
Yes
1.1 Request DCM information
1.2 All information self-explanatory?
1.8 Analyse DCM Request
No
1.11 Propose Candidate DCM
1.13 Is Proposed Candidate DCM correct?
Yes
1.14 Send proposed value sets and possibilities mappings for review
1.18 Finalise Value Sets
1.19 Publish Candidate DCM
1.20 Published Candidate DCM
1.12 Review Proposed Candidate DCM
No
1.15 Harmonise Value Sets
1.16 Needs more information from domain expert?
Yes
1.17 Provide feedback

Figure 7: BPMN model of Make Candidate DCM.

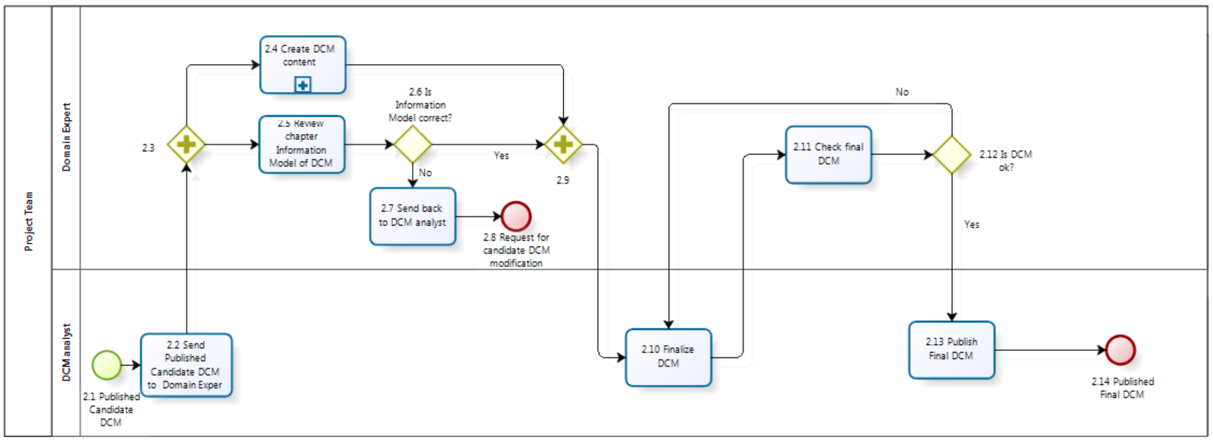

Project Team
Domain Expert
DCM analyst
2.4 Create DCM content
2.3
2.5 Review chapter Information Model of DCM
2.6 Is Information Model correct?
Yes
No
2.9
2.7 Send back to DCM analyst
2.8 Request for candidate DCM modification
No
2.11 Check final DCM
2.12 Is DCM ok?
Yes
2.1 Published Candidate DCM
2.2 Send Published Candidate DCM to Domain Exper
2.10 Finalize DCM
2.13 Publish Final DCM
2.14 Published Final DCM

Figure 8: BPMN model of Make Final DCM.

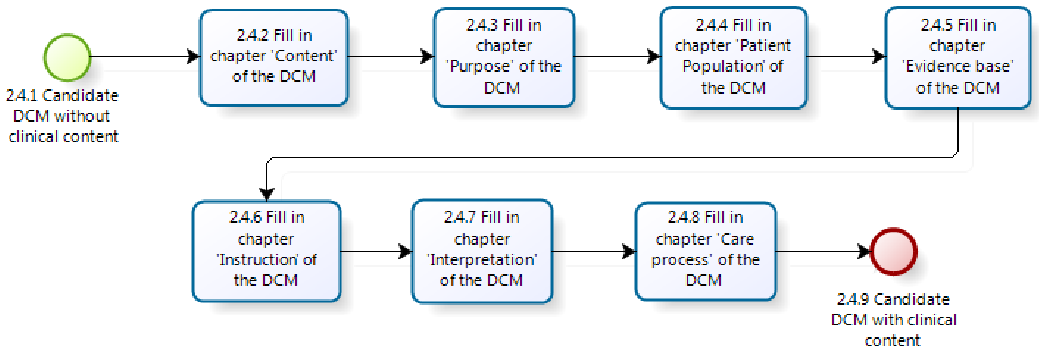

2.4.1 Candidate DCM without clinical content
2.4.2 Fill in chapter 'Content' of the DCM
2.4.3 Fill in chapter 'Purpose' of the DCM
2.4.4 Fill in chapter 'Patient Population' of the DCM
2.4.5 Fill in chapter 'Evidence base' of the DCM
2.4.6 Fill in chapter 'Instruction' of the DCM
2.4.7 Fill in chapter 'Interpretation' of the DCM
2.4.8 Fill in chapter 'Care process' of the DCM
2.4.9 Candidate DCM with clinical content

Figure 9: BPMN model of Create DCM content.

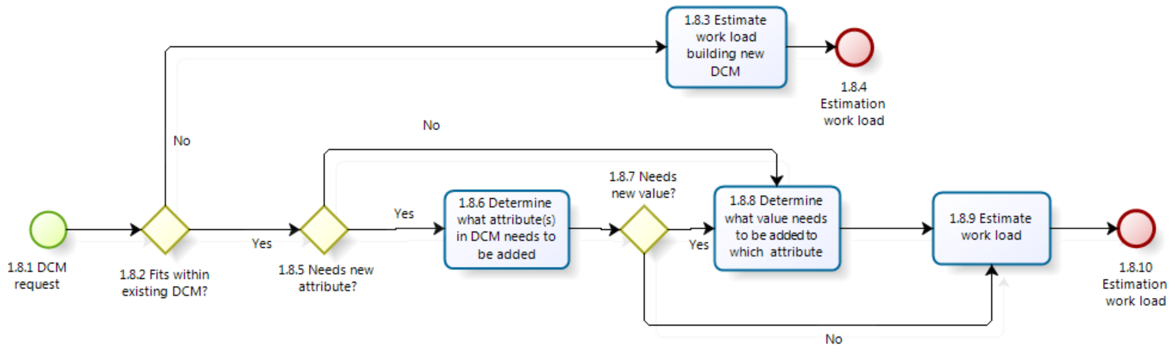

1.8.1 DCM request
1.8.2 Fits within existing DCM?
No
Yes
1.8.3 Estimate work load building new DCM
1.8.4 Estimation work load
1.8.5 Needs new attribute?
No
Yes
1.8.6 Determine what attribute(s) in DCM needs to be added
1.8.7 Needs new value?
Yes
1.8.8 Determine what value needs to be added to which attribute
No
1.8.9 Estimate work load
1.8.10 Estimation work load

Figure 10: BPMN model of Analyze DCM Request.

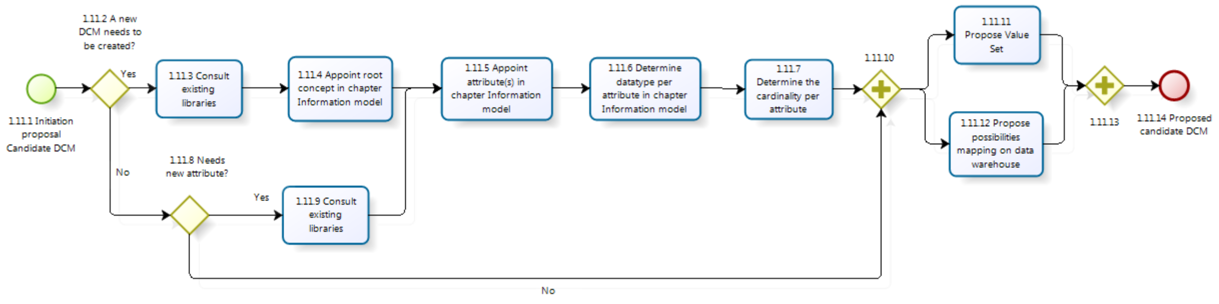

1.11.1 Initiation proposal Candidate DCM
1.11.2 A new DCM needs to be created?
Yes
No
1.11.3 Consult existing libraries
1.11.4 Appoint root concept in chapter Information model
1.11.5 Appoint attribute(s) in chapter Information model
1.11.6 Determine datatype per attribute in chapter Information model
1.11.7 Determine the cardinality per attribute
1.11.8 Needs new attribute?
Yes
1.11.9 Consult existing libraries
No
1.11.10
1.11.11 Propose Value Set
1.11.12 Propose possibilities mapping on data warehouse
1.11.13
1.11.14 Proposed candidate DCM

Figure 11: BPMN model of Propose Candidate DCM.

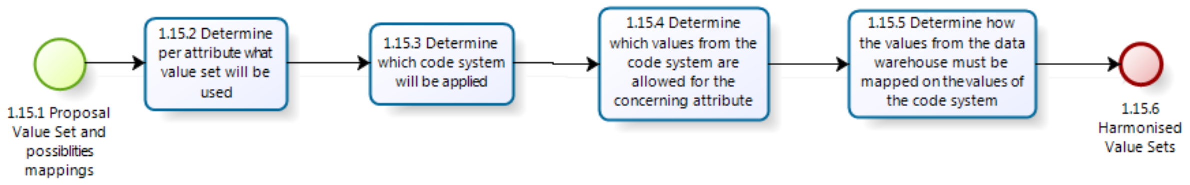

1.15.1 Proposal Value Set and possiblities mappings
1.15.2 Determine per attribute what value set will be used
1.15.3 Determine which code system will be applied
1.15.4 Determine which values from the code system are allowed for the concerning attribute
1.15.5 Determine how the values from the data warehouse must be mapped on the values of the code system
1.15.6 Harmonised Value Sets

Figure 12: BPMN model of Harmonize Value sets.

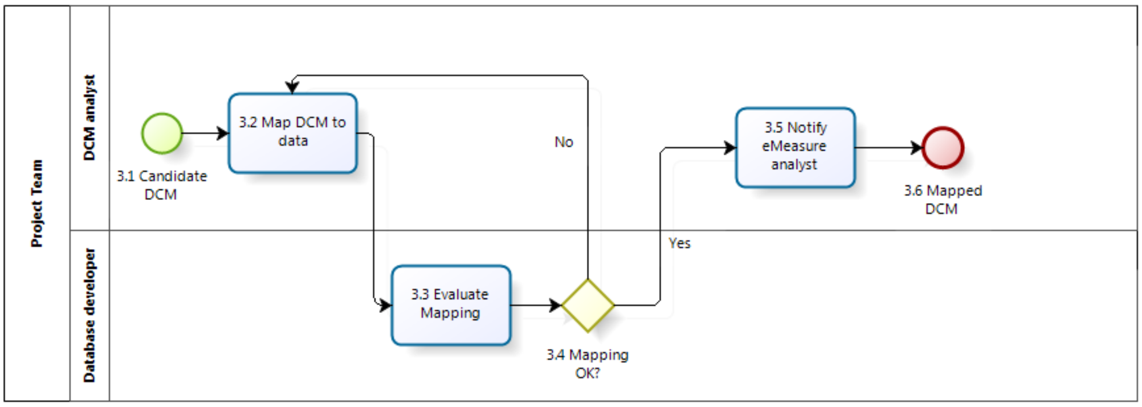


Figure 13: BPMN model of Map DCM.

Appendix 3: The BPMN models explained

Explanation BPMN models of Van de Laar (2015)

In this appendix all activities and gateways from the process models are briefly explained. The numbering of the processes indicates the aggregate level of the process.

Request Information Product
This activity is the starting event of the entire process and starts with an information need from the applicant.

Available in Digital Repository?
This gateway lets the applicant check in a digital repository if there already is an information product that fulfills their information need

Generate Information Product
When the IP is available it can be used to provide the needed information
Input: An information need
Output: The needed information

Apply for Information Product
When the IP is not available an application for an IP is send
Input: An information need where an IP is not available for
Output: An IP application

Receive Information Product
When the IP is finished it is received by the applicant
Input: A published IP
Output: A received IP

Create Information Product
This nested activity creates the IP, the IP is the requested information in the requested format. Examples can be a percentage, a dataset in excel, a dashboard that gives a signal.
Input: An IP application
Output: A finished IP

Receive IP application
The application for an IP is received by the eMeasure analyst
Input: An IP application
Output: A received IP application

Is application in requested format?
This gateway checks if the application is in the requested format, this ensures that the needed basic information is always available on request and that there is a possibility for automation.

Request application in format
When the application is not in format a request is send to the applicant to ensure the application is in format and contains all the preliminary needed information.
Input: an IP application not in format
Output: a request for an IP application in format

Send application in format
A new IP application is send to the eMeasure analyst
Input: a request for an IP application in format
Output: An IP application

Analyze request on achievability
This nested process checks if it is possible to create the IP and what needs to be made
Input: An in format IP application
Output: Information if in IP is makeable

IP achievable?
Is it possible to make the IP within the given criteria

Estimate Workload
Based on the analysis if an IP is makeable it is also known what is already available and what not. Based on this information an estimation of the workload can be made.
Input: Information if in IP is makeable and what needs to be made
Output: a workload estimation

Prioritize request
Based on the urgency for the needed IP and the workload all the requests are prioritized
Input: Information product application and workload estimation
Output: Prioritized IP request

Appoint request
Based on priorities and specialties all the tasks are appointed to the team members
Input: Prioritized IP request
Output: task appointments

Analyze request on content
When the task is appointed a more detailed analysis on the content of the request is done. Questions like what information is requested, how they want to receive this information etc.
Input: task appointment and IP application
Output: Request analysis

All information self-explanatory?
Check if all the needed information is available and if the given information is understood enough for further work.

Request Information
If not all the information is there or understood a request by the applicant is send for more information
Input: Request analysis
Output: Request for additional information

Provide Information
Give additional information needed for IP development
Input: Request for additional information
Output: Additional information

All eMeasures available?
Check if all the eMeasures needed for this information product are available for use

Make Information Product
When all the needed information is there and all the eMeasure are available the information product can be made.
Input: IP request, eMeasures
Output: Information Product

Finalize Information Product
The last information is added to the IP, this is non-functional information like meta-data, production data, creator etc.
Input: Information Product
Output: Finalized Information Product

Validate Information Product
The information product needs to be validated if it meets the requirements and if the information it provides is within the expected range. Most of the time the domain expert and the applicant are the same person
Input: Finalized Information Product
Output: Validated Information Product

Information Product Correct?
Check if the IP met all the validation criteria

Store Information Product
When the IP is finished and validated it is stored in a repository
Input: Validated information product
Output: Stored Information Product

Publish Information Product
A message is send to the applicant and possible others that the IP is available for use
Input: Stored Information Product
Output: Published Information Product

Receive Information Product
The information product can be retrieved from the repository
Input: Notification of publication
Output: Received IP

Generate Information
Input: Received IP
Output: Requested Information

Analyze eMeasure request
Is a nested process where is checked if all the needed information for an eMeasure is available
the steps within this process are directly related to the content of the HQMF standard
Input: Request for eMeasure
Output: Analyzed eMeasure request, eMeasure definitions

Is needed data available?
Based on the definitions of the eMeasure check if all the needed data is available

Is data required?
If data is not available check if the data is required to provide the needed information

Request Data Availability
If data is not available send a request to the data specialist to make the data available
Input: a need for data
Output: a request to make data available

Make data available
Make the data available for usage, mostly for DCM usage
Input: a request to make data available
Output: available data

Create Candidate eMeasure
A nested activity that contains all the preliminary steps for an eMeasure
Input: eMeasure definitions
Output: Candidate eMeasure

All information available in DCMs
Check if all the needed information is available in DCM attributes

Request DCM information
Make a request to the DCM analyst to make the needed data available in a DCM
Input: A need for data from a DCM
Output: A request for adding data to a DCM

Connect eMeasure to existing/candidate DCM
Connect the eMeasure definitions to the needed DCMs
Input: Candidate eMeasure
Output: Candidate eMeasure with connected DCMs

Validate DCM connections
Technical validation to check if all the connections are working and provide the needed data
Input: Candidate eMeasure with connected DCMs
Output: Candidate eMeasure validated on DCM connections

Connection OK?
Check if all the connections are ok

Rework connection
When not all the connections are ok the not correct connections are reworked until they are working
Input: Candidate eMeasure with incorrect DCM connections
Output: Candidate eMeasure with connected DCMs

Review eMeasure
A review on technical and content level
Input: Candidate eMeasure with working DCM connections, request for eMeasure
Output: reviewed candidate eMeasure

Is eMeasure Correct?
Check if during the review no problems were found

Provide feedback
When during the review problems were found feedback is given to the creator of the eMeasure
Input: reviewed candidate eMeasure with problems
Output: Feedback on the problems

Generate data
Generation of data to check if the eMeasure is technically correct
Input: Reviewed candidate eMeasure
Output: Generated data

Generated data correct?
Check if all the needed output is there and if it is within the range of expectations

Analyze Problem
When the generated data is incomplete or not within the range of expectations an analysis is done on what the problem is.
Input: Reviewed candidate eMeasure, generated data
Output: Problem analysis

DCM problem or eMeasure problem?
Is the problem related to the DCM or the eMeasure

Mapping problem?
If it is a problem with the DCM is it the mapping or something else

Finalize eMeasure
Last information is added to the eMeasure, this is non-functional information like meta-data, production data, creator etc.
Input: reviewed eMeasure
Output: Finalized eMeasure

Link eMeasure to eMeasure set
When an eMeasure is part of a bigger set it needs to be linked to these other eMeasures
Input: finalized eMeasure
Output: eMeasure set

Publish eMeasure
The eMeasure is made available for usage
Input: finalized eMeasure
Output: published eMeasure

Explanation BPMN models of Beukeboom (2015)

In this appendix all activities and gateways from the process models are briefly explained. The numbering of the processes indicates the aggregate level of the process. For example, number 1.8.1 refers to: the first number to the highest aggregate level, the second number to one level deeper and the third number to another level deeper.

1. Make Candidate DCM

This nested activity shows that a candidate DCM is made. A candidate DCM is a DCM where the DCM analyst does propositions for and can be used for a preliminary eMeasure connection.

Input: Request for DCM information, Request for a candidate DCM modification.

Output: Candidate DCM

1.1. Request DCM information

This start event indicates the need for DCM information since it is not available for the eMeasure analyst yet.

1.2. All information self-explanatory?

In this gateway the DCM analyst walks through a checklist to see whether all needed information is given and understood. The checklist consist of the following bullet points:

- Did the eMeasure analyst propose a possible DCM?
- Did the eMeasure analyst come up with a proposal for a grouping of attributes or with (a list of) (different) attribute(s)?
- Did the eMeasure analyst come up with the data type per attribute?
- Did the eMeasure analyst come up with requested value sets?
- Is all the needed clinical terminology clear?

Output: if all questions answered positively: Yes, otherwise: No

1.3. Request information

If the DCM analyst did not fully understand the incoming request, he/she makes a request for more information/explanation to the eMeasure analyst. What kind of information depends on the questions which is answered 'No' to in the previous gateway.

1.4. All information available?

In this gateway the eMeasure analyst checks whether he/she has all information available to answer the questions of the DCM analyst regarding the checklist.

Output: Yes/No

1.5. Request information

If in step 1.4 the output was 'No', the eMeasure analyst requests more information to the applicant in case of any unclarities. Since the applicant is most likely not familiar with DCM's, this will be regarding unclear clinical terminology. The eMeasure analyst functions here as the link between the DCM analyst and the applicant.

1.6. Provide information

The applicant provides information about any unclarities what he/she really wants. Especially, very specific clinical terminology is what mostly needs some more attention.

Input: Request for information

Output: Information/Explanation

1.7. Provide information

The eMeasure analyst provides information to the DCM analyst for unclarities regarding the request for DCM information which came up after the DCM's checklist.

Input: Provided information from the applicant, 'Yes' after the check if all information was available

Output: Information/Explanation regarding the request for DCM information.

1.8. Analyze DCM request

In this nested activity the DCM request is analyzed on what needs to be done and an estimation is given of the expected workload.

Input: A clear DCM information request

Output: An expected workload indication

1.8.1. DCM request

This start event indicates a request for DCM information in which is clear to the DCM analyst what needs to be done.

1.8.2. Fits within existing DCM?

This gateway checks whether the requested DCM information fits within an existing DCM.

Output: Yes/No

1.8.3. Estimate workload building new DCM

When in gateway 1.8.2, the output is 'No', an estimation will be made for the workload of building a new DCM. This will in general be most time-consuming compared to the other options regarding DCM's.

1.8.4. Estimation workload

The outcome of the previous step results in the end-event of this sub-process, namely an estimation of the workload of building a new DCM.

1.8.5. Needs new attribute?

This gateway checks whether a new attribute needs to be added within an existing DCM. For explanatory reasons, the hierarchy is given by: a DCM consist of numerous attributes, which on its turn may exist out of numerous values.

Output: Yes/No

1.8.6. Determine what attribute in DCM needs to be added

In case the output of 1.8.5 is 'Yes', it is determined what attribute needs to be added to the DCM.

1.8.7. Needs new value?

This gateway checks whether a new value needs to be added to an attribute. Not all attributes needs values.

1.8.8. Determine what value needs to be added to which attribute

In case the output of 1.8.5 is 'No' and/or the output of gateway 1.8.7 is 'Yes', it is determined what value needs to be added to which attribute of the DCM.

1.8.9. Estimate workload

Subsequently an estimation is made about the workload of adding an attribute and/or adding a new value(s).

1.8.10. Estimation workload

The outcome of the previous step results in the end-event of this sub-process, namely an estimation of the workload.

1.9. Prioritize requests

The team leader prioritizes requests for DCM information based on their estimation of workload and importance.

Input: Estimated workloads of all incoming requests

Output: A prioritized list of DCM requests

1.10. Appoint request

The team leader appoints the request to the DCM analyst based on its prioritized list.

Input: Prioritized list

Output: Appointed DCM requests

1.11. Propose Candidate DCM

This nested activity proposes a candidate DCM both for its structure as for values and their related possible mappings.

1.11.1. Initiation proposal candidate DCM

This start event indicates the initiation of the making of a proposal candidate DCM.

1.11.2. A new DCM needs to be created?

This gateway checks whether a whole new DCM needs to be created or not.

Output: Yes/No

1.11.3. Consult Existing libraries

In the case of a whole new creation of a DCM, existing libraries are consulted for generating ideas. Examples are DCM's from other UMC-projects (PSI, GenOGeg, eOverdracht) or clinical models at CEM, CDE and OpenEHR.

1.11.4. Appoint root concept

In this activity the root concept is determined for the new to-be-developed DCM in the chapter Information Model. An example is 'Drugs Usage'.

1.11.5. Appoint attribute(s)

In this activity the attribute(s) is/are determined for the concerning DCM in the chapter Information Model. An example within 'Drugs Usage' is 'Sort of Drugs'.

1.11.6. Determine data type per attribute

In this activity the data type per attribute is determined for the concerning DCM in the chapter Information Model. The data type for 'Sort of Drugs' is CD.

1.11.7. Determine cardinality

In this activity the cardinality per attribute is determined in the chapter Information Model of the DCM. The cardinality of 'Sort of Drugs' is 0…*.

1.11.8. Needs new attribute?

This gateway checks whether a whole new attribute needs to be added to an existing DCM or only a new value to an existing attribute

Output: Yes/No

1.11.9. Consult existing libraries

In the case of adding a new attribute within an existing DCM, existing libraries are consulted for generating ideas. Examples are DCM's from other UMC-projects (PSI, GenOGeg, eOverdracht) or clinical models at CEM, CDE and OpenEHR

1.11.10. Parallel gateway

This gateway shows that from this point multiple activities take place simultaneously, i.e. in parallel.

1.11.11. Propose value set

In this activity a proposition is made by the DCM analyst for the value sets that are options to choose from for doctors. A few examples of values within the attribute 'Sort of Drugs' are Heroin (substance), Benzodiazepine (substance) and Methadone (substance).

1.11.12. Propose possibilities mapping on data warehouse

Together with the propositions for value sets, possible mappings for these values on the data warehouse are proposed. This is done this way, so no value sets are proposed that are not available in the data warehouse.

1.11.13. Parallel gateway

This parallel gateway shows that first the previously two activities needs to be completed before the process can be continued

1.11.14. Proposed candidate DCM

This end event indicates the end of the sub-process Propose candidate DCM with its output a proposed candidate DCM.

1.12. Review proposed Candidate DCM

The proposed candidate DCM needs to be reviewed by the architect, who is specialized in designing and defining DCM's according the official standards set. The architect focuses on the structure of the DCM (e.g. root concept, attributes, data type and cardinality) and not on the clinical content or mapping.

1.13. Is proposed Candidate DCM correct?

This gateway checks whether the Candidate DCM's structure is correct according to the architect. Either the candidate DCM needs to be adjusted based on the feedback of the architect or the process can be continued.

Output: Yes/No

1.14. Send Proposed value sets and possibilities mappings for review

When the DCM's structure is approved, the proposed value sets and possibilities for mapping are send towards the governance architect to harmonize these and he makes the final call.

1.15. Harmonize Value sets

This nested activity determines which value sets, which codes and mappings are to be used in the DCM. This is done by the governance architect.

1.15.1. Proposal Value Set and possibilities mappings

This is the start event where the governance architect receives the proposed value sets and possibilities for mappings.

1.15.2. Determine per attribute what value set will be used

This activity determines per attribute what value set will be eventually used, regarding the possibilities of mappings.

1.15.3. Determine which code system will be applied

This activity determines which code system will be applied for the value sets for each attribute.

1.15.4. Determine which values from the code system are allowed for the concerning attribute

This activity determines which values from the code system are allowed for the concerning attribute.

1.15.5. Determine which values from the data warehouse must be mapped on the values of the code system

This activity determines which values from the data warehouse must be mapped on the values of the code system. In case of multiple candidate fields in the data warehouse, it is indicated how the different fields map on the attributes of the candidate DCM.

1.15.6. Harmonized Value Sets

This start event indicates the eventual harmonized Value Sets.

1.16. Needs more information from domain expert?

This gateway checks whether more information is needed from a domain expert to harmonize the value sets.

Output: Yes/No

1.17. Provide feedback

In case the governance architect is not able to determine certain aspects within the harmonization process of the value sets and therefore needs more information, a domain expert is consulted, who provides feedback.

1.18. Finalize Value Sets

In case the Value Sets are harmonized in a satisfied manner by the governance architect, the DCM analyst can finalize the value sets within the DCM. Mainly, the code system has to be adopted and a finishing touch is given to the chapter Information Model.

1.19. Publish Candidate DCM

This activity publishes the Candidate DCM so the eMeasure analyst can use this for further practices.

1.20. Published Candidate DCM

The end-event indicates the output of the sub-process 'Make candidate DCM', namely a published candidate DCM.

2. Make Final DCM

This nested activity shows that a final DCM is made. Final means that the DCM is reviewed by a domain expert and all clinical information is added and validated. A final DCM is not necessarily also mapped.

Input: Candidate DCM

Output: Final DCM

2.1. Published Candidate DCM

This start event indicates that the process of making a final DCM starts with a published candidate DCM.

2.2. Send published Candidate DCM to Domain Expert

The DCM analyst sends the published Candidate DCM to a domain expert related to that expertise for review and let him/her add clinical content.

2.3. Parallel gateway

This gateway shows that from this point two activities are performed simultaneously.

2.4. Create DCM content

One of these parallel activities is the nested activity of creating DCM content with regard to the clinical aspects.

2.4.1 Candidate DCM without clinical content

This start-event indicates that the process of creating DCM content starts with a candidate DCM without clinical content.

2.4.2 Fill in chapter 'Content' of the DCM

This activity shows that the chapter 'Content' of the DCM is filled in by the domain expert.

2.4.3 Fill in chapter 'Purpose' of the DCM

This activity shows that the chapter 'Purpose' of the DCM is filled in by the domain expert.

2.4.4 Fill in chapter 'Patient Population' of the DCM

This activity shows that chapter 'Patient Population' of the DCM is filled in by domain expert.

2.4.5 Fill in chapter 'Evidence base' of the DCM

This activity shows that the chapter 'Evidence base' of the DCM is filled in by the domain expert.

2.4.6 Fill in chapter 'Instruction' of the DCM

This activity shows that the chapter 'Instruction' of the DCM is filled in by the domain expert.

2.4.7 Fill in chapter 'Interpretation' of the DCM

This activity shows that the chapter 'Interpretation' of the DCM is filled in by the domain expert.

2.4.8 Fill in chapter 'Care Process' of the DCM

This activity shows that the chapter 'Care Process' of the DCM is filled in by the domain expert.

2.4.9 Candidate DCM with clinical content

This end-event indicates that the activity 'create DCM content' is ended with as output a candidate DCM with clinical content.

2.5. Review chapter Information Model of DCM

The other parallel activity that is performed by the domain expert is reviewing the chapter Information Model of the DCM.

2.6. Is Information Model correct?

This gateway checks what the outcome is of the review of the domain expert regarding the Information Model. When the domain expert still see some flaws, the candidate DCM needs to be modified.

Outcome: Yes/No

2.7. Send back to DCM analyst

In this activity the domain expert sends the Information Model back to the DCM analyst with his comments for modifications to the Candidate DCM.

2.8. Request for Candidate DCM modification

This end-event shows the output of where the process ends when the Information model is not fully correct, namely a request to the DCM analyst for a modification towards the Candidate DCM.

2.9. Parallel gateway

This gateway shows that the previously done activities have to be completed first before the process can be continued.

2.10. Finalize DCM

In this activity the DCM is finalized by putting all information in the correct chapters of the DCM and metadata (e.g. author, version update etc.) is added.

2.11. Check final DCM

The domain expert does a final quality check on the finalized DCM.

2.12. Is DCM ok?

This gateway checks whether the final DCM is approved by the domain expert. If not, the DCM is sent back to the DCM analyst to modify the DCM.

Output: Yes/No.

2.13. Publish final DCM

If the final DCM is fully approved, it will be published, so that the eMeasure analyst can make the final eMeasure-DCM connection.

2.14. Published Final DCM

This end-event shows the output of the sub-process 'Make final DCM', namely a published final DCM.

3. Map DCM

This nested activity shows that a candidate DCM is mapped onto the existing data warehouse, so actual data can be generated from the DCM.

Input: Candidate DCM

Output: Mapped DCM

3.1. Candidate DCM

This start-event indicates that the process of 'Map DCM' starts with a candidate DCM.

3.2. Map DCM to data

In this activity the eventual determined value sets with their related mappings from the DCM are actually mapped on the fields of the existing data warehouse by the DCM analyst.

3.3. Evaluate Mapping

To verify the mapping of the previous activity, an evaluation is done by the database developer. This is the person who is responsible for the data at the source concerning the mapped fields, i.e. the existing data warehouse. This is a technical evaluation where is tested whether the mapping results in the correct data generation.

3.4. Mapping OK?

This gateway checks whether the mapping is accepted by the database developer or not. In case this is not the case, the mapping has to be modified according to the feedback of the database developer.

Output: Yes/No

3.5. Notify eMeasure analyst

If the mapping is approved by the database developer, the DCM analyst notifies the eMeasure analyst that the mapping is completed and he/she can generate the data for the eMeasures.

3.6. Mapped DCM

This end-event indicates that the output of the activity 'Map DCM' is a mapped DCM.

4. Parallel gateway

This parallel gateway shows that when a candidate DCM is published, three activities are done in parallel, namely 'connect eMeasure to candidate DCM' by the eMeasure analyst and 'Make final DCM' and 'Map DCM' by the DCM analyst.

5. Request for candidate DCM modification?

This gateway checks which of the two end-events in the nested activity 'Make final DCM' is chosen. This is done by formulating the question whether there is a request for a candidate DCM modification?

Output: Yes/No

Appendix 4: The ORM diagrams

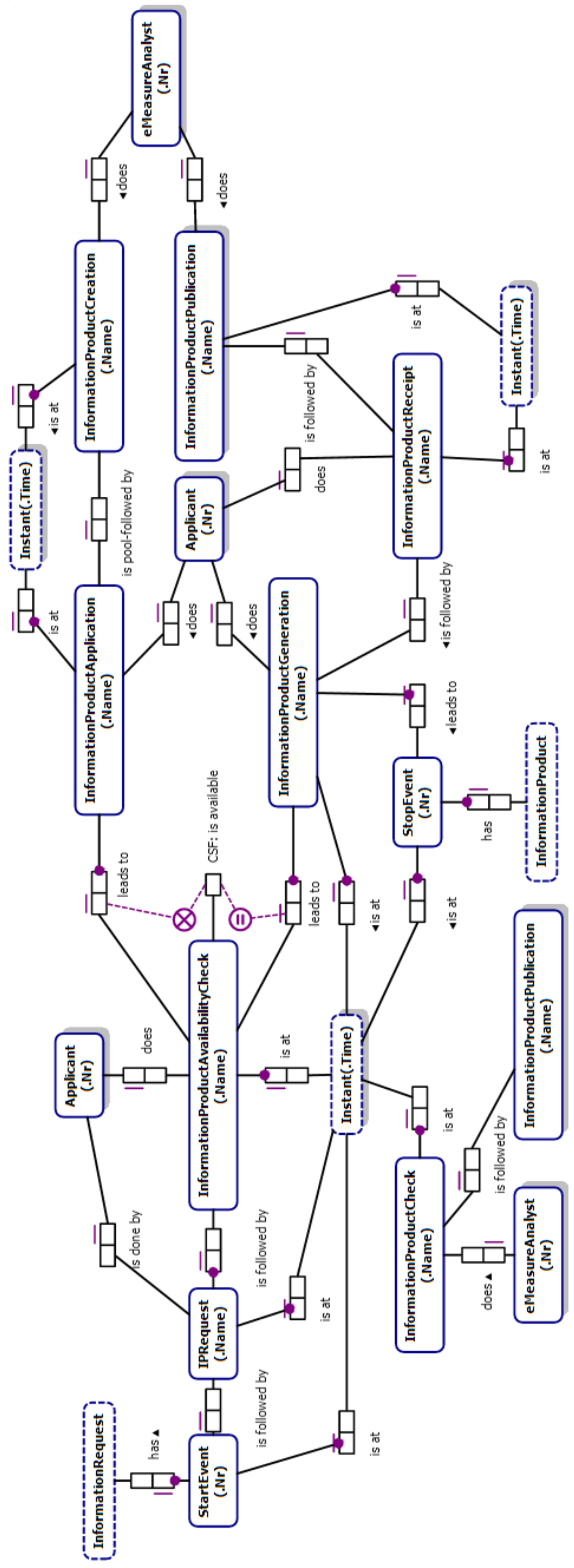
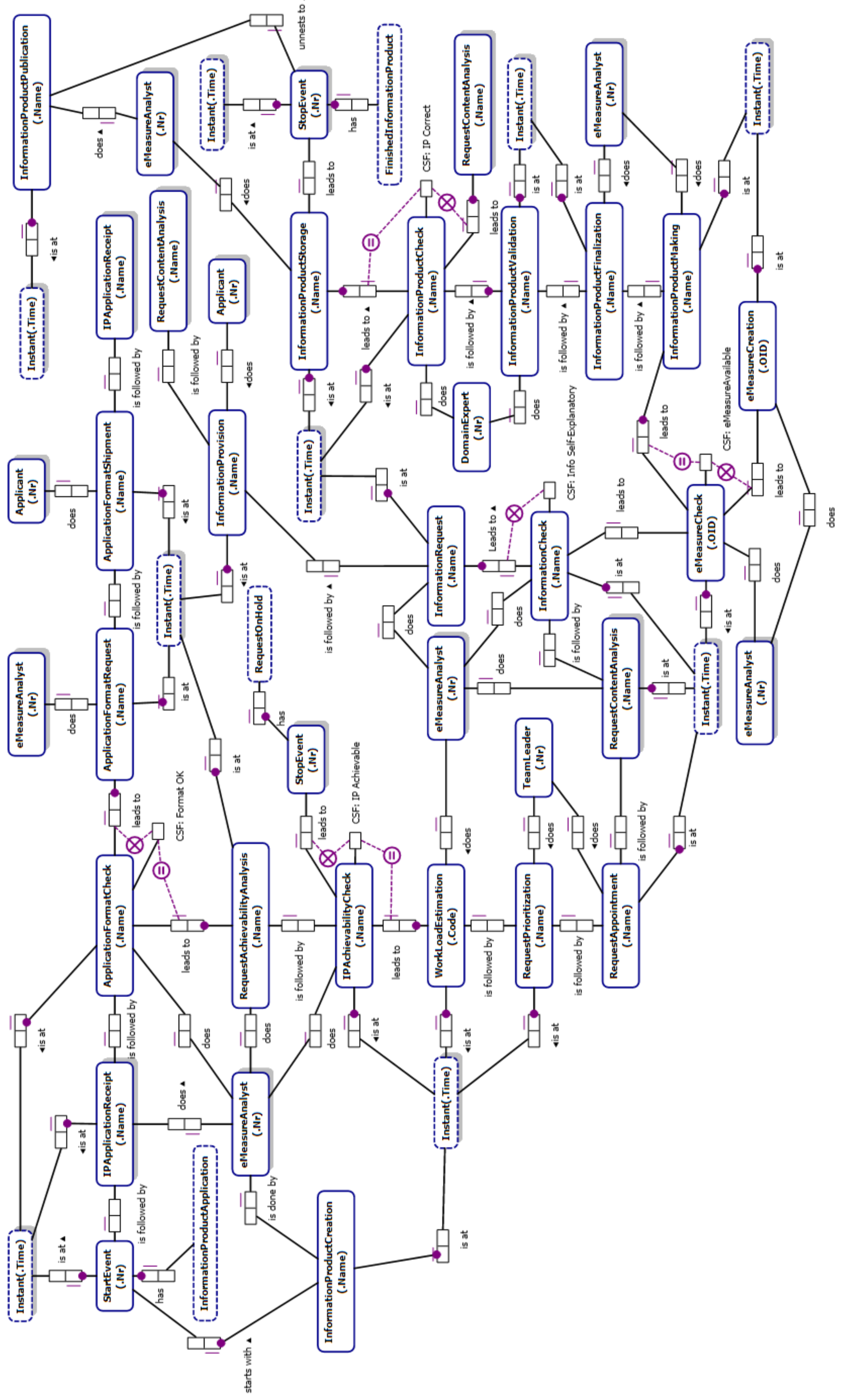


Figure 14: ORM diagram of the first part of the Universe of Discourse.



InformationProductCreation (.Name)
StartEvent (.Nr)
Instant(.Time)
InformationProductApplication
eMeasureAnalyst (.Nr)
IPApplicationReceipt (.Name)
ApplicationFormatCheck (.Name)
ApplicationFormatRequest (.Name)
ApplicationFormatShipment (.Name)
Applicant (.Nr)
RequestAchievabilityAnalysis (.Name)
IPAchievabilityCheck (.Name)
WorkLoadEstimation (.Code)
RequestPrioritization (.Name)
RequestAppointment (.Name)
TeamLeader (.Nr)
StopEvent (.Nr)
RequestOnHold
InformationProvision (.Name)
RequestContentAnalysis (.Name)
InformationRequest (.Name)
InformationCheck (.Name)
eMeasureCheck (.OID)
eMeasureCreation (.OID)
InformationProductStorage (.Name)
InformationProductCheck (.Name)
InformationProductValidation (.Name)
InformationProductFinalization (.Name)
InformationProductMaking (.Name)
DomainExpert (.Nr)
InformationProductPublication (.Name)
FinishedInformationProduct
CSF: Format OK
CSF: IP Achievable
CSF: IP Correct
CSF: Info Self-Explanatory
CSF: eMeasureAvailable
starts with
is done by
is followed by
leads to
is at
does
has
unnests to

Figure 15: ORM diagram of Create Information Product.

The ‘Create eMeasure’ BPMN model is divided into three models and three database blueprints due to the relatively high level of complexity of this model.

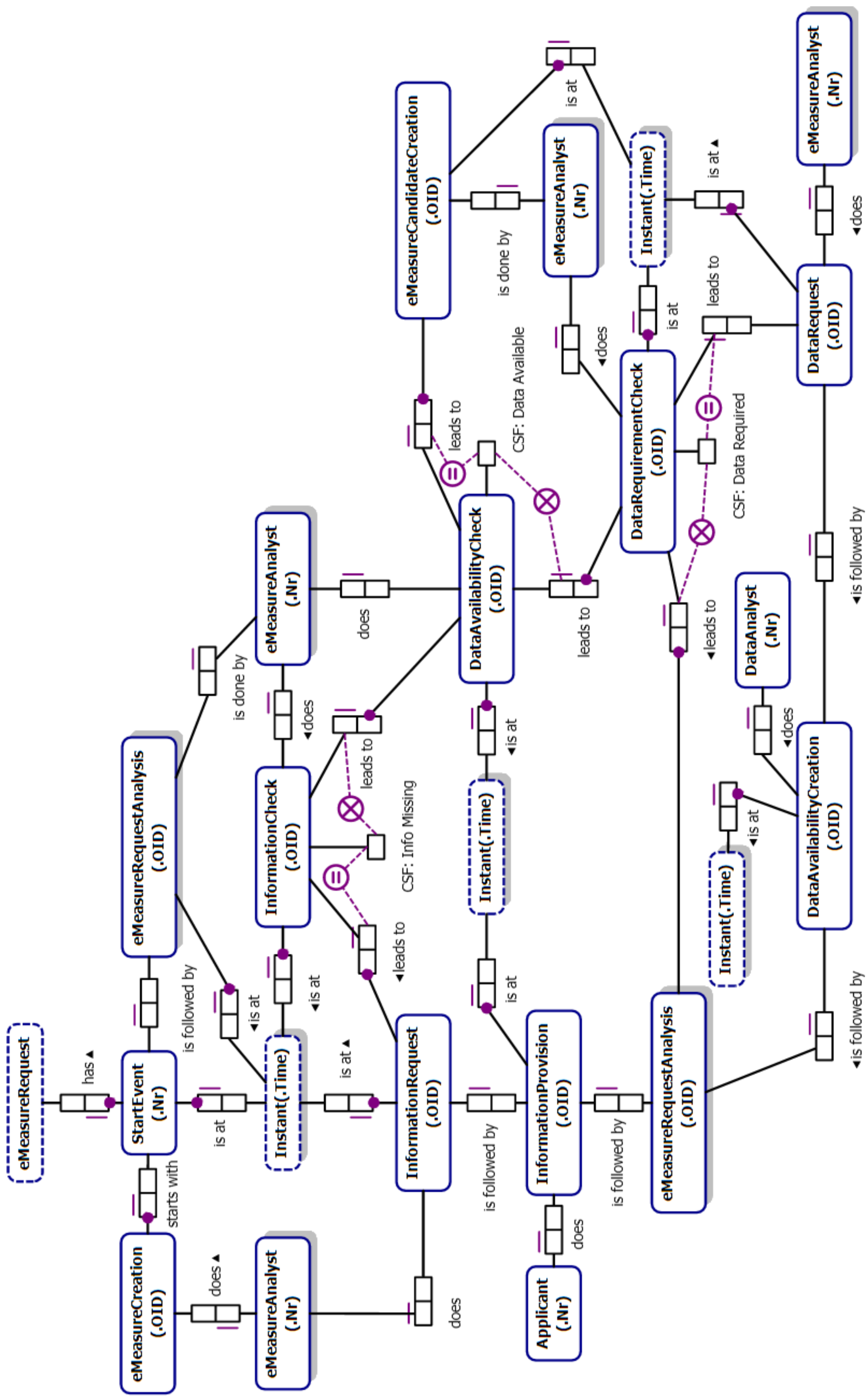


Figure 16: ORM diagram of Create eMeasure (first part).

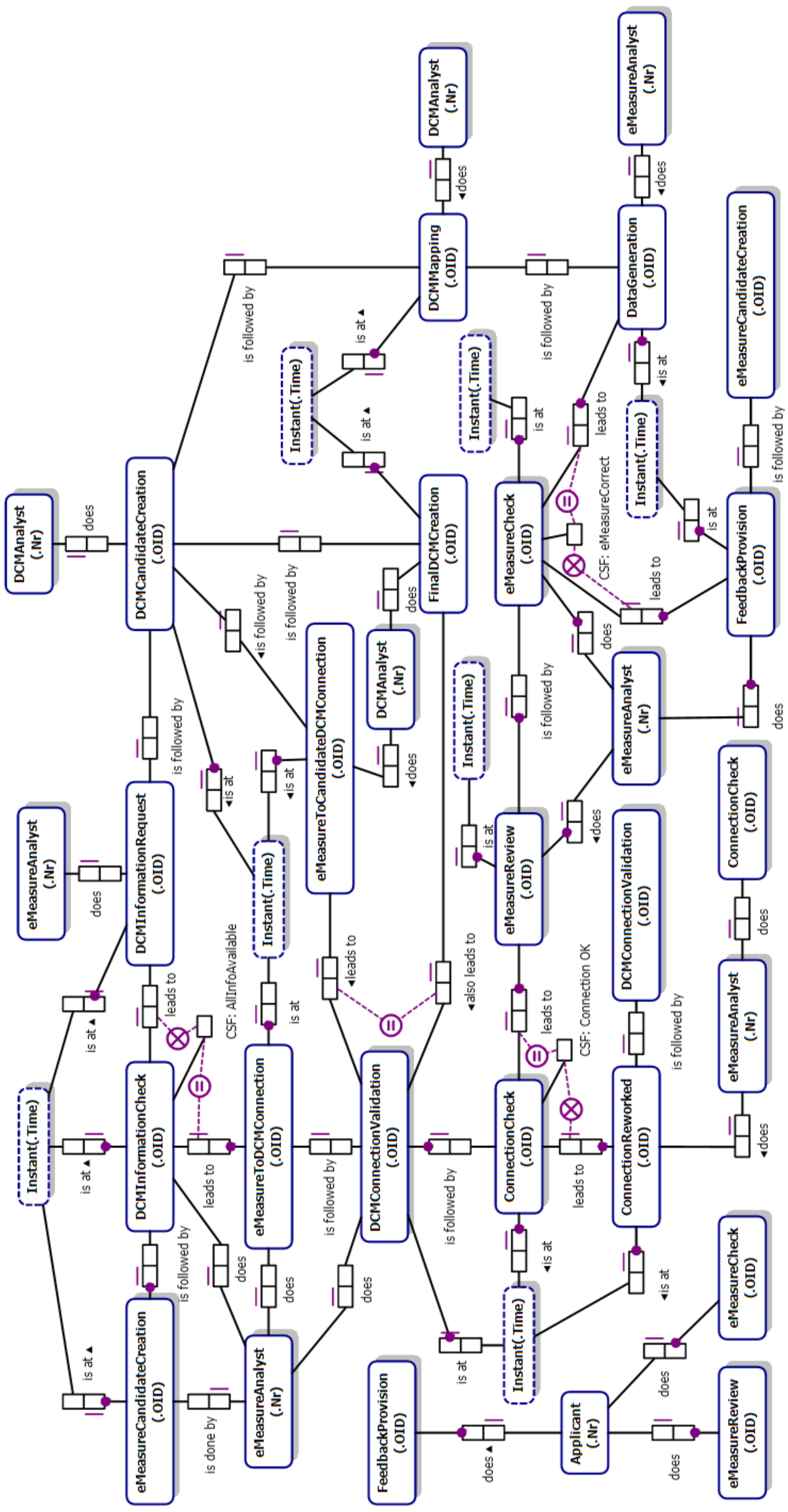


Figure 17: ORM diagram of Create eMeasure (second part).

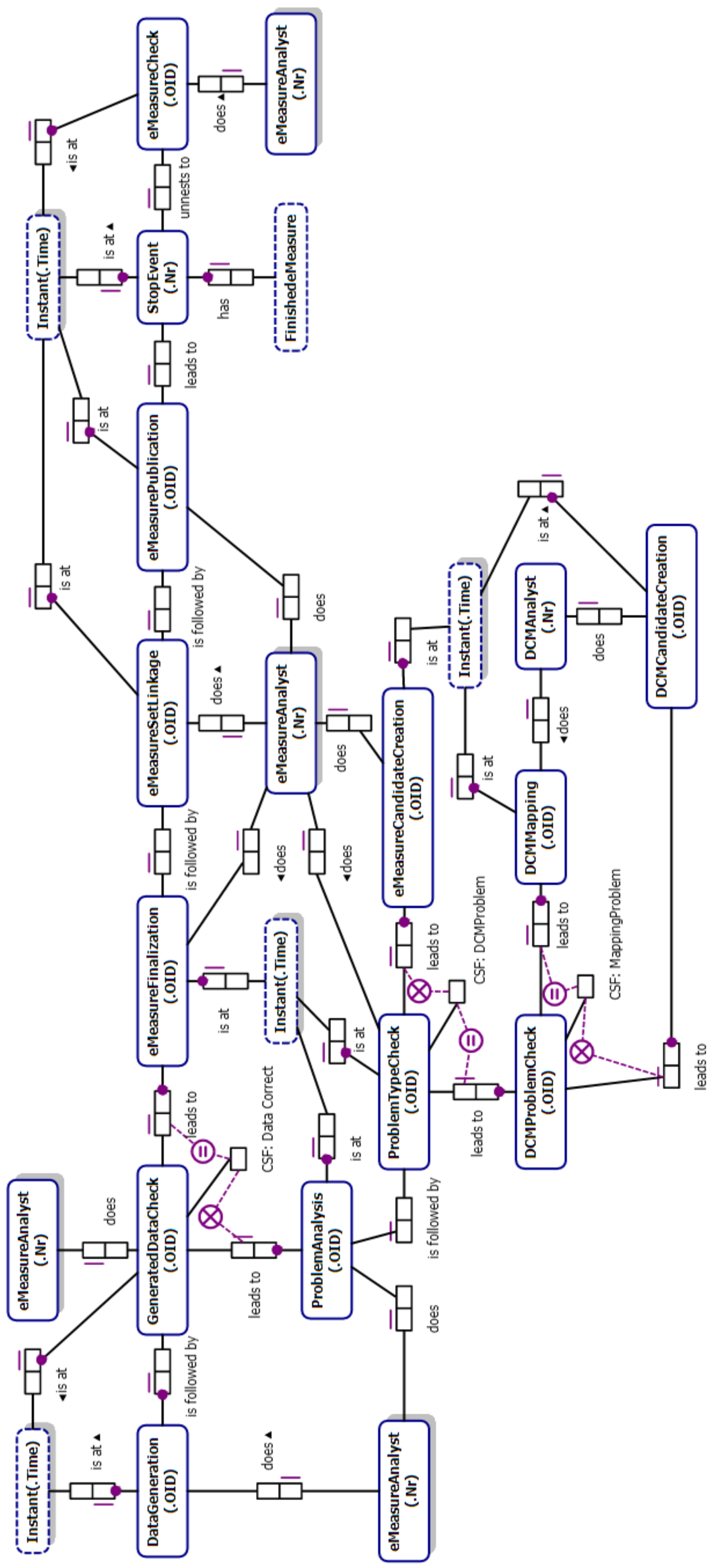


Figure 18: ORM diagram of Create eMeasure (third part).

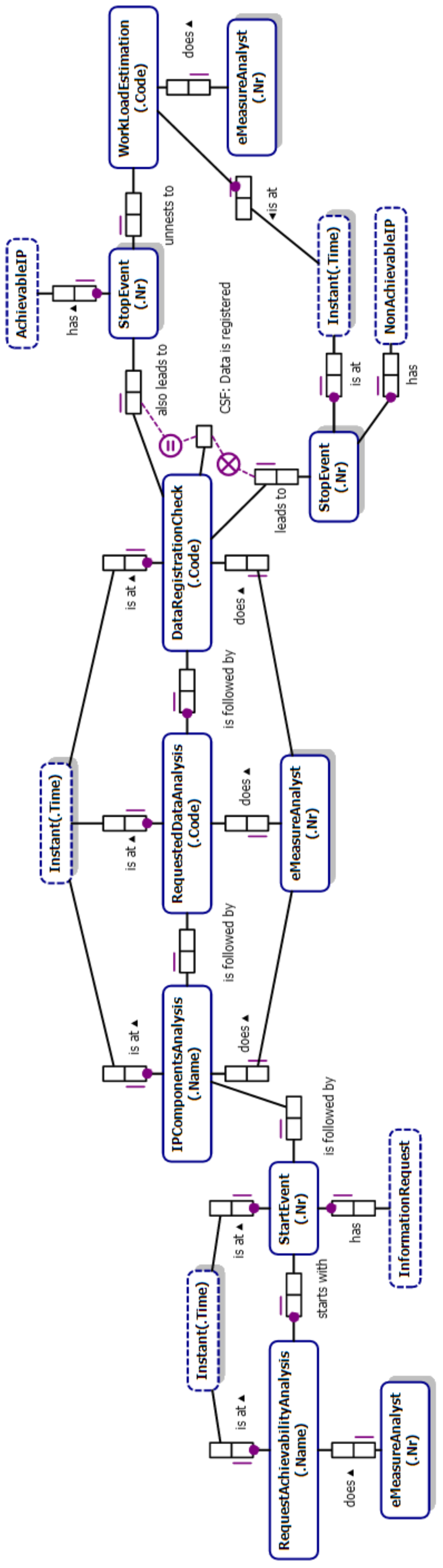


Figure 19: ORM diagram of Analyze Request on Achievability.

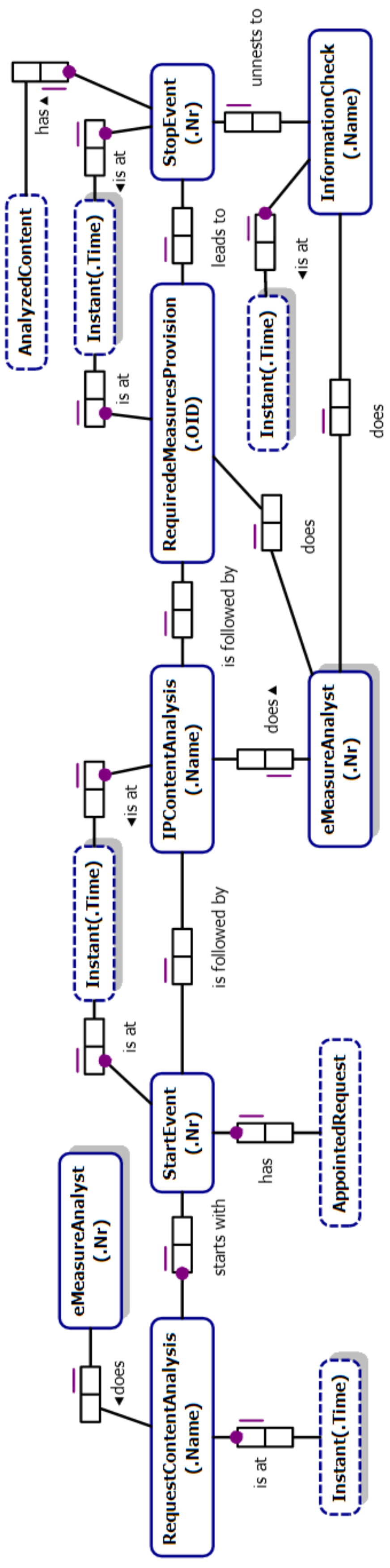


Figure 20: ORM diagram of Analyze Request on Content.

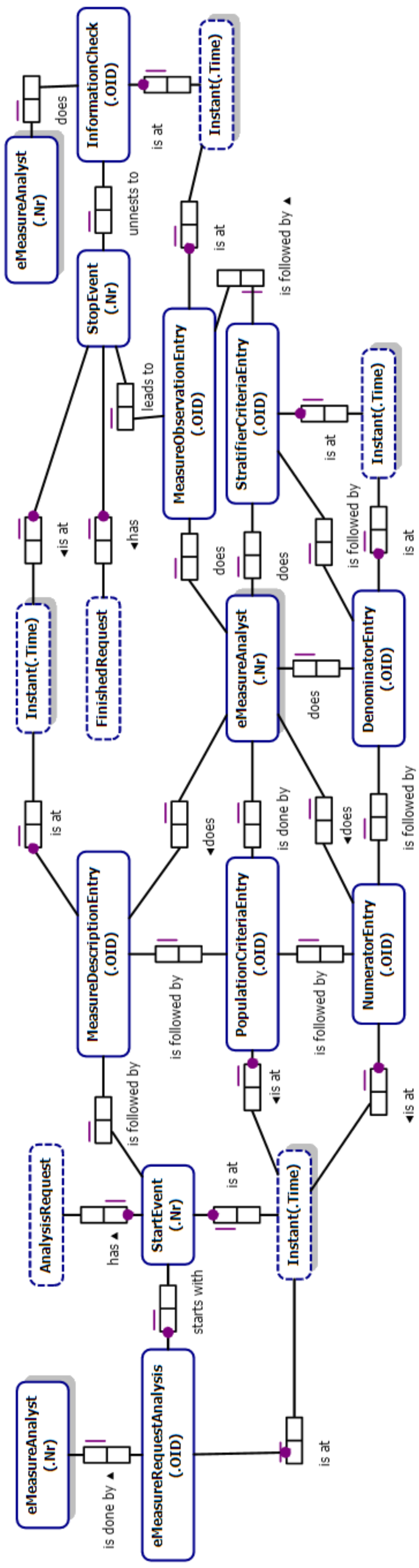


Figure 21: ORM diagram of Analyze eMeasure Request.

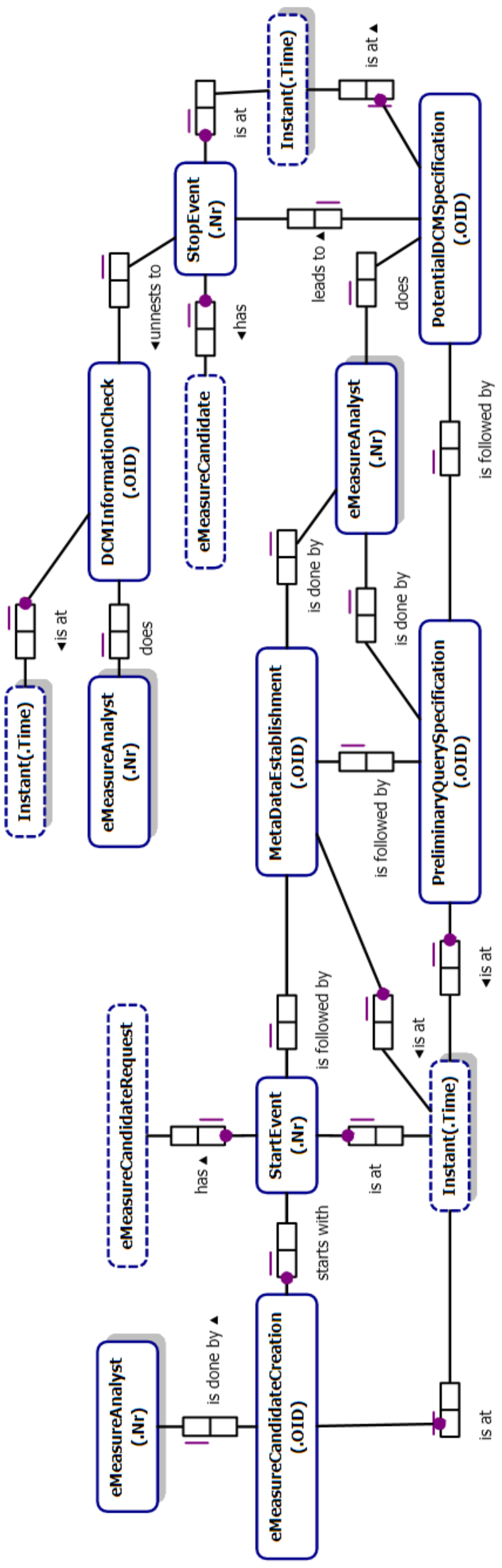
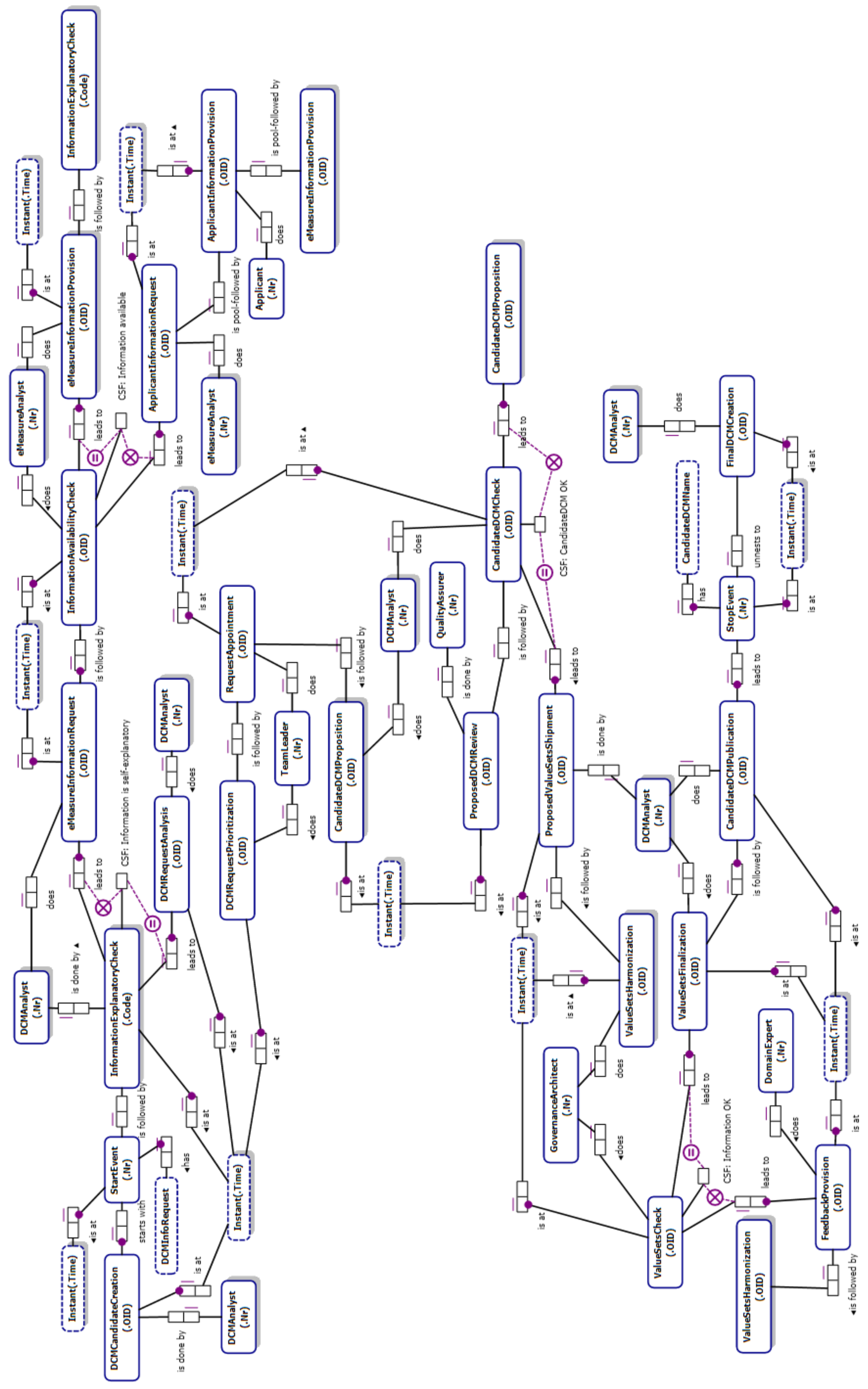


Figure 22: ORM diagram of Create Candidate eMeasure.


InformationExplanatoryCheck (.Code)
Instant(.Time)
eMeasureInformationProvision (.OID)
ApplicantInformationProvision (.OID)
eMeasureInformationProvision (.OID)
Applicant (.Nr)
ApplicantInformationRequest (.OID)
eMeasureAnalyst (.Nr)
CSF: Information available
InformationAvailabilityCheck (.OID)
eMeasureInformationRequest (.OID)
CSF: Information is self-explanatory
DCMAnalyst (.Nr)
DCMRequestAnalysis (.OID)
DCMRequestPrioritization (.OID)
RequestAppointment (.OID)
TeamLeader (.Nr)
CandidateDCMProposition (.OID)
CandidateDCMCheck (.OID)
CandidateDCMProposition (.OID)
QualityAssurer (.Nr)
ProposedDCMReview (.OID)
CSF: CandidateDCM OK
ProposedValueSetsShipment (.OID)
ValueSetsHarmonization (.OID)
ValueSetsFinalization (.OID)
GovernanceArchitect (.Nr)
ValueSetsCheck (.OID)
CSF: Information OK
DomainExpert (.Nr)
FeedbackProvision (.OID)
ValueSetsHarmonization (.OID)
CandidateDCMPublication (.OID)
StopEvent (.Nr)
CandidateDCMName
FinalDCMCreation (.OID)
StartEvent (.Nr)
DCMInfoRequest
DCMCandidateCreation (.OID)
does
is at
leads to
is followed by
is pool-followed by
is done by
starts with
has
unrests to
Figure 23: ORM diagram of Make candidate DCM.

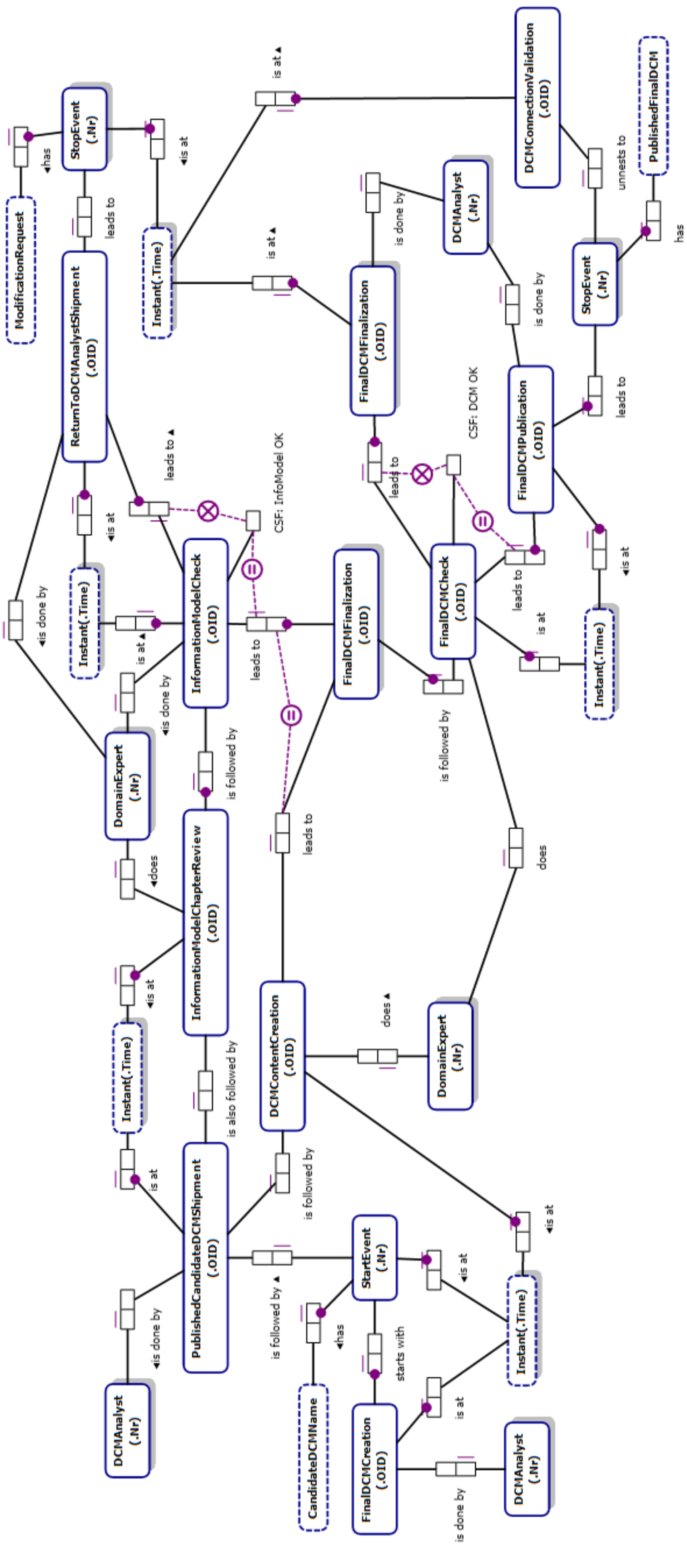
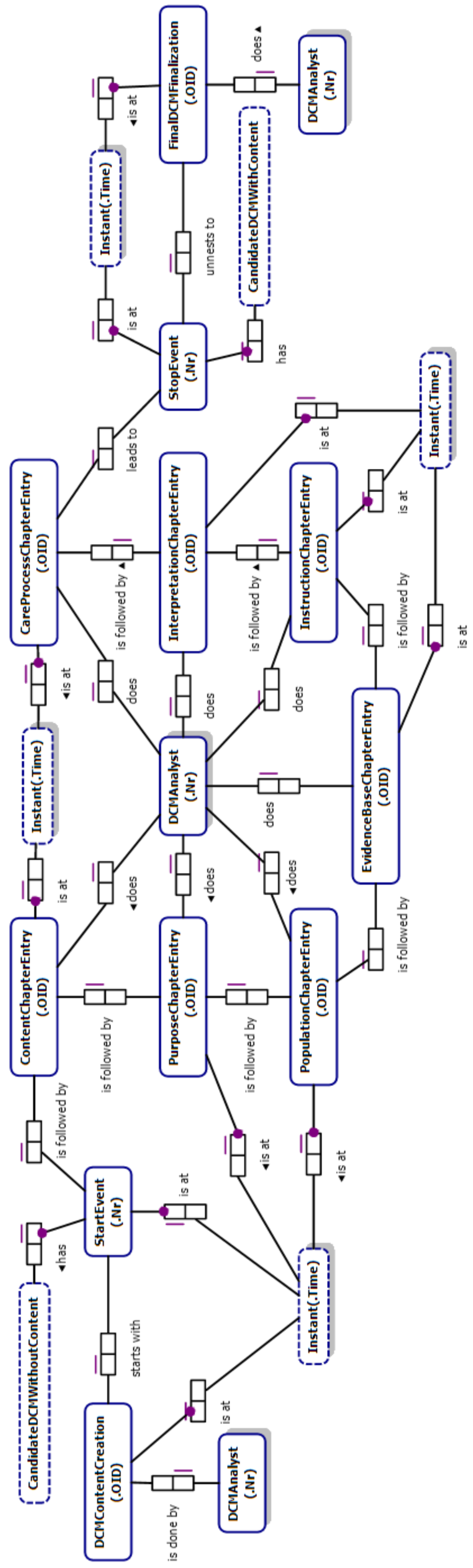


Figure 24: ORM diagram of Make final DCM.



FinalDCMFinalization (.OID)
DCMAnalyst (.Nr)
Instant(.Time)
CandidateDCMWithContent
StopEvent (.Nr)
CareProcessChapterEntry (.OID)
InterpretationChapterEntry (.OID)
InstructionChapterEntry (.OID)
EvidenceBaseChapterEntry (.OID)
ContentChapterEntry (.OID)
PurposeChapterEntry (.OID)
PopulationChapterEntry (.OID)
StartEvent (.Nr)
CandidateDCMWithoutContent
DCMContentCreation (.OID)
is at
does
unnests to
has
leads to
is followed by
starts with
is done by

Figure 25: ORM diagram of Create DCM Content.

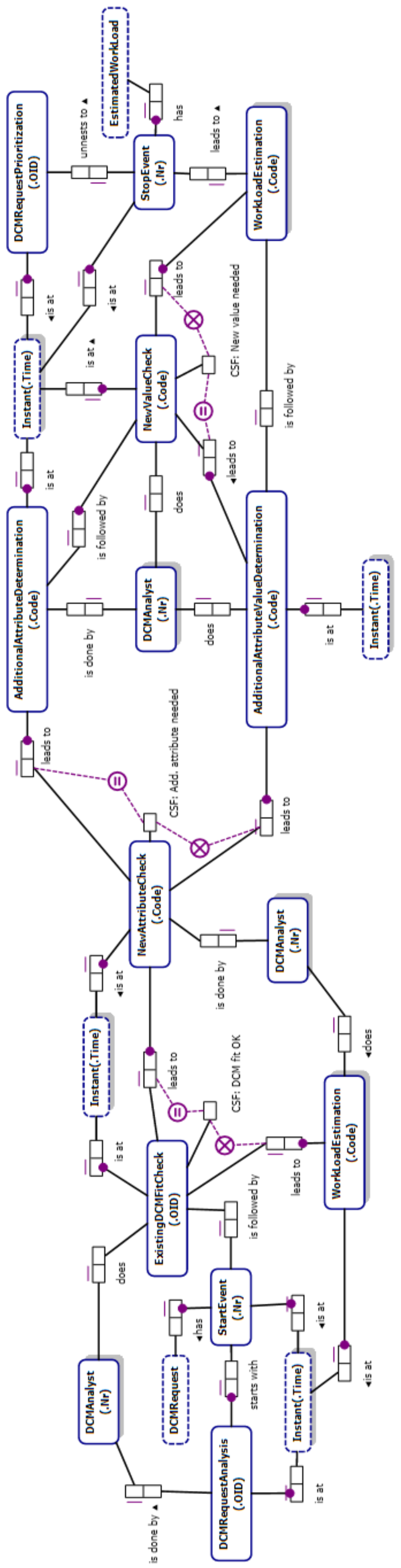


Figure 26: ORM diagram of Analyze DCM request.

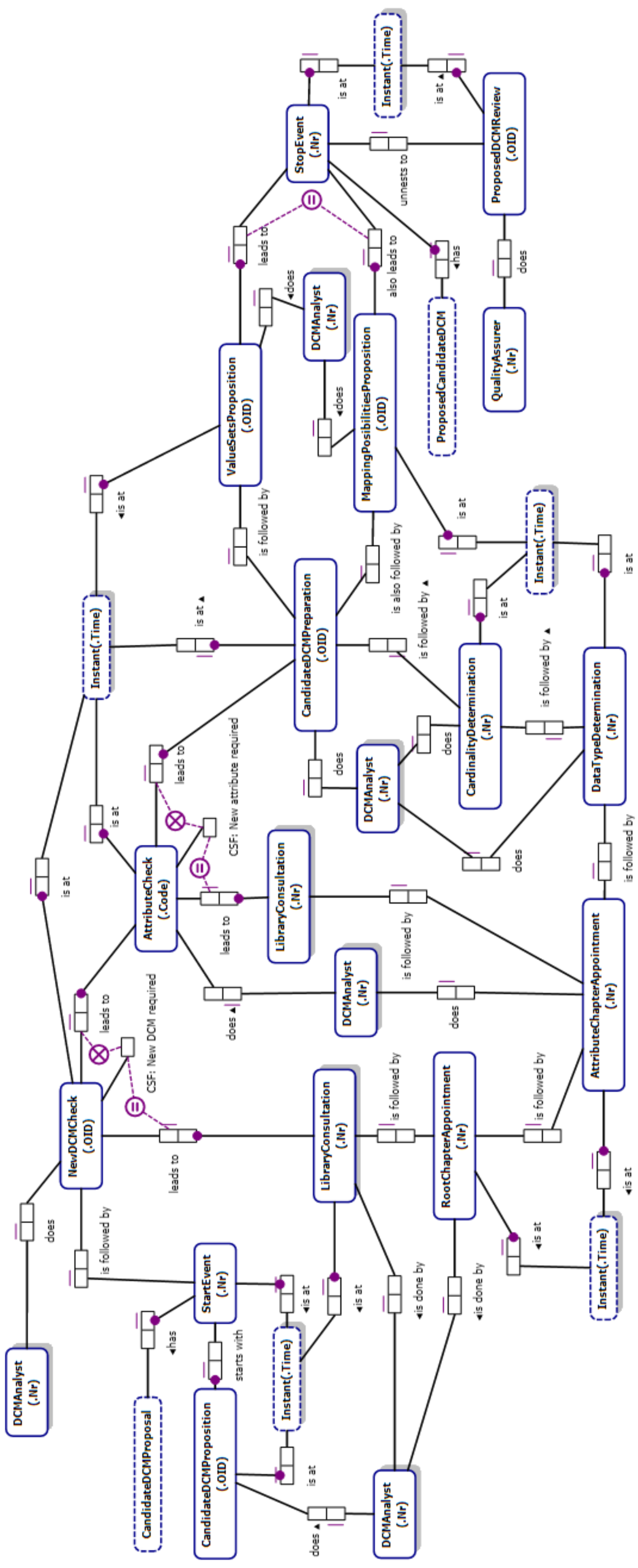


Figure 27: ORM diagram of Propose Candidate DCM.

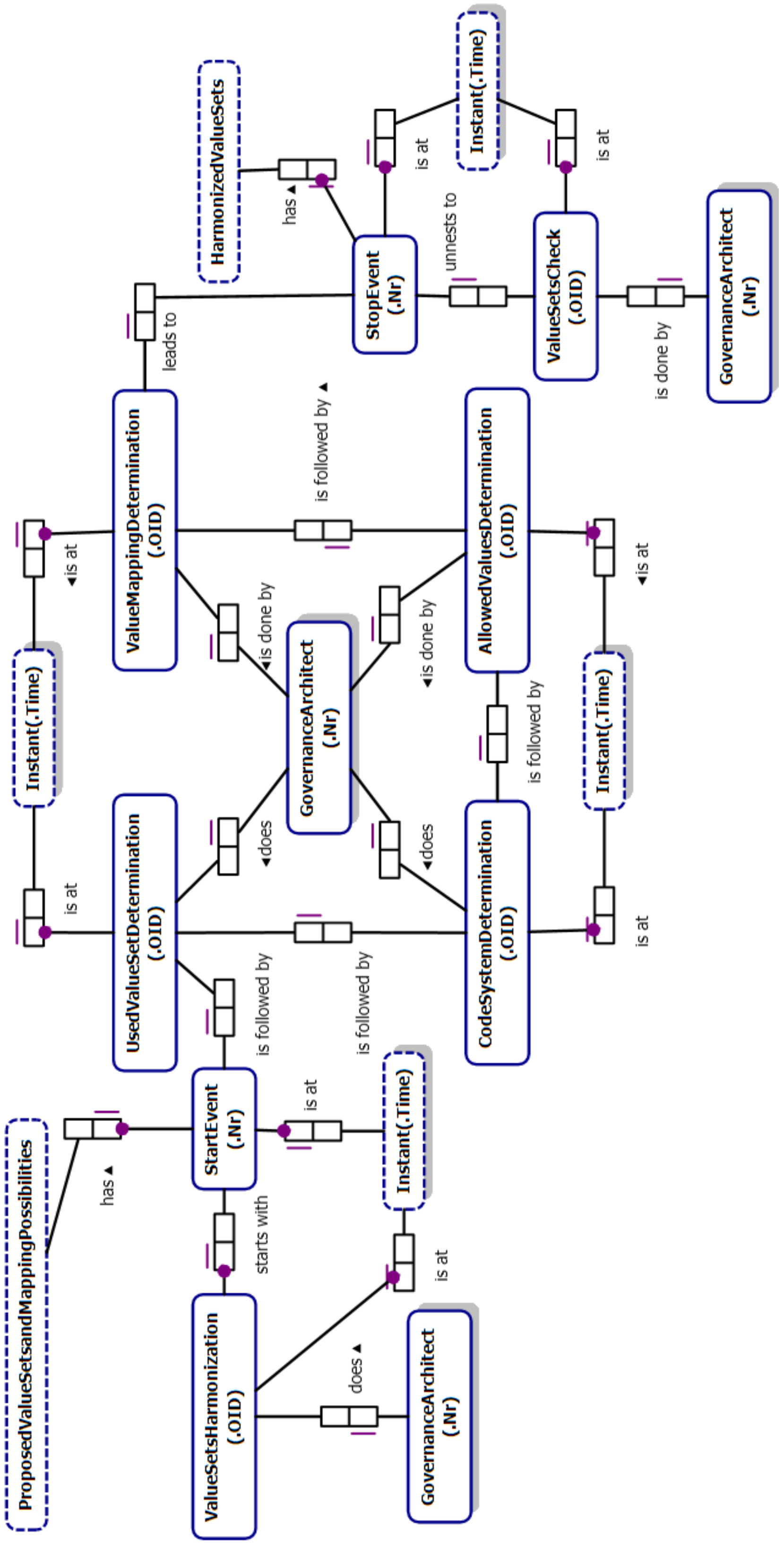


Figure 28: ORM diagram of Harmonize Value sets.

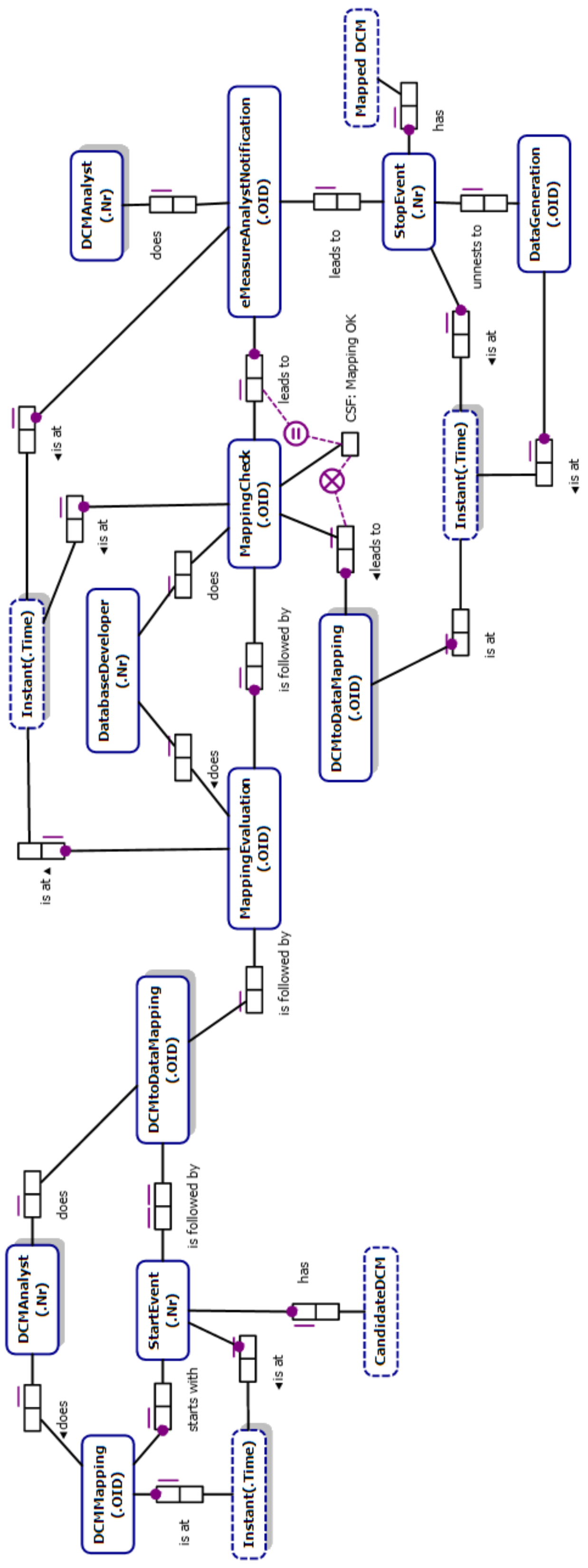


Figure 29: ORM diagram op Map DCM.

Appendix 5: Database blueprints

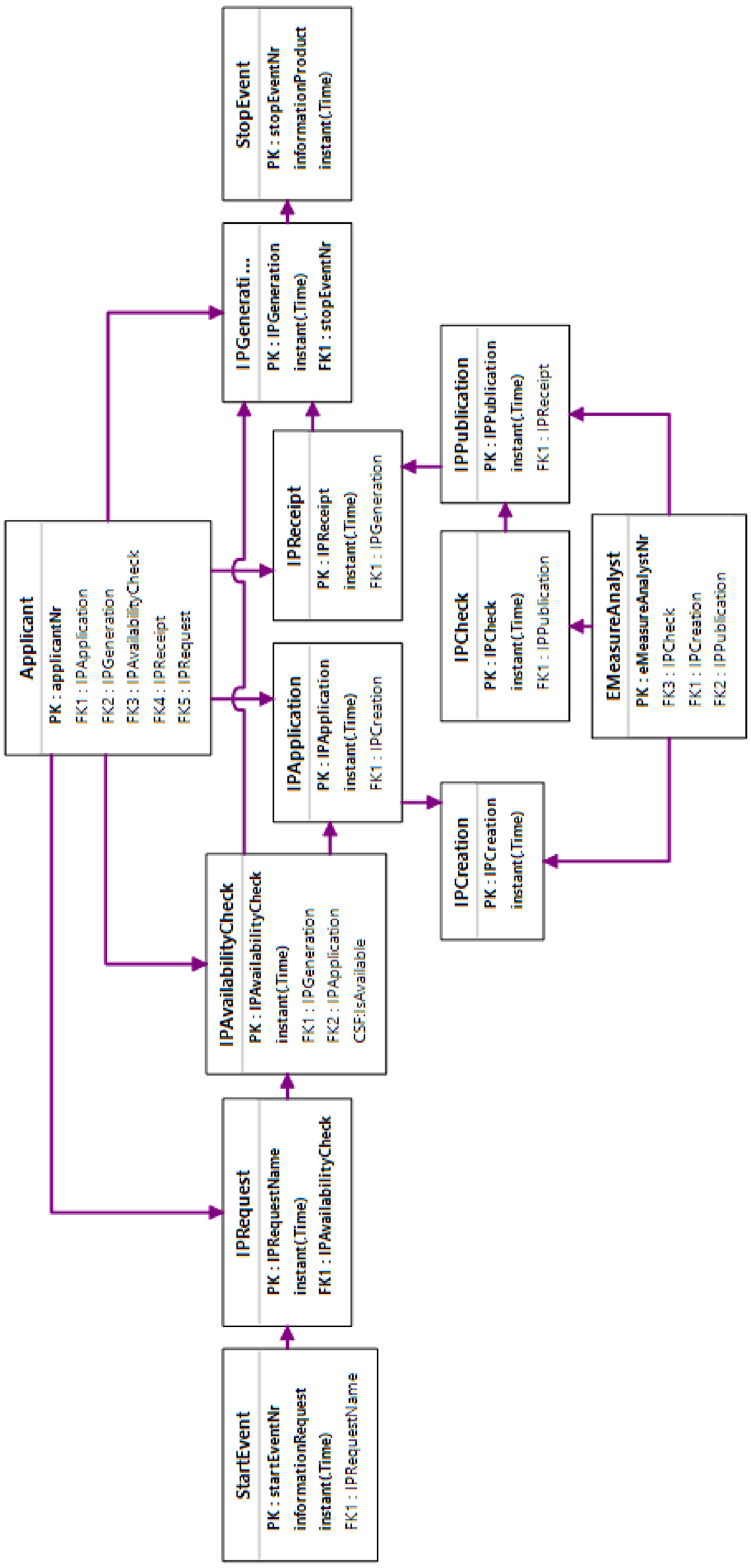


Figure 30: Database blueprint of the first part of the Universe of Discourse.

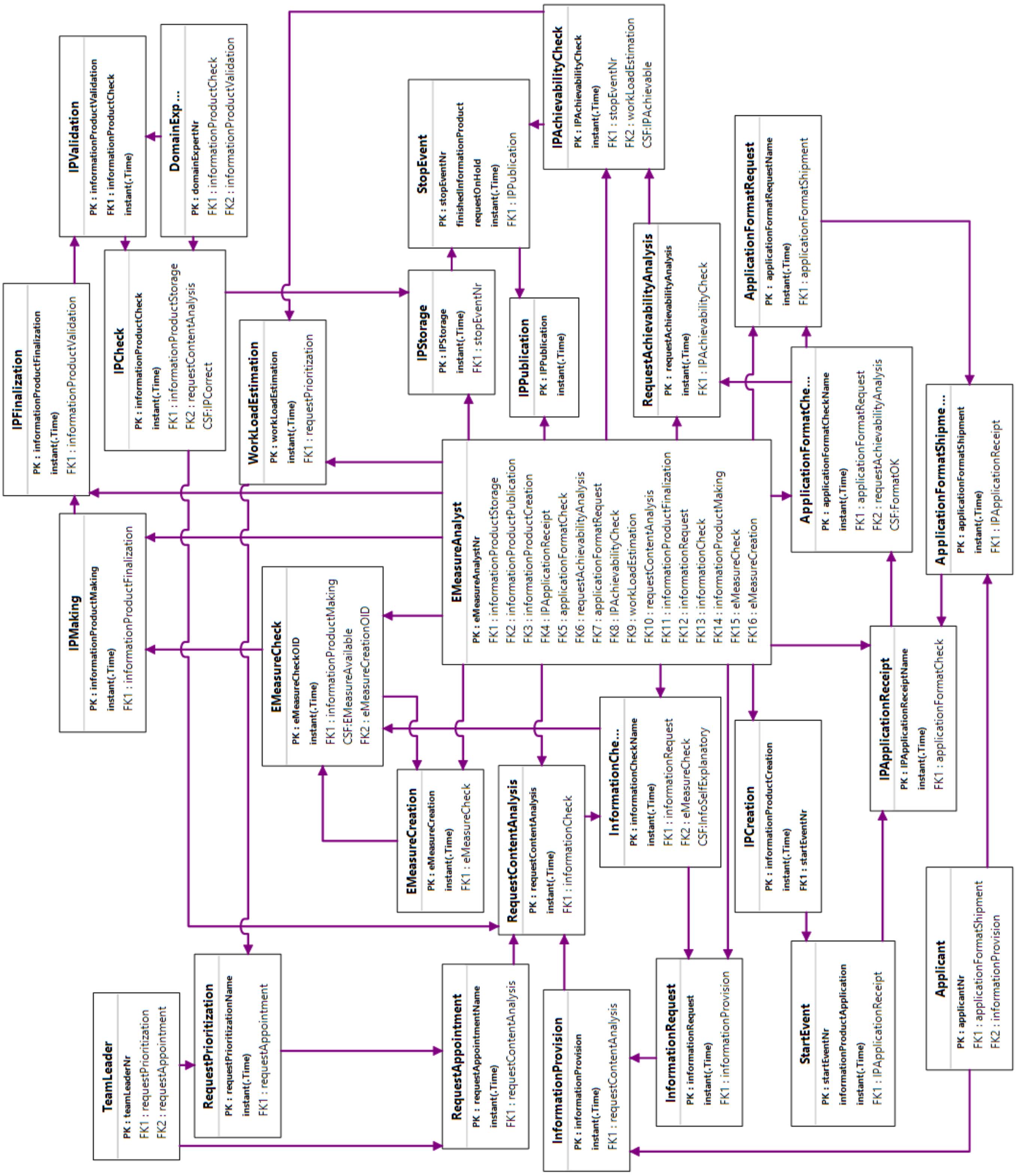

IPValidation
PK : informationProductValidation
FK1 : informationProductCheck
instant(.Time)
DomainExp...
PK : domainExpertNr
FK1 : informationProductCheck
FK2 : informationProductValidation
IPFinalization
PK : informationProductFinalization
instant(.Time)
FK1 : informationProductValidation
IPCheck
PK : informationProductCheck
instant(.Time)
FK1 : informationProductStorage
FK2 : requestContentAnalysis
CSF:IPCorrect
WorkLoadEstimation
PK : workLoadEstimation
instant(.Time)
FK1 : requestPrioritization
StopEvent
PK : stopEventNr
finishedInformationProduct
requestOnHold
instant(.Time)
FK1 : IPPublication
IPStorage
PK : IPStorage
instant(.Time)
FK1 : stopEventNr
IPPublication
PK : IPPublication
instant(.Time)
IPAchievabilityCheck
PK : IPAchievabilityCheck
instant(.Time)
FK1 : stopEventNr
FK2 : workLoadEstimation
CSF:IPAchievable
RequestAchievabilityAnalysis
PK : requestAchievabilityAnalysis
instant(.Time)
FK1 : IPAchievabilityCheck
ApplicationFormatRequest
PK : applicationFormatRequestName
instant(.Time)
FK1 : applicationFormatShipment
ApplicationFormatChe...
PK : applicationFormatCheckName
instant(.Time)
FK1 : applicationFormatRequest
FK2 : requestAchievabilityAnalysis
CSF:FormatOK
ApplicationFormatShipme...
PK : applicationFormatShipment
instant(.Time)
FK1 : IPApplicationReceipt
EMeasureAnalyst
PK : eMeasureAnalystNr
FK1 : informationProductStorage
FK2 : informationProductPublication
FK3 : informationProductCreation
FK4 : IPApplicationReceipt
FK5 : applicationFormatCheck
FK6 : requestAchievabilityAnalysis
FK7 : applicationFormatRequest
FK8 : IPAchievabilityCheck
FK9 : workLoadEstimation
FK10 : requestContentAnalysis
FK11 : informationProductFinalization
FK12 : informationRequest
FK13 : informationCheck
FK14 : informationProductMaking
FK15 : eMeasureCheck
FK16 : eMeasureCreation
IPMaking
PK : informationProductMaking
instant(.Time)
FK1 : informationProductFinalization
EMeasureCheck
PK : eMeasureCheckOID
instant(.Time)
FK1 : informationProductMaking
CSF:EMeasureAvailable
FK2 : eMeasureCreationOID
EMeasureCreation
PK : eMeasureCreation
instant(.Time)
FK1 : eMeasureCheck
RequestContentAnalysis
PK : requestContentAnalysis
instant(.Time)
FK1 : informationCheck
InformationChe...
PK : informationCheckName
instant(.Time)
FK1 : informationRequest
FK2 : eMeasureCheck
CSF:InfoSelfExplanatory
IPCreation
PK : informationProductCreation
instant(.Time)
FK1 : startEventNr
IPApplicationReceipt
PK : IPApplicationReceiptName
instant(.Time)
FK1 : applicationFormatCheck
TeamLeader
PK : teamLeaderNr
FK1 : requestPrioritization
FK2 : requestAppointment
RequestPrioritization
PK : requestPrioritizationName
instant(.Time)
FK1 : requestAppointment
RequestAppointment
PK : requestAppointmentName
instant(.Time)
FK1 : requestContentAnalysis
InformationProvision
PK : informationProvision
instant(.Time)
FK1 : requestContentAnalysis
InformationRequest
PK : informationRequest
instant(.Time)
FK1 : informationProvision
StartEvent
PK : startEventNr
informationProductApplication
instant(.Time)
FK1 : IPApplicationReceipt
Applicant
PK : applicantNr
FK1 : applicationFormatShipment
FK2 : informationProvision

Figure 31: Database blueprint of Create Information Product (IP).

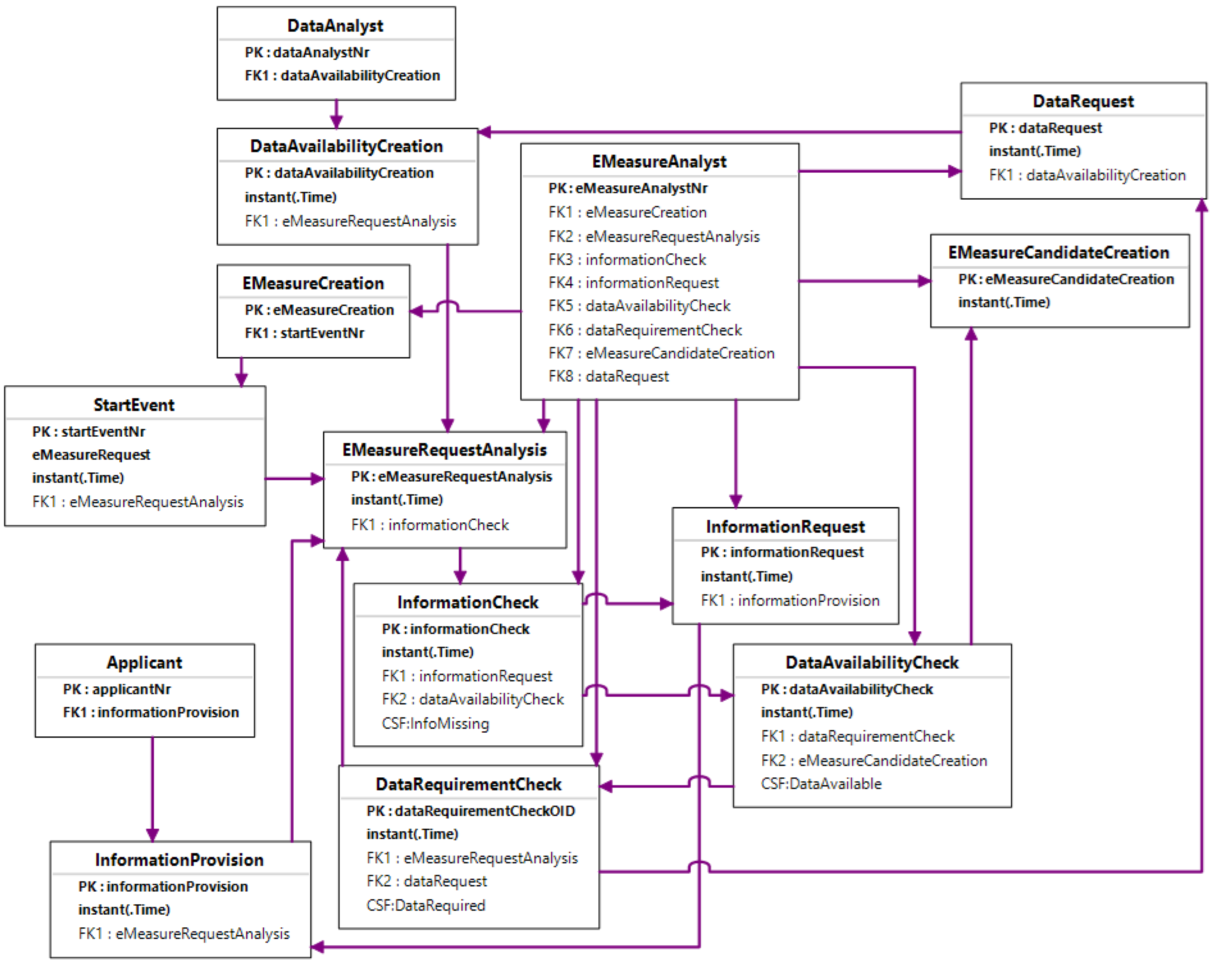


Figure 32: Database blueprint of Create eMeasure (first part).

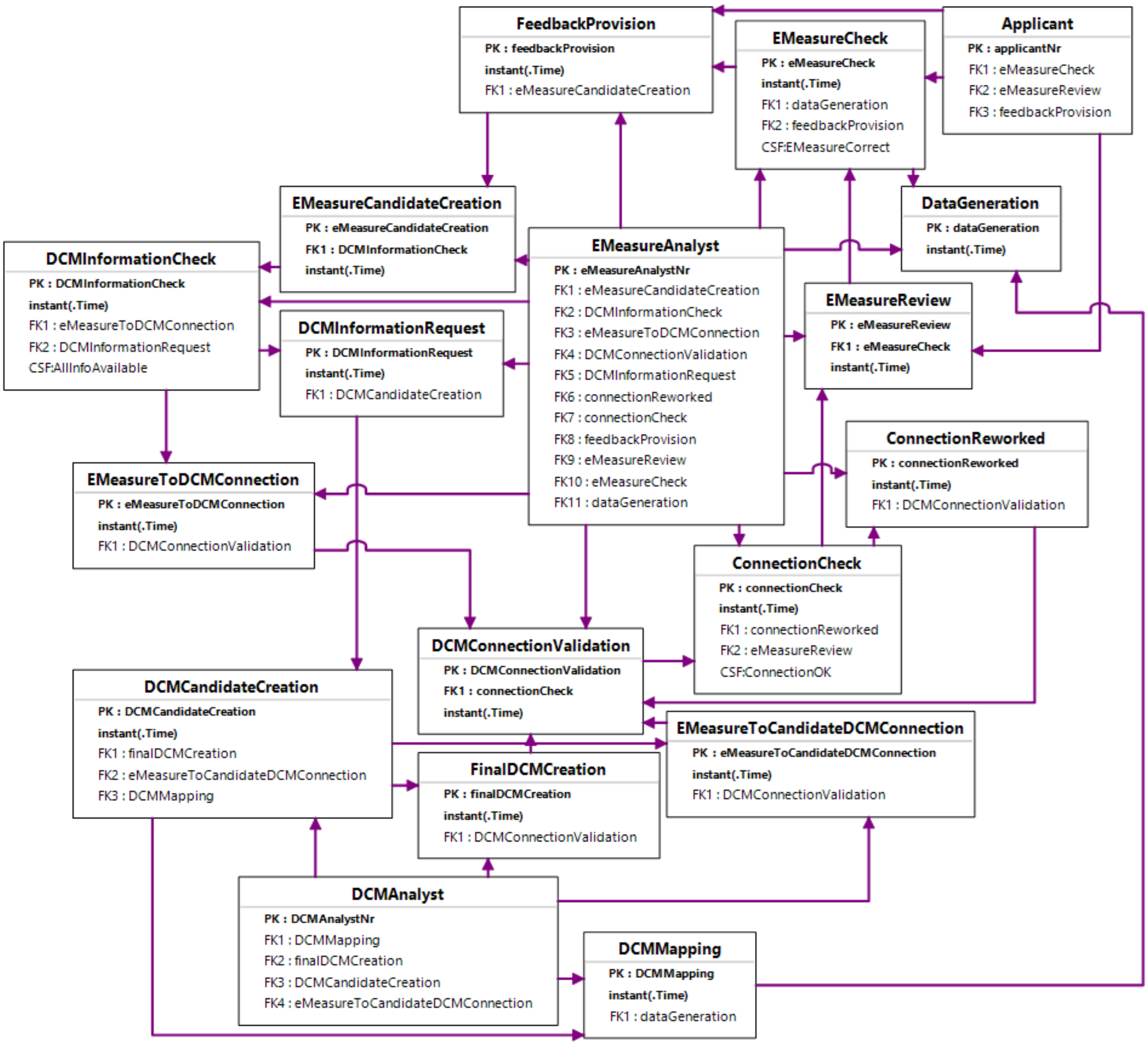


Figure 33: Database blueprint of Create eMeasure (second part).

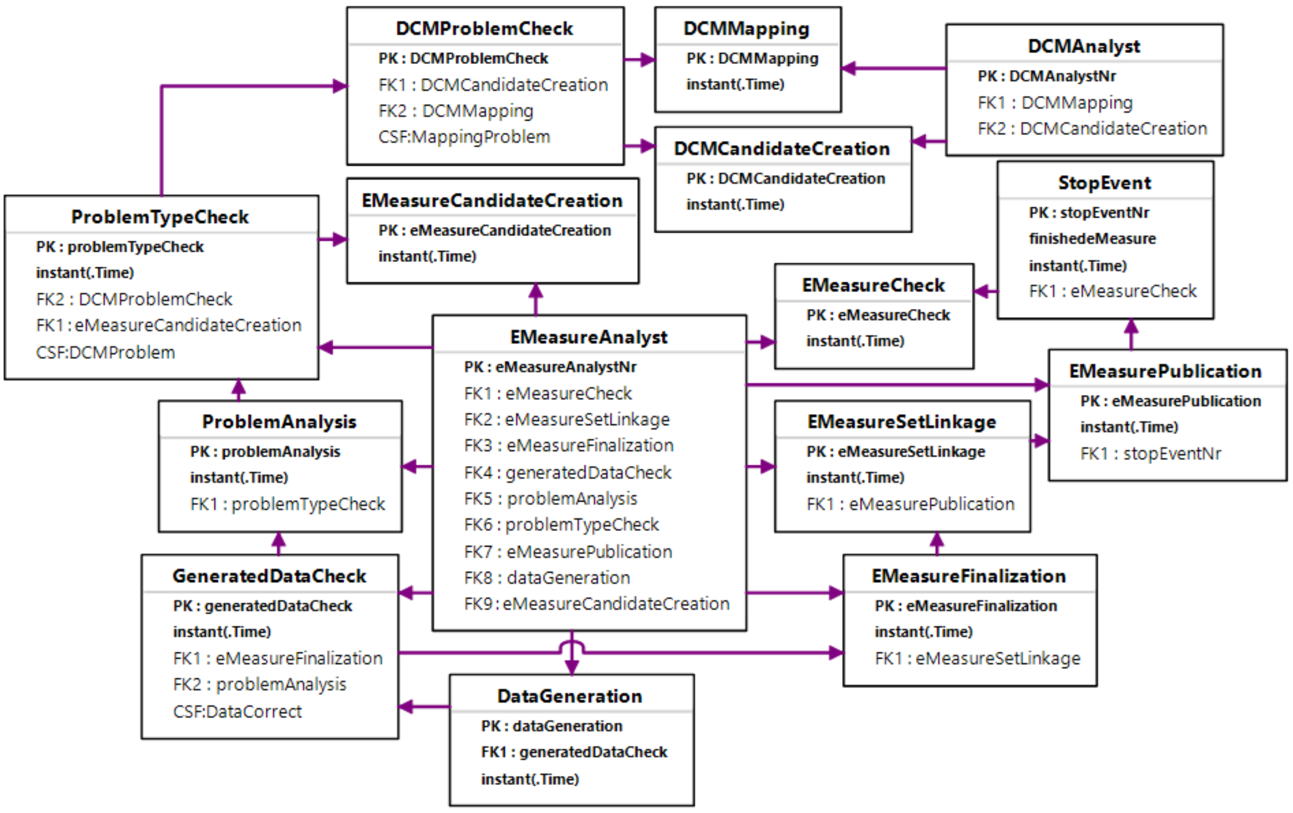


Figure 34: Database blueprint of Create eMeasure (third part).

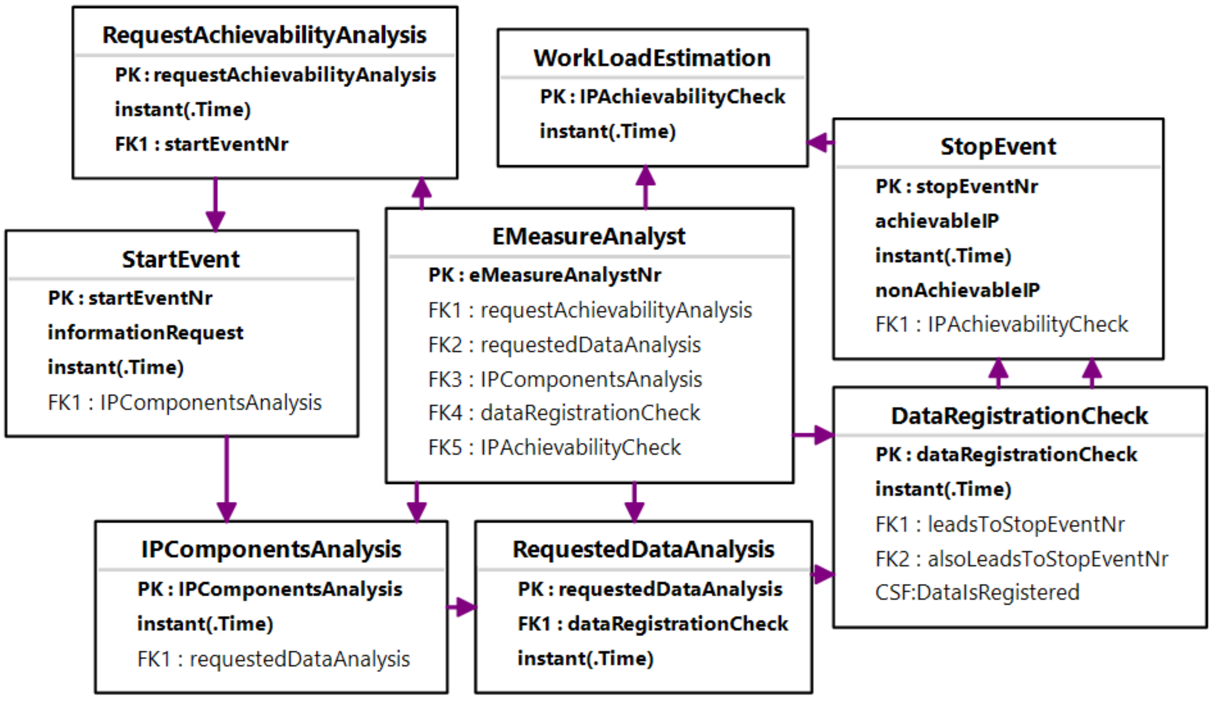

RequestAchievabilityAnalysis
PK : requestAchievabilityAnalysis
instant(.Time)
FK1 : startEventNr
WorkLoadEstimation
PK : IPAchievabilityCheck
instant(.Time)
StopEvent
PK : stopEventNr
achievableIP
instant(.Time)
nonAchievableIP
FK1 : IPAchievabilityCheck
StartEvent
PK : startEventNr
informationRequest
instant(.Time)
FK1 : IPComponentsAnalysis
EMeasureAnalyst
PK : eMeasureAnalystNr
FK1 : requestAchievabilityAnalysis
FK2 : requestedDataAnalysis
FK3 : IPComponentsAnalysis
FK4 : dataRegistrationCheck
FK5 : IPAchievabilityCheck
DataRegistrationCheck
PK : dataRegistrationCheck
instant(.Time)
FK1 : leadsToStopEventNr
FK2 : alsoLeadsToStopEventNr
CSF:DataIsRegistered
IPComponentsAnalysis
PK : IPComponentsAnalysis
instant(.Time)
FK1 : requestedDataAnalysis
RequestedDataAnalysis
PK : requestedDataAnalysis
FK1 : dataRegistrationCheck
instant(.Time)

Figure 35: Database blueprint of Analyze Request on Achievability.

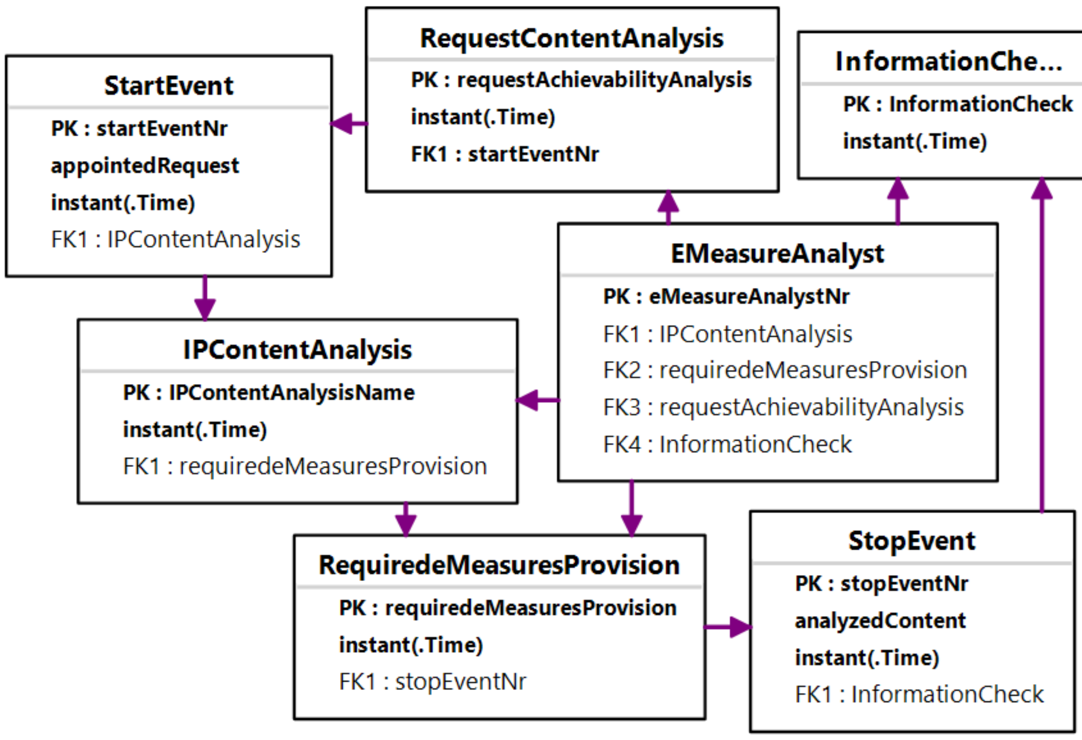


Figure 36: Database blueprint of Analyze Request on Content.

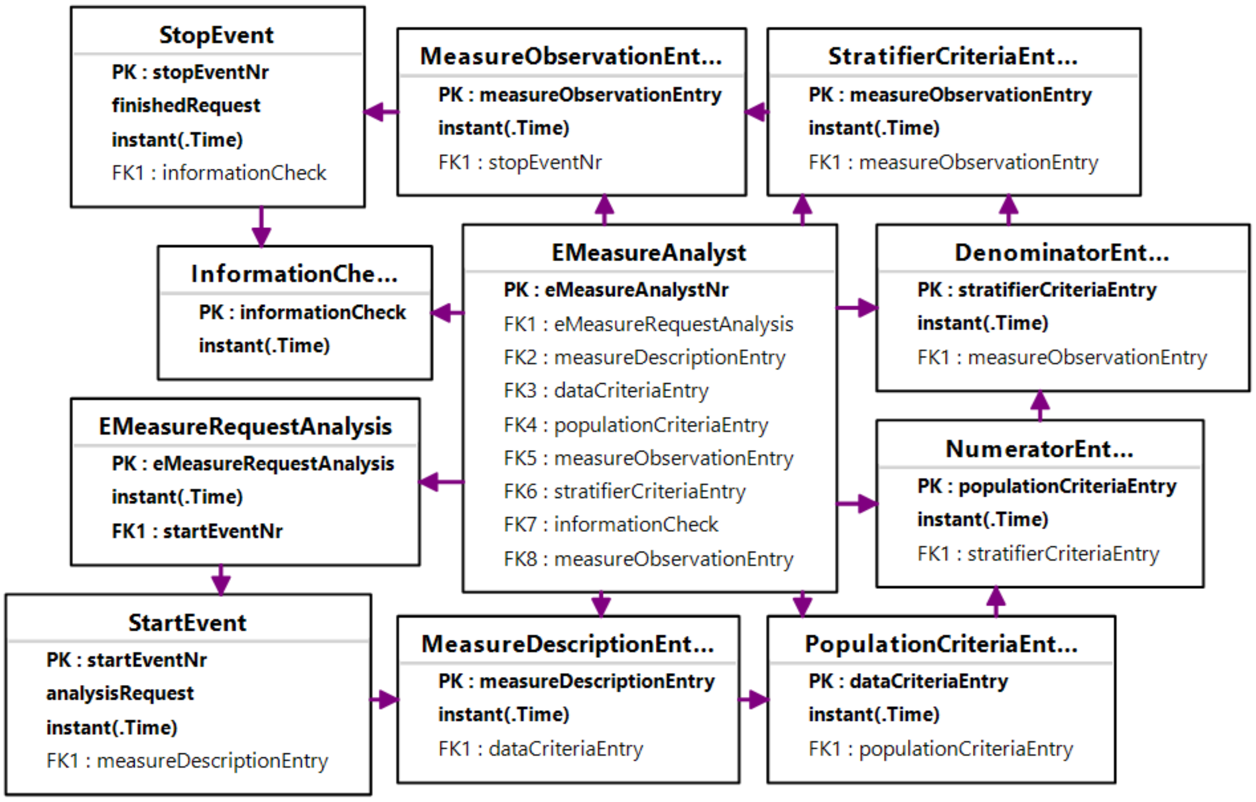


Figure 37: Database blueprint of Analyze eMeasure Request.

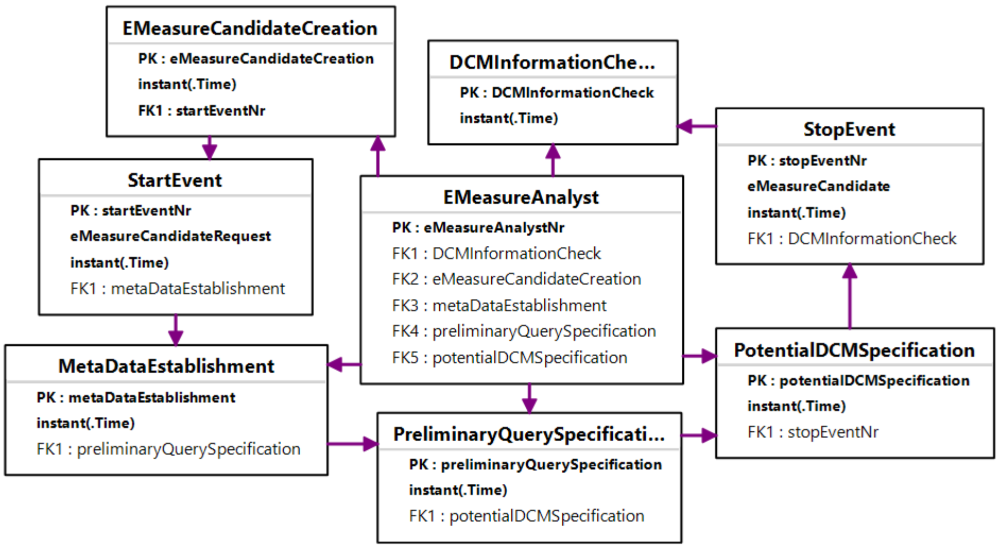

EMeasureCandidateCreation
PK : eMeasureCandidateCreation
instant(.Time)
FK1 : startEventNr
DCMInformationChe...
PK : DCMInformationCheck
instant(.Time)
StopEvent
PK : stopEventNr
eMeasureCandidate
instant(.Time)
FK1 : DCMInformationCheck
StartEvent
PK : startEventNr
eMeasureCandidateRequest
instant(.Time)
FK1 : metaDataEstablishment
EMeasureAnalyst
PK : eMeasureAnalystNr
FK1 : DCMInformationCheck
FK2 : eMeasureCandidateCreation
FK3 : metaDataEstablishment
FK4 : preliminaryQuerySpecification
FK5 : potentialDCMSpecification
MetaDataEstablishment
PK : metaDataEstablishment
instant(.Time)
FK1 : preliminaryQuerySpecification
PotentialDCMSpecification
PK : potentialDCMSpecification
instant(.Time)
FK1 : stopEventNr
PreliminaryQuerySpecificati...
PK : preliminaryQuerySpecification
instant(.Time)
FK1 : potentialDCMSpecification

Figure 38: Database blueprint of Create Candidate eMeasure.

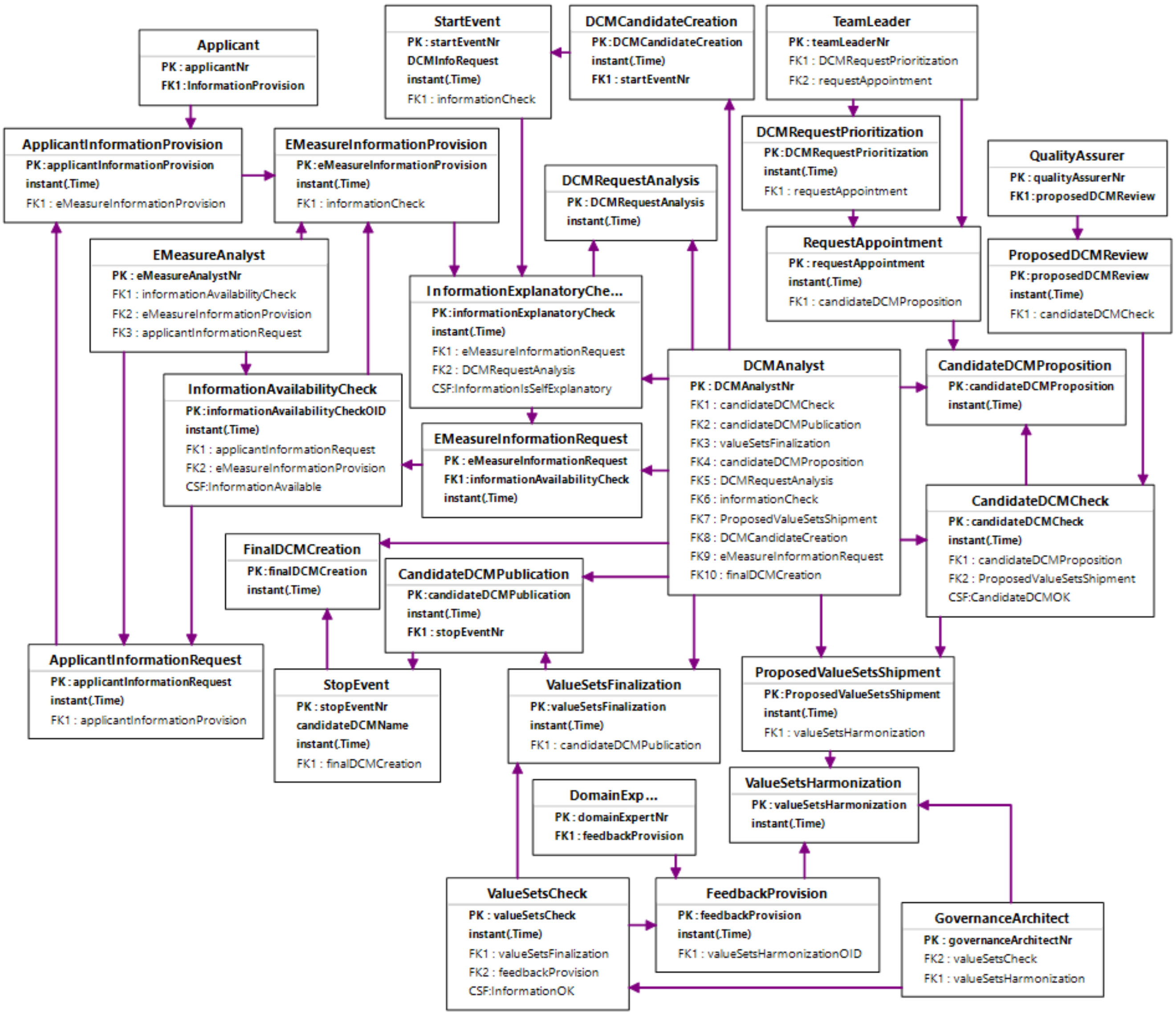


Figure 39: Database blueprint of Make candidate DCM.

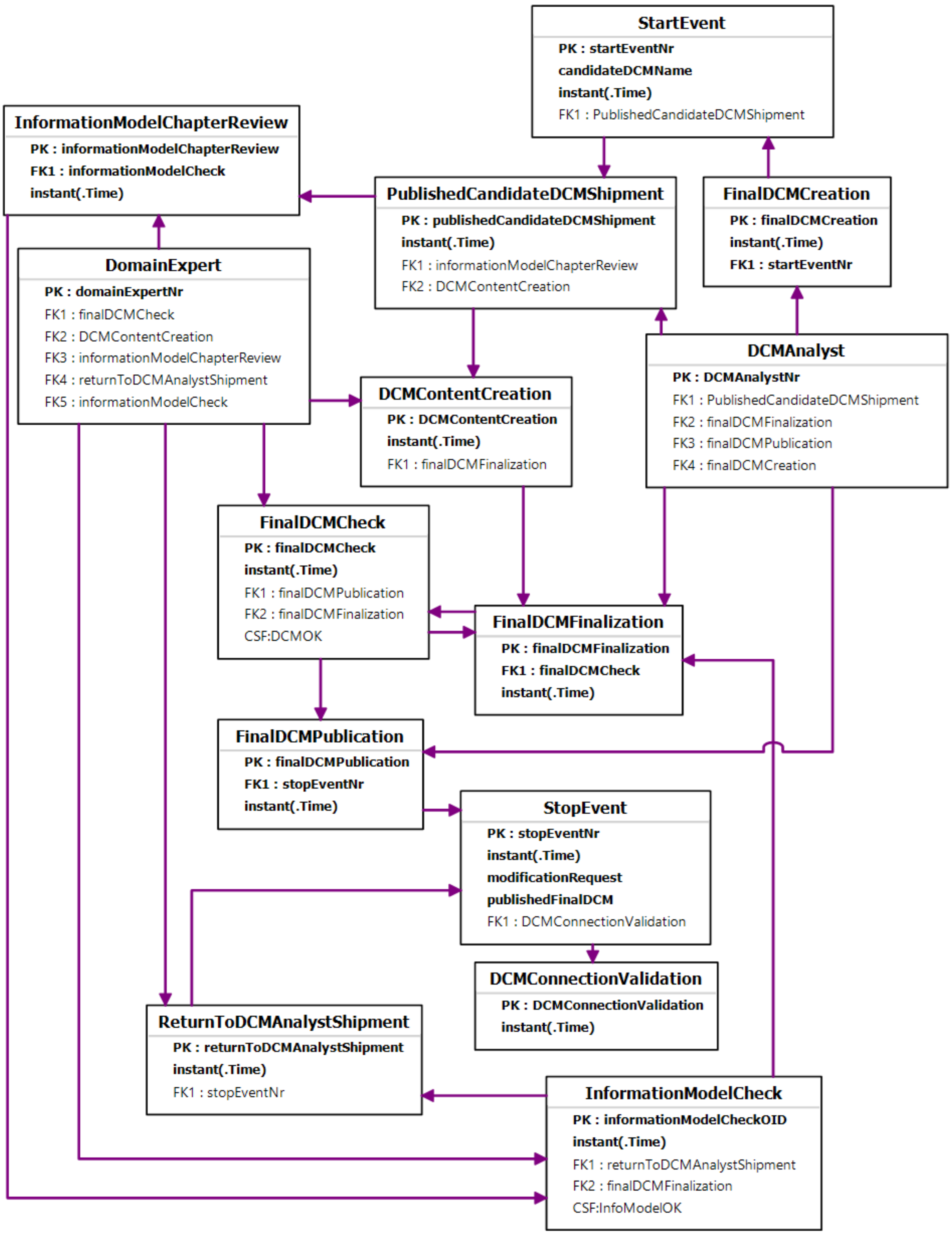
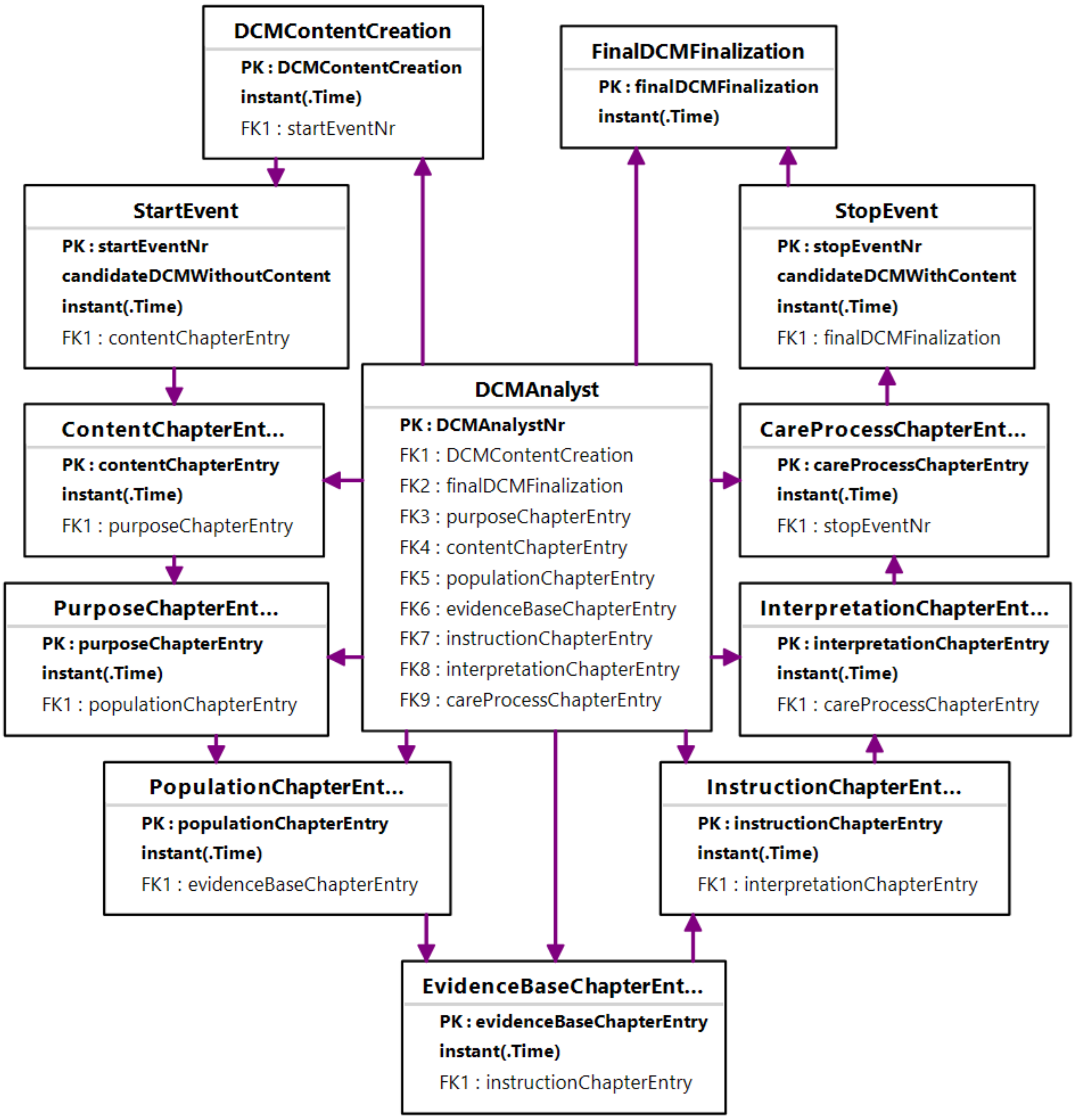


Figure 40: Database blueprint of Make final DCM.



DCMContentCreation
PK : DCMContentCreation
instant(.Time)
FK1 : startEventNr
FinalDCMFinalization
PK : finalDCMFinalization
instant(.Time)
StartEvent
PK : startEventNr
candidateDCMWithoutContent
instant(.Time)
FK1 : contentChapterEntry
StopEvent
PK : stopEventNr
candidateDCMWithContent
instant(.Time)
FK1 : finalDCMFinalization
DCMAnalyst
PK : DCMAnalystNr
FK1 : DCMContentCreation
FK2 : finalDCMFinalization
FK3 : purposeChapterEntry
FK4 : contentChapterEntry
FK5 : populationChapterEntry
FK6 : evidenceBaseChapterEntry
FK7 : instructionChapterEntry
FK8 : interpretationChapterEntry
FK9 : careProcessChapterEntry
ContentChapterEnt...
PK : contentChapterEntry
instant(.Time)
FK1 : purposeChapterEntry
CareProcessChapterEnt...
PK : careProcessChapterEntry
instant(.Time)
FK1 : stopEventNr
PurposeChapterEnt...
PK : purposeChapterEntry
instant(.Time)
FK1 : populationChapterEntry
InterpretationChapterEnt...
PK : interpretationChapterEntry
instant(.Time)
FK1 : careProcessChapterEntry
PopulationChapterEnt...
PK : populationChapterEntry
instant(.Time)
FK1 : evidenceBaseChapterEntry
InstructionChapterEnt...
PK : instructionChapterEntry
instant(.Time)
FK1 : interpretationChapterEntry
EvidenceBaseChapterEnt...
PK : evidenceBaseChapterEntry
instant(.Time)
FK1 : instructionChapterEntry

Figure 41: Database blueprint of Create DCM Content.

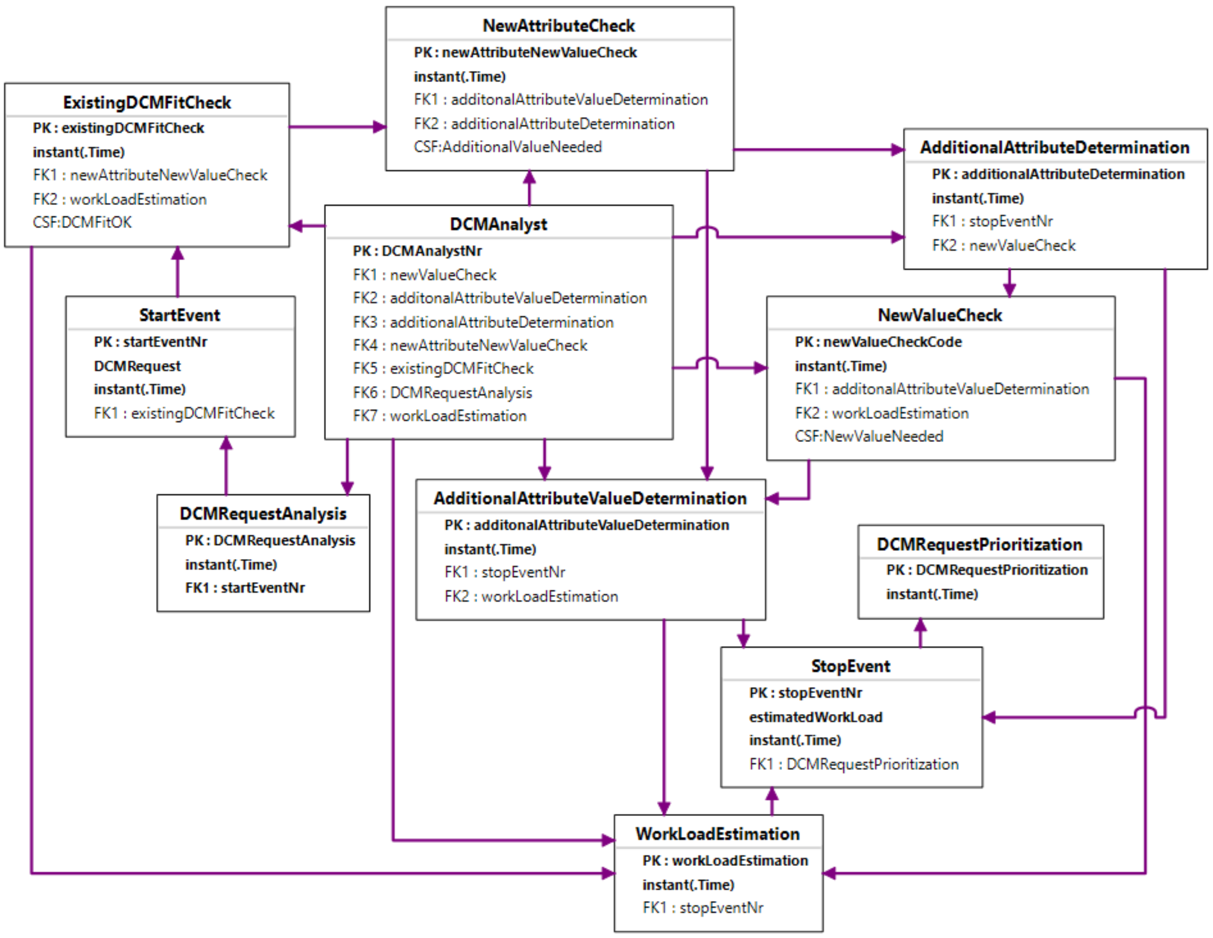


Figure 42: Database blueprint of Analyze DCM request.

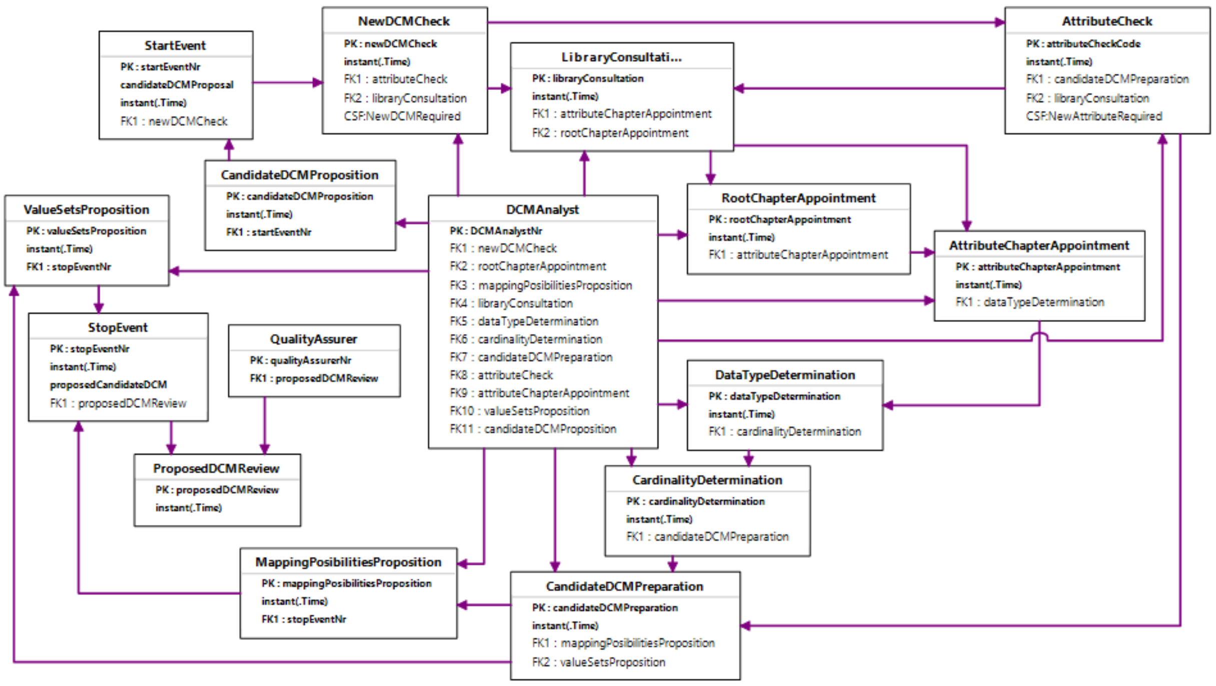

StartEvent
PK : startEventNr
candidateDCMProposal
instant(.Time)
FK1 : newDCMCheck
NewDCMCheck
PK : newDCMCheck
instant(.Time)
FK1 : attributeCheck
FK2 : libraryConsultation
CSF:NewDCMRequired
LibraryConsultati...
PK : libraryConsultation
instant(.Time)
FK1 : attributeChapterAppointment
FK2 : rootChapterAppointment
AttributeCheck
PK : attributeCheckCode
instant(.Time)
FK1 : candidateDCMPreparation
FK2 : libraryConsultation
CSF:NewAttributeRequired
CandidateDCMProposition
PK : candidateDCMProposition
instant(.Time)
FK1 : startEventNr
ValueSetsProposition
PK : valueSetsProposition
instant(.Time)
FK1 : stopEventNr
DCMAnalyst
PK : DCMAnalystNr
FK1 : newDCMCheck
FK2 : rootChapterAppointment
FK3 : mappingPosibilitiesProposition
FK4 : libraryConsultation
FK5 : dataTypeDetermination
FK6 : cardinalityDetermination
FK7 : candidateDCMPreparation
FK8 : attributeCheck
FK9 : attributeChapterAppointment
FK10 : valueSetsProposition
FK11 : candidateDCMProposition
RootChapterAppointment
PK : rootChapterAppointment
instant(.Time)
FK1 : attributeChapterAppointment
AttributeChapterAppointment
PK : attributeChapterAppointment
instant(.Time)
FK1 : dataTypeDetermination
StopEvent
PK : stopEventNr
instant(.Time)
proposedCandidateDCM
FK1 : proposedDCMReview
QualityAssurer
PK : qualityAssurerNr
FK1 : proposedDCMReview
DataTypeDetermination
PK : dataTypeDetermination
instant(.Time)
FK1 : cardinalityDetermination
ProposedDCMReview
PK : proposedDCMReview
instant(.Time)
CardinalityDetermination
PK : cardinalityDetermination
instant(.Time)
FK1 : candidateDCMPreparation
MappingPosibilitiesProposition
PK : mappingPosibilitiesProposition
instant(.Time)
FK1 : stopEventNr
CandidateDCMPreparation
PK : candidateDCMPreparation
instant(.Time)
FK1 : mappingPosibilitiesProposition
FK2 : valueSetsProposition

Figure 43: Database blueprint of Propose Candidate DCM.

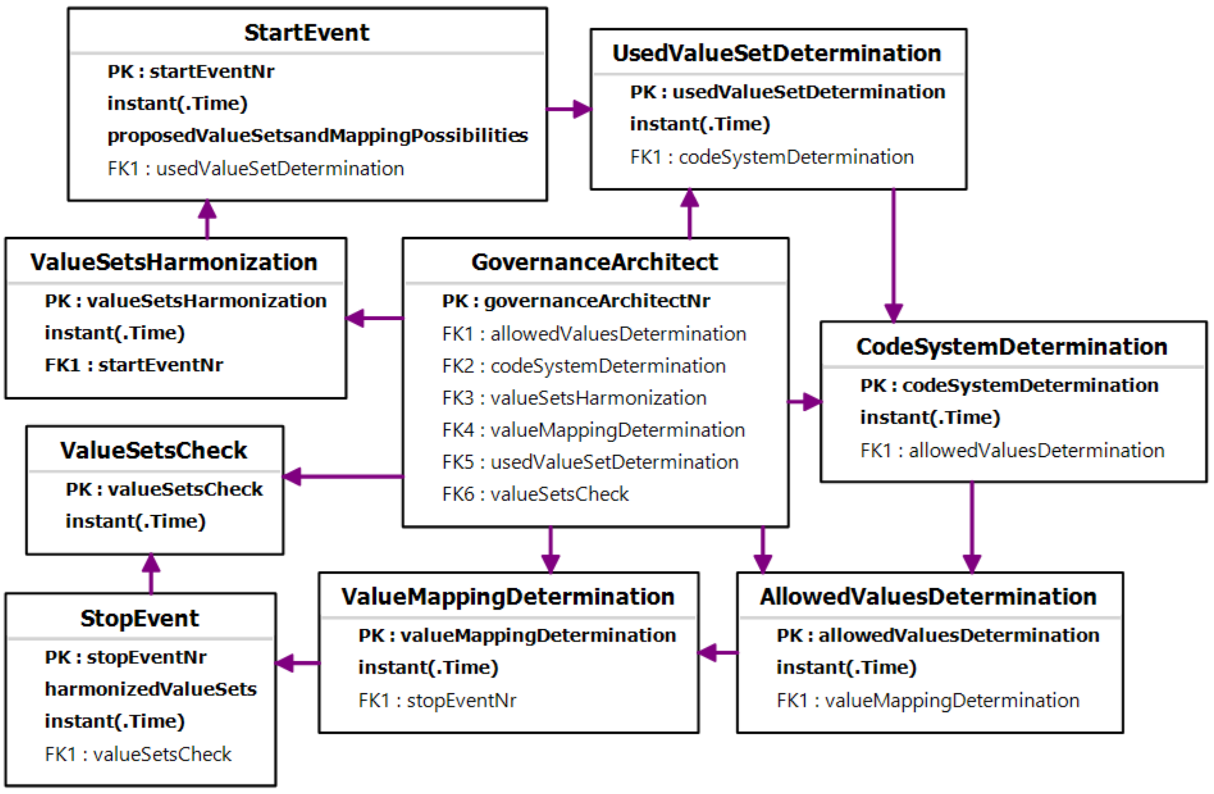

StartEvent
PK : startEventNr
instant(.Time)
proposedValueSetsandMappingPossibilities
FK1 : usedValueSetDetermination
UsedValueSetDetermination
PK : usedValueSetDetermination
instant(.Time)
FK1 : codeSystemDetermination
ValueSetsHarmonization
PK : valueSetsHarmonization
instant(.Time)
FK1 : startEventNr
GovernanceArchitect
PK : governanceArchitectNr
FK1 : allowedValuesDetermination
FK2 : codeSystemDetermination
FK3 : valueSetsHarmonization
FK4 : valueMappingDetermination
FK5 : usedValueSetDetermination
FK6 : valueSetsCheck
CodeSystemDetermination
PK : codeSystemDetermination
instant(.Time)
FK1 : allowedValuesDetermination
ValueSetsCheck
PK : valueSetsCheck
instant(.Time)
StopEvent
PK : stopEventNr
harmonizedValueSets
instant(.Time)
FK1 : valueSetsCheck
ValueMappingDetermination
PK : valueMappingDetermination
instant(.Time)
FK1 : stopEventNr
AllowedValuesDetermination
PK : allowedValuesDetermination
instant(.Time)
FK1 : valueMappingDetermination

Figure 44: Database blueprint of Harmonize Value sets.

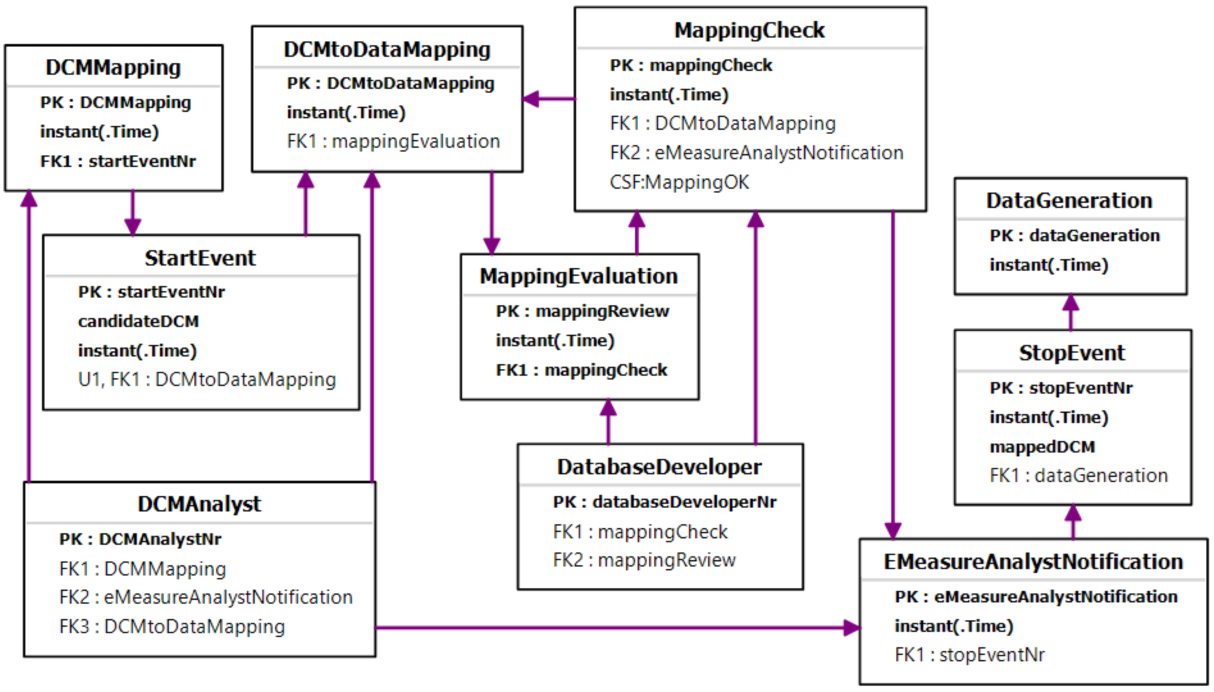


Figure 45: Database blueprint of Map DCM.

Appendix 6: Example of the designed process in practice

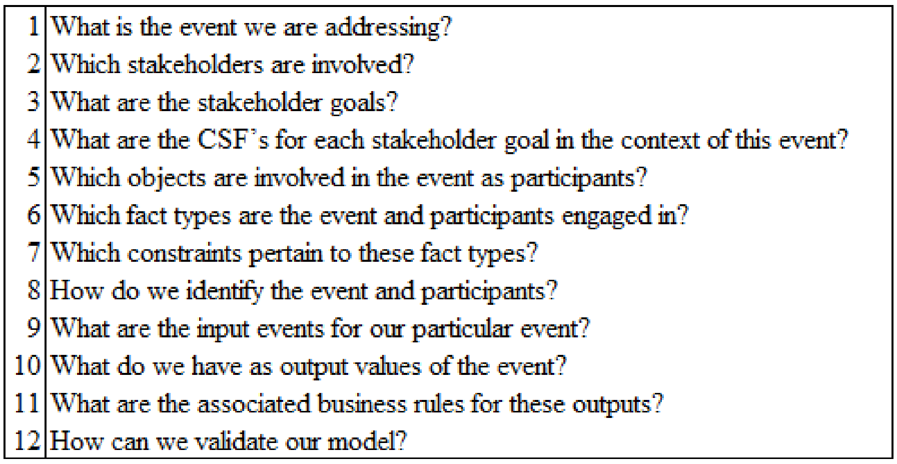
This appendix discusses the designed process in reality and uses a knee replacement indicator request as an example. The aim of this appendix is to create a better understanding of the actual designed process. Table 1 is an example of a request as received by the LTHN. The LTHN receives these in Dutch, hence the example is in Dutch.

Indicator 1: Antibiotische profylaxe	
Relatie met kwaliteit van zorg	Antibiotische profylaxe is bewezen effectief in de preventie van diepe wondinfecties bij plaatsing van een totale knieprothese. Het optreden van diepe wondinfecties wordt mede beïnvloed door aanwezige co-morbiditeit (o.a. hypertensie, diabetes, obesitas). Deze antibiotische profylaxe moet wel op het juiste ogenblik wordt gestart: 60 tot 15 minuten vóór de incisie of vóór het opwekken van bloedleegte kan als het optimale tijdsinterval worden beschouwd. Het percentage wondinfecties dat na opereren optreedt weerspiegelt de kwaliteit van de geboden profylaxe. Overigens moet er rekening mee worden gehouden dat ook patiëntenkenmerken zoals de gezondheidsstatus het optreden van wondinfecties kunnen beïnvloeden.
Operationalisatie 1a	Is er een richtlijn of protocol beschikbaar voor antibiotische profylaxe in geval van een totale knieprothese? *Ja/Nee*
Operationalisatie 1b	Percentage operaties waarbij de patiënt peri-operatief antibiotica toegediend heeft gekregen, in geval van een totale knieprothese.
Teller 1b	Aantal operaties waarbij de patiënt peri-operatief antibiotica toegediend heeft gekregen, in geval van een totale knieprothese
Noemer 1b	Aantal operaties waarbij de patiënt een totale knieprothese heeft ondergaan
Operationalisatie 1c	Percentage operaties waarbij de patiënt 60 tot 15 minuten vóór de incisie of vóór het opwekken van bloedleegte antibiotica toegediend heeft gekregen, in geval van een totale knieprothese.
Teller 1c	Aantal operaties waarbij de patiënt 60 tot 15 minuten vóór de incisie of vóór het opwekken van bloedleegte antibiotica toegediend heeft gekregen, in geval van een totale knieprothese
Noemer 1c	Aantal operaties waarbij de patiënt peri-operatief antibiotica toegediend heeft gekregen, in geval van een totale knieprothese
Operationalisatie 1d	Percentage diepe wondinfecties in geval van een totale knieprothese
Teller 1d	Aantal diepe wondinfecties tot zes weken na de operatie bij patiënten in geval van een totale knieprothese
Noemer 1d	Aantal operaties waarbij de patiënt een totale knieprothese heeft ondergaan

Definities	Peri-operatief: Gedurende de klinische opname De volgende definitie (WIP) van een diepe wondinfectie is van toepassing. De infectie is ontstaan binnen 1 jaar na operatie en de infectie lijkt het gevolg te zijn van de operatie en betreft de diepliggende weefsels van de incisie (zoals fascie en spier) en voldoet bovendien aan één of meer van de volgende bevindingen: 1) Purulente afscheiding uit een diepe incisie maar niet van de organen en anatomische ruimten van het operatiegebied. 2) Spontane wonddehiscentie of wond geopend door de chirurg terwijl de patiënt koorts (>38°C) en/of lokale pijn of gevoeligheid heeft tenzij een wondkweek negatief blijkt. 3) Abces of ander teken van infectie van het gebied van de diepe incisie gezien bij directe observatie, tijdens heroperatie of histopathologisch of radiologisch onderzoek. 4) Diagnose 'diepe infectie van het operatiegebied' door de chirurg of behandelend arts. NB: Infecties die zowel oppervlakkig als diep zijn worden geclassificeerd als diepe postoperatieve infecties van het operatiegebied.
In-/exclusiecriteria	1d: Exclusie: Patiënten met ASA-klasse □ 3
Bron	1a: Richtlijnen of protocollen 1b: Datamanagementsysteem anesthesiologie, anesthesielijst in patiëntendossier (teller), ZIS, DBC- en verrichtingenregistratie (noemer) 1c: Datamanagementsysteem anesthesiologie, anesthesielijst in

Table 1: Example request for a knee replacement indicator set.

The example will be converted to data by using the proposed BPMN-ORM methodology.



1	What is the event we are addressing?
2	Which stakeholders are involved?
3	What are the stakeholder goals?
4	What are the CSF's for each stakeholder goal in the context of this event?
5	Which objects are involved in the event as participants?
6	Which fact types are the event and participants engaged in?
7	Which constraints pertain to these fact types?
8	How do we identify the event and participants?
9	What are the input events for our particular event?
10	What do we have as output values of the event?
11	What are the associated business rules for these outputs?
12	How can we validate our model?

Table 2: Proposed BPMN-ORM methodology.

This sub-process is chosen as it provides an excellent example for the conversion of BPMN to ORM (and corresponding database blueprint) based on the knee replacement indicator set.

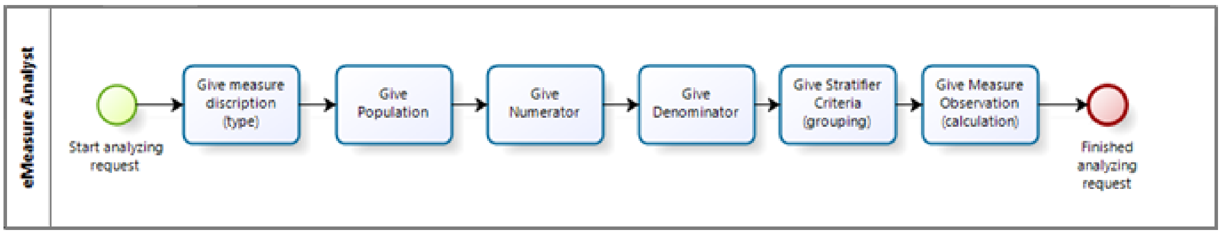
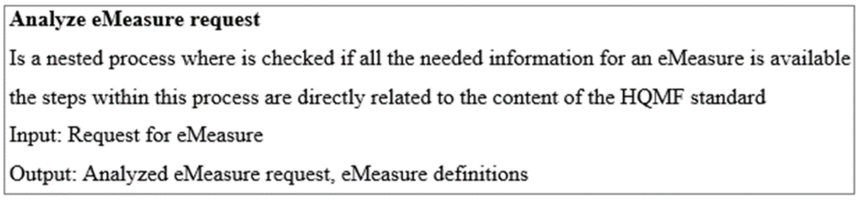

eMeasure Analyst
Start analyzing request
Give measure discription (type)
Give Population
Give Numerator
Give Denominator
Give Stratifier Criteria (grouping)
Give Measure Observation (calculation)
Finished analyzing request

Figure 46: BPMN model of eMeasure request analysis.

Van de Laar (2015) provides the following explanation for this particular BPMN model:



Analyze eMeasure request
Is a nested process where is checked if all the needed information for an eMeasure is available
the steps within this process are directly related to the content of the HQMF standard
Input: Request for eMeasure
Output: Analyzed eMeasure request, eMeasure definitions

Table 3: Analyze eMeasure request explanation (Van de Laar, 2015).

For explanation purposes, this appendix will focus on the following indicator (1d):

Operationalisatie 1d	Percentage diepe wondinfecties in geval van een totale knieprothese
Teller 1d	Aantal diepe wondinfecties tot zes weken na de operatie bij patiënten in geval van een totale knieprothese
Noemer 1d	Aantal operaties waarbij de patiënt een totale knieprothese heeft ondergaan

Table 4: Example of a single indicator of the requested knee replacement indicator set.

The next paragraphs will address the 12 FTI questions of the proposed methodology.

1. "What is the event we are addressing?"

The event that is being addressed is the Analyze eMeasure request (figure 46).

2. "Which stakeholders are involved?"

This event has one stakeholder, which is the eMeasure analyst.

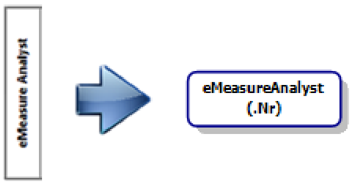


Figure 47: Stakeholder conversion, from BPMN to ORM.

3. "What are the stakeholder goals?"

According to Van de Laar (2015), the goal of the eMeasure analyst is to analyze the eMeasure request and check if all needed information in order to create an eMeasure is available.

4. "What are the stakeholder CSF's for each stakeholder goal in the context of this event?"

A Critical Success Factor (CSF) is modeled in BPMN as a diamond (gateway) and modeled to ORM as a unary predicate. The BPMN model of figure 46 does not include any gateways.

5. "Which objects are involved in the event as participants?"

Figure 48 shows the event (eMeasureRequestAnalysis) and the participants in the event. Since this process is stated as a sub-process, it has its own start event and stop event.

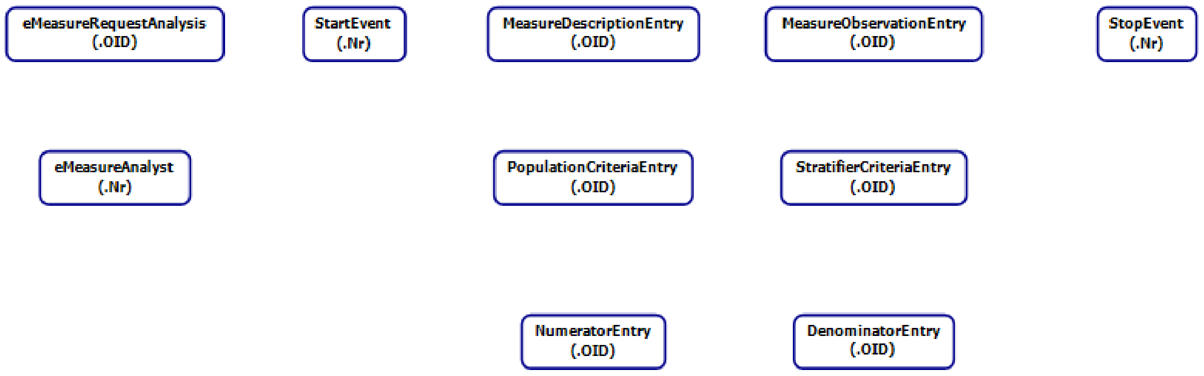

eMeasureRequestAnalysis (.OID)
StartEvent (.Nr)
MeasureDescriptionEntry (.OID)
MeasureObservationEntry (.OID)
StopEvent (.Nr)
eMeasureAnalyst (.Nr)
PopulationCriteriaEntry (.OID)
StratifierCriteriaEntry (.OID)
NumeratorEntry (.OID)
DenominatorEntry (.OID)

Figure 48: Involved objects as participants in the event.

6. "Which fact types are the event and participants engaged in?"

Let us take the objects 'eMeasureRequestAnalysis' and 'StartEvent' as an example. Because the 'eMeasureRequestAnalysis' is a nested process, it has to have its own start event (and stop event). The fact type becomes 'eMeasureRequestAnalysis' starts with StartEvent'.

Because the assumption has been made that everything in the newly designed process needs to be logged, the following fact types apply to 'eMeasureRequestanalysis' and 'StartEvent': 'eMeasureRequestAnalysis' is at Instant(.Time)' and 'StartEvent is at Instant(.Time)'. The start event and stop event are also given a number, timestamp and description. Figure 49 shows the various fact types that apply to the objects of figure 48.

These next paragraphs will explain how indicator 1d of the knee replacement example is connected to the models and diagrams of this appendix. The sub-process of Analyze eMeasure Request has its own start event, which is followed by the MeasureDescriptionEntry.

MeasureDescriptionEntry includes the type of request (e.g. dataset or indicator), the requestor and a description of the request. This description relates to the complete knee replacement indicator set of table 1; the rest of the process steps in this appendix only discuss indicator 1d.

Relatie met kwaliteit van zorg	Antibiotische profylaxe is bewezen effectief in de preventie van diepe wondinfecties bij plaatsing van een totale knieprothese. Het optreden van diepe wondinfecties wordt mede beïnvloed door aanwezige co-morbiditeit (o.a. hypertensie, diabetes, obesitas). Deze antibiotische profylaxe moet wel op het juiste ogenblik wordt gestart: 60 tot 15 minuten vóór de incisie of vóór het opwekken van bloedleegte kan als het optimale tijdsinterval worden beschouwd. Het percentage wondinfecties dat na opereren optreedt weerspiegelt de kwaliteit van de geboden profylaxe. Overigens moet er rekening mee worden gehouden dat ook patiëntenkenmerken zoals de gezondheidsstatus het optreden van wondinfecties kunnen beïnvloeden.

Table 5: Measure description.

PopulationCriteriaEntry defines the required population and, in this example, requires the following DCM's: Infection Type, Patient and Operation Type.

Operationalisatie 1d	Percentage diepe wondinfecties in geval van een totale knieprothese

Table 6: Population criteria.

NumeratorEntry defines the numerator ("Teller") and requires the following DCM's: Infection Type, Patient, Operation Type, Start Date and End Date.

Teller 1d	Aantal diepe wondinfecties tot zes weken na de operatie bij patiënten in geval van een totale knieprothese

Table 7: Numerator.

DenominatorEntry defines the denominator ("Noemer") and requires the following DCM's: Patient, Operation Type, Start Date and End Date.

Noemer 1d	Aantal operaties waarbij de patiënt een totale knieprothese heeft ondergaan

Table 8: Denominator.

StratifierCriteriaEntry defines the grouping criteria (e.g. gender, age). Table 9 applies to this entry (exclusion of patients within ASA category).

In-/exclusiecriteria	1d: Exclusie: Patiënten met ASA-klasse □ 3

Table 9: Stratifier (grouping) criteria.

MeasureObservationEntry defines the variables and calculations that are required for the requested eMeasure (e.g. numerator/denominator).

7. "Which constraints pertain to these fact types?"

Two types of constraints apply to the fact types of the BPMN model in figure 4.6; uniqueness constraints and mandatory constraints. For example, each start event is at exactly one instant in time (uniqueness) and the start event must be at an instant in time (mandatory). Figure 49 shows all constraints of the fact types.

8. "How do we identify the event and participants?"

Stakeholders are identified by .Nr and instants are identified by a timestamp, which consists of a date (day, month, and year) and time. The eMeasures are identified by an object identifier (OID), which consist of a root and a consecutive number (e.g. 2.16.840.1.113883.2.4.3.8.1000.36:001). The identification of a DCM done by using an OID and a consecutive number, like 2.16.840.1.113883.2.4.3.8.1000.36:{9146E8C3-A236-4010-A6A2-FF8F9D024EFF}. Valuesets are also identified by an OID. The attributes are identified by a (SNOMED-CT) code (e.g. 361055000). Figure 49 shows the objects with their respective identifications (.OID, .Time, .Code .Name and .Nr).

9. "What are the input events for our particular event?"
According to the explanation of Van de Laar (2015) the input is an eMeasure request.

10. "What do we have as output values for our event?"

The input for this event, according to the explanation of Van de Laar (2015) are an analyzed request for an eMeasure and eMeasure definitions (e.g. denominator, numerator, grouping criteria).

11. "What are the associated business rules for these output?"

All information products, eMeasures and DCM's need to meet the HL-7 standards. This not only a standard for the LTHN but an international (accepted) standard for all hospitals.

12. "How can we validate our model?"

The ORM diagram for the sub-process can be validated by an eMeasure analyst using a validation form. The final database blueprint is shown in figure 50 and concludes this example.

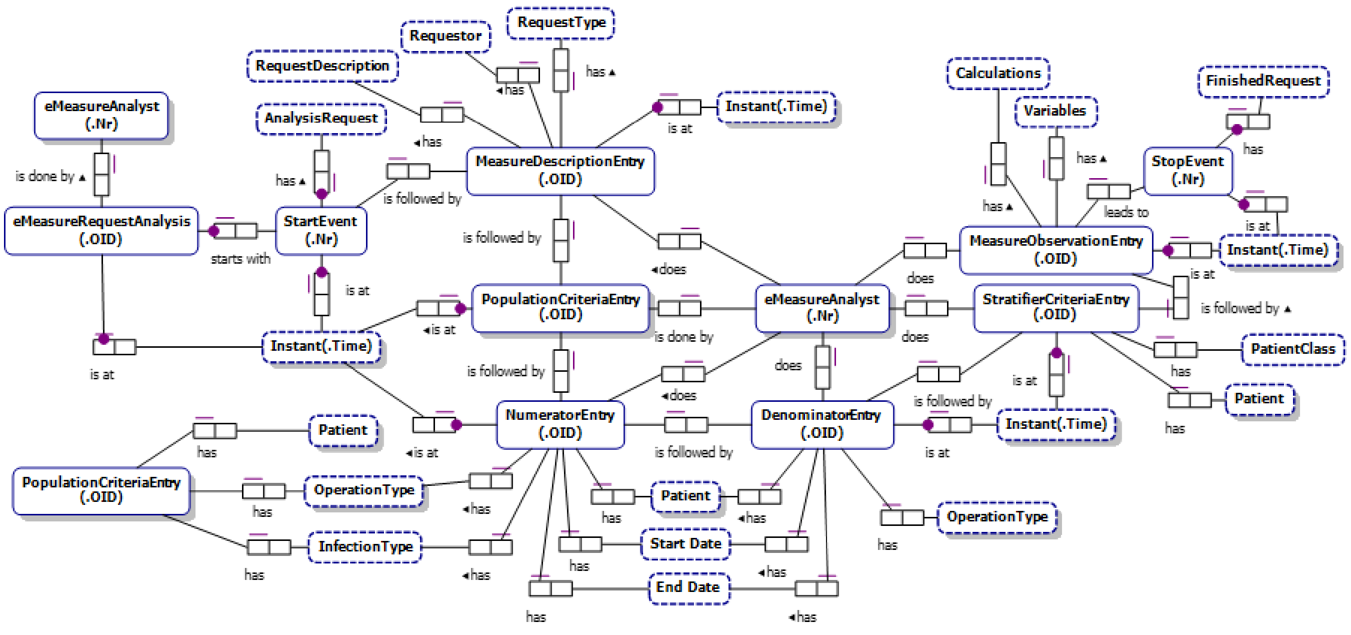


Figure 49: ORM diagram of Analyze eMeasure Request for indicator 1d.

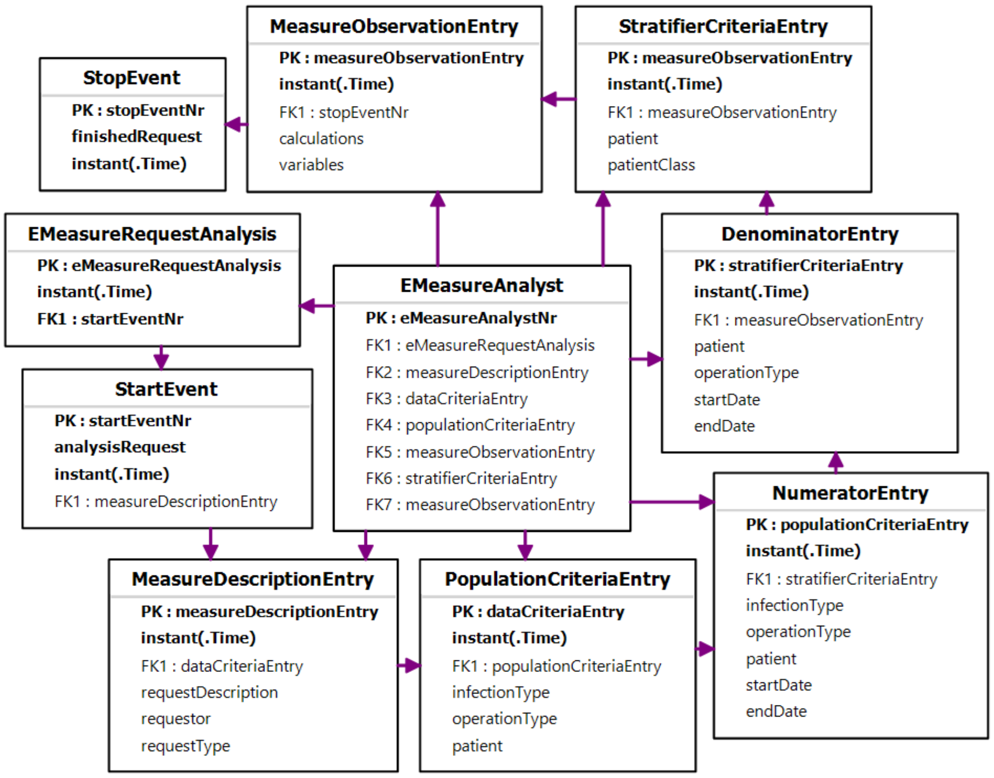


Figure 50: Database blueprint for Analyze eMeasure Request (indicator 1d).