Extensibility of product data model with XBRL in workflow management system

Erataman, K.T.

Award date:
2011

Eindhoven, August 2011

Extensibility of Product Data Model with XBRL in Workflow Management System

by

K. Teoman Erataman

BSc Industrial Engineering and Management Science

Student identity number 0728790

in partial fulfilment of the requirements for the degree of

Master of Science

In Innovation Management

Supervisors:

dr.ir. I.T.P. (Irene) Vanderfeesten, TU/e, IS

dr.ir. H.A. (Hajo) Reijers, TU/e, IS

Dave van den Ende, Deloitte Innovation

TUE. School of Industrial Engineering.

Series Master Theses Innovation Management

Subject headings: Product Based Workflow Design (PBWD), Product Data Model (PDM), workflow management systems, eXtensible Business Reporting Language (XBRL)

Management Summary

Deloitte Nederland is an independent accountancy firm that processes tax declarations and submits them to the Tax Authorities for its clients. Currently in the filing process, taxes are filed using the traditional approach (the paper-based reporting) and thus the process of preparing these declarations takes longer. With the use of SBR (Standard Business Reporting), a shift from the paper-based reports to the electronic-based formats is made to support the filing process. Then the next step would be to investigate how the workflow management systems can help to automate this process. In this sense, the previous research by Y. Sun is made to investigate the possibilities on how to use XBRL (eXtensible Business Reporting Language) in workflow management systems by using the PBWD (Product Based Workflow Design) methodology. PDM (Product Data Model) is part of this methodology for deriving process models to be supported with workflow systems and similarities between XBRL and PDM (e.g. both frameworks have hierarchical structure and data-focused instead of being process-focused) are used as a starting point. As the outcome of her research, it is concluded that the XBRL-based filing process can be supported with workflow management systems by using PDM and the feasibility of the overall approach is tested by applying only one example. The main question of this research is to generalize this approach in order to give an overview of what is missing to automatically translate XBRL into PDM (Product Data Model).

In order to support the XBRL-based reporting with PDM, we started by investigating both frameworks in terms of which requirements are missing to support the tax filing process. Then the relevant parts (constructs) in XBRL are clarified for the use of PDM. After this mapping, an analysis has been made with regards to the multiple values (instances) of a data element in the XBRL-based reports and the “Reviewing and Reading” rules of the financial filing process. As a result of this analysis, it has seen that both frameworks need conceptual solutions to support the filing process in workflow management systems and then a list of requirements has been specified for both frameworks. Afterwards conceptual solutions are proposed and implemented for each framework by taking these requirements into account. As a result of this implementation, both frameworks are extended accordingly. In the end, a sample report from the extended Dutch taxonomy which was also used as a prototype for the previous research is automatically translated into the extended PDM as a proof of concept. This translation is made completely automatic and can therefore easily be repeated for any XBRL taxonomy under certain limitations (considering single-layered XBRL structure with one presentation and one definition linkbases).

As a result of this research, we prove that the XBRL and the PDM frameworks can be mapped to each other and that the first step to use XBRL in a workflow management system is made by showing the automatic XBRL-PDM transformation. Along the way, we have made several

assumptions as only including XBRL 2.1 base specifications, using the simplified version of the XBRL structure, listing the important shortcoming of the Dutch XBRL taxonomy (e.g. dimensions and formulas) and implementing the extensions on the PDM schema. These limitations have also some implications on the general applicability of the approach. First of all, XBRL addresses the current recommendations that companies can consider for their filing process. The constructs from the current recommendations can be useful to facilitate the filing process. Secondly, the Dutch taxonomy is built up using a modular approach and the transformation file has to incorporate with the files from different layers in order to generate a PDM. In this sense, an advanced computer programming language (e.g. Java, C++) is needed to make this interface automatic in practice. Thirdly, the proposed list of requirements (e.g. status, department information of concepts) for the "Reviewing and Reading" rules shows that the current XBRL structure has also a way to go in order to support the filing process. Fourthly, in order to include all the filing rules (e.g. partial ordering relations) in the current PDM structure, a file (e.g. XSLT) next to the PDM schema will be needed to restrict and execute such rules which are now specified in the PDM schema. Furthermore, current algorithms for generating a process model from PDM are developed by taking the formal definition of PDM into account. It will not be possible to generate a complete process model by considering the extensions proposed for this research for the new definition of PDM.

As to suggest the next steps for business, investigating the real XBRL architecture for the automatic translation of PDM, solving these requirements by using the dimensions and formulas in XBRL, implementing a file in the extended PDM schema to perform the restrictions from the filing rules, adapting (or improving) algorithms for deriving process models from PDM can be seen as the initial steps. A first step to use this approach in practice is made by proposing the translation file given as a proof of concept. Extending this file by considering the real XBRL architecture will make it applicable for any Dutch XBRL taxonomy and then it can be used in practice.

To sum up, this project provides valuable insights as to see what is missing in XBRL and PDM frameworks. While incorporating with the necessary data needed for the filing process, both frameworks, as they were, need extensions. It has now become clear that future researches should not only focus on extending the PDM or the process model but also investigate the possibilities on how to extend the XBRL structure to support this filing process.

Acknowledgment

This master thesis you are holding is the result of the research I have conducted during the past eight months. Many persons helped me and supported me along the way and I would like to express my thanks to them here.

First of all, I would like to thank my supervisors, Dave van den Ende, Irene Vanderfeesten and Hajo Reijers. Especially I am thankful for Dave for giving me the opportunity to explore XBRL and helping me to clarify the terms which cause confusion in my mind during the project. I have gained a lot of experience from him in terms of having a "helicopter view" on the project. Sometimes it was hard to have this view when you involved in the technicalities of the project and you should take some distance to your work before reflecting your ideas on it. He immediately understood these situations and helped me a lot along the way. Furthermore, my project management skills have been improved while I was working with him. For these reasons, working in Deloitte was invaluable experience for me. I am also grateful for Irene Vanderfeesten for stimulating me to explore and extend my knowledge on the PDM. Specially developing conceptual solutions on PDM, she always steered my focus into the right direction. She believed in my capabilities at the time I did not believe myself. Without her support I could not have proceed this far. I would like to express my special thanks to Hajo Reijers, for his feedbacks & comments when I made my mid-term presentation.

I would also like to say my very special thanks to Paul Hulst from Deloitte. He helped me a lot on developing the transformation file from XBRL to PDM. Without his support, I could not handle the technicalities of the XSLT transformation file. Furthermore, he put valuable comments while I was exploring the XBRL and Dutch taxonomy structures. Also I would like to express my gratefulness to Yaqing Sun for her feedbacks. I took her framework as a basis and focused on the first step of the framework. She guided me a lot into the right decision by asking the right questions all the time. Whenever I had a question about XBRL, Bas Groenveld from Deloitte was always of help to clarify the things for me. Therefore I would also express my gratefulness to him.

While investigating the solutions for the financial filing rules, I had many meetings with people from Deloitte. Petro Vosselman, Twan van Gool from Audit, Sander van Wijk from Consulting, and Marc Verdonk from Innovation have all been interviewed in order to redefine the rules seen in the financial filing process.

Finally, I would like to say a big thanks to my mother and brother, especially showing their faith in me when I was down. Without you I would not have become the person I am now.

Teoman Erataman, July 2011

Contents

List of Figures

List of Tables

1. Introduction

This thesis is the outcome of the graduation project in the Master Program of Innovation Management of the Eindhoven University of Technology (TU/e). The master project is carried out in Deloitte Innovation B.V. and the Information Systems (IS) group of the TU/e. Throughout the project the possibilities have been investigated on how to extend the formal definition of PDM (Product Data Model) to support XBRL (eXtensible Business Reporting Language) information flow in workflow management system.

1.1. Deloitte Innovation

Deloitte Innovation BV enables its clients to increase their innovatory strength and instill a culture of innovation throughout the organization. Deloitte's innovation services are the culmination of a program rolled out within Deloitte Netherlands which has become a best practice for clients seeking a pragmatic approach to delivering on the promise of innovation. Together with clients innovative ways of thinking and different perspectives are explored for solving business issues. Deloitte's innovation program not only delivers tangible short-term results, but also a commitment to innovation as a long-term strategic focus that offers sustainable growth [7].

1.2. Background Business Value

A comprehensive report of a company's performance must be submitted by all public companies to the different governmental parties. These reports disclose relevant information regarding the financial position of companies and give ideas about how well they are performing. As can be understood, an audit by independent third parties, such as Deloitte, is needed to confirm that these companies are operating with the law and comprehensive about their financial statements and economic situations. This whole auditing process which starts with the client engagement and ends with the statuary filing is called the financial filing process. There are many other definitions used for describing the filing process but the definition described here is going to be used throughout the thesis.

Currently in the field of financial filing process, the financial reports are prepared by using the traditional approach (the traditional approach refers to the paper-based reporting) and the processes of producing these reports (e.g. balance sheet) take too long. By considering the current situation, there are two important reasons behind this. First, there are many manual steps taken in the filing process and a workflow management system is not used. Based on this it creates inefficient and error-prone processes. Second, the construction of financial reports is made in parallel flows, not integrated. As a result, the same data is gathered more than once. With the use of Standard Business Reporting (SBR), a first start is made to standardize this information flow [23]. The method

used to achieve this goal is to define a “common language” (or taxonomy) by using XBRL (eXtensible Business Reporting Language). Unlike the traditional paper-based reports, the taxonomy provides necessary data elements on an electronic format which makes the XBRL-based reports readable and understandable by the IT systems. In the filing process, this gives an opportunity to integrate the parallel construction of financial reports. Second step is to support the financial filing process by using a workflow management system. This makes the analysis and exchange of corporate financial information easier, more reliable and processes become faster, less costly and more efficient for data reuse [22, 23, 19]. This project is going to investigate solutions for a combination of the use of the XBRL and the Product Based Workflow Design (PBWD) methodology.

XBRL with Workflow Management System

There are many advantages of using XBRL in workflow management systems. The main benefits of this combination are to reduce human errors, to increase efficient data reuse and to improve data assurance [3, 23, 19]. Efficient data reuse means that one can use the same data element for producing more than one (financial) report. The term “data assurance” refers to checking the content of data and can be achieved by adding a control perspective to a workflow management system. In other words, implementing the financial process rules (e.g. “create” and “review” should be done by different resources, “sign off” should be done by “manager”) into a workflow management system will help to see which steps should be taken before the financial product is ready. Therefore one can keep track of which data is created, reviewed and signed off by whom.

PDM with Workflow Management System

PBWD is one of the design methodologies from the area of workflow management system and the workflow structure is defined by a PDM [19]. In other words, PDM is part of this methodology for deriving process models to be supported with workflow systems. Direct execution of PDM (instead of deriving a process model first) provides more dynamic and flexible support for the workflow process. There are a number of limitations related to the current PDM structure which needs further research in order to execute it in a workflow management system [19, 20].

XBRL with PDM

In order to integrate XBRL to the workflow management system, a PDM can be taken as a starting point. The reason why a product data model can be used to support XBRL is that the PDM and XBRL frameworks show resemblances to each other. First similarity is that the hierarchical structure of XBRL is built in tree-like structure and looks like the structure of PDM. Other important similarity is that they are both data-focused instead of being process-focused [19, 20]. These similarities are

taken as inspiration points and drive the notion of using XBRL with PDM in workflow management systems.

1.3. Problem Statement

In the financial filing process, three main reports, “Annual Report”, “Tax Return” and “Credit Risk”, can be taken as an exemplar way of working, and the current financial filing process is depicted at Figure 1. An organization has the chart of its financial accounts along with the detailed level of transactions shown for these accounts (sub-ledgers). By processing these accounts, financial reports are prepared and submitted to the responsible parties. Since each report is submitted to different party, different departments (e.g. Tax, Audit and Consultant) are involved in the process and gather information to produce the financial report. After producing and submitting the report, the governmental parties are responsible for reviewing clients’ data and checking whether the financial filing is done in a proper way. At this moment, this process imposes a significant burden on businesses and the cost to the government for the administration is high. Therefore the Dutch government is aiming to increase the efficiency and to lower the cost of government collection by introducing the “Horizontal Oversight” framework [5].

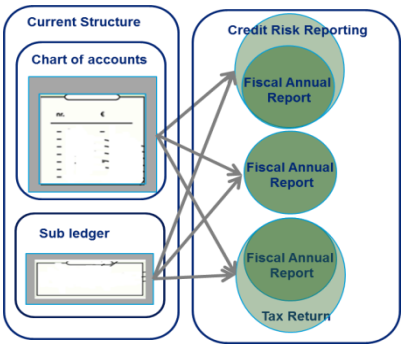


Figure 1- Current Filing Process

With the requirements from “Horizontal Oversight”, the Tax office will be in the role of surveillance and rely on the processes of intermediary organizations such Deloitte. In other words, the requirements which are now reviewed by the Tax office will be done by Deloitte and the Tax office will review Deloitte’s processes and make less review at the company level [5, 22]. The financial rules are the part of the requirements from “Horizontal Oversight” and implementing these rules into the XBRL structure will help to reduce this significant burden in several ways. In one way, the parallel flows are integrated into each other by specifying the necessary elements for each report. Then the efficiency of the filing process is improved. Another way is the reuse of data elements in XBRL so that the same data elements can be used more than one (financial) report and less rework will be done. By considering these benefits of XBRL, the next step is to use XBRL with a workflow management system which will add a control perspective on the filing process and make the whole

process more reliable and faster. The new way of working is depicted at Figure 2 (green dotted line indicates the scope of “Horizontal Oversight”.

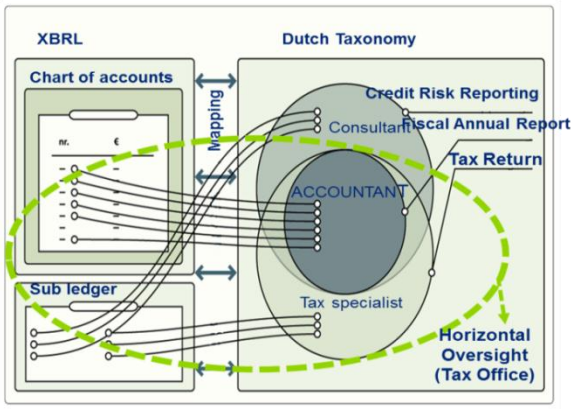


Figure 2- New Filing Process

Previous research by Sun [19] focused on possibilities to extend XBRL with a workflow management system that facilitates XBRL based financial filing processes. Two of the main problems that were addressed in her thesis are XBRL data assurance and XBRL data reuse. A framework was proposed and tested on one single report “Balance Sheet”. The result of the concluding work indicates that a PDM can be used as a tool to support XBRL in workflow management systems along with the assumptions and limitations being made. These limitations create opportunities for further work. This research therefore takes her framework as a starting point and focus on the assumptions and limitations being made. It will mainly focus on extending the formal definition of PDM to support the Dutch taxonomy.

1.4. Research Goal

The main research question, that can be derived from the problem statement and research challenges described above, can be formulated as follows:

How can the definition of a PDM be extended to support XBRL information flow in workflow management systems?

In order to answer the main research question, sub-questions are formulated as follows.

1) **Which requirements on the PDM are posed by the XBRL taxonomy?**
2) **How can the multiple instances be added to the semantics of PDM?**
3) **How can the financial process rules be added to the semantics of PDM?**
4) **How can the Dutch taxonomy be translated automatically to the PDM?**

There are several reasons why these questions are selected for the scope of this project. First of all, these questions are taken from her research [19] as limitations and further researches. Secondly, these questions investigate both frameworks (XBRL and PDM) whether they can incorporate with all data needed, if not which constructs are missing, for the translation of XBRL to PDM. In order to answer the questions, first the mapping is made between PDM and XBRL constructs to define requirements on the PDM posed by the Dutch taxonomy. Therefore relevant constructs for the Dutch taxonomy structure is clarified (e.g. linkbases, tuples). In the previous research [19], a detailed analysis of the XBRL and Dutch taxonomy were not investigated. Afterwards, the multiple instances problem of a data element is tackled. In the XBRL-based reports, generally two or more sets of values are assigned to a data element (e.g. a comparison of financial information between the beginning of the current fiscal year and the beginning of previous fiscal year in Balance Sheet). This problem refers to the multiple instances of a data element in the PDM and also one of the limitations of the PDM. In the previous research [19], the multiple instances problem was one of the limitations. A general overview of the financial filing process is given by clarifying the "as-is" and "to-be" states of the process. After clarifying the process, the filing rules are redefined by taking the previous research as a basis [19]. Then the "Reviewing and Reading" rules from this process are implemented into the PDM structure. Currently rules executed in the filing process are described on paper (manually executed) and not stored anywhere in workflow management systems. Finally, the automatic creation of the PDM from the XBRL taxonomy will show that this step can be made without any manual interaction and this will create a more reliable (and repeatable) process at the end.

1.5. Research Framework

The goal of the research by Y. Sun [19] was to propose a model for data assurance and data reuse to control data gathering in the XBRL financial process (see Figure 3). The structure of the project starts from the XBRL taxonomy and ends with the XBRL instance file. The focus of this project is shown with the red dotted lines in Figure 3. Instead of focusing the project framework as a whole, this this project focuses on the upper left part of the framework and provides solutions for the technical as well as practical challenges faced within this part. Each number in Figure 3 corresponds to the following sub-question given in Research Goal. A detail analysis has been made with the mapping from the Dutch taxonomy to the PDM in order to clarify the relevant constructs. After this mapping, it has seen that both frameworks (PDM and XBRL) have missing constructs. Then the extensions are proposed on both frameworks by considering the multiple instances question and "Reviewing and

Reading" rules. At the end, the step from XBRL to PDM is automated by taking the extended definitions of XBRL and PDM into account.

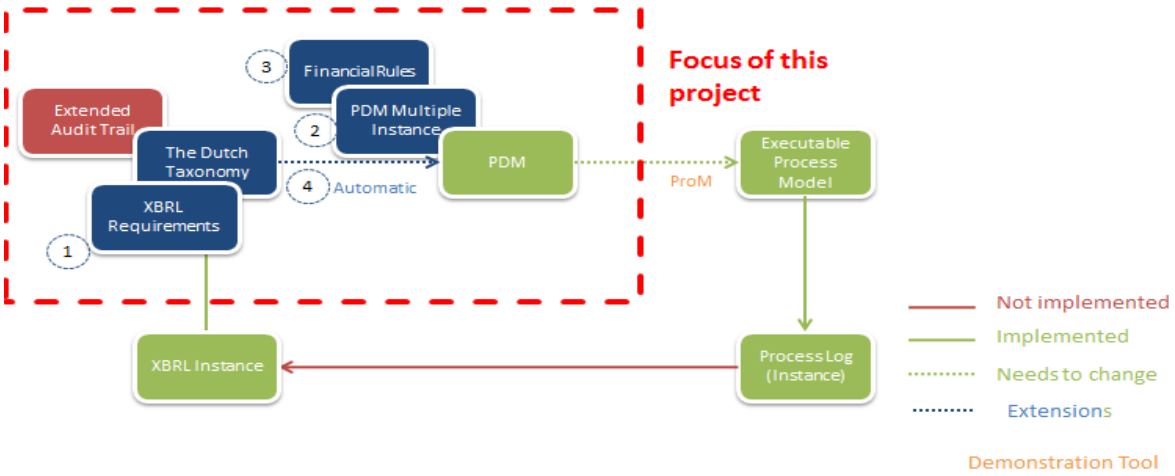


Figure 3- Research Framework

1.6. Research Approach

During the execution of this project the guidelines for the design science in Information Systems research methodology are adopted [13]. As can be understood, this is a design-oriented research and the framework in Figure 4 elucidates each phase by putting emphasis on the technical and business relevance and explains which research questions will be answered after each phase.

The project starts with the "analysis" phase in which the current situation is analyzed and the problem statements of the research questions are identified in detail. At the end of this phase, the outcome is the mapping between the Dutch taxonomy and the PDM constructs along with the important issues to be solved. These requirements are the technical input for the "analysis" phase and the second and third research questions are answered based on these requirements. The "design" phase is devoted to the conceptual solution of PDM and XBRL frameworks to support the multiple instance patterns and the "Reviewing and Reading" rules. New conceptual models (an extended version of PDM and XBRL) are introduced as an output. Afterwards, a proof of concept is given as the implementation of these solutions in the current PDM schema. Then a practical solution for the automatic translation of XBRL to PDM is made. During the project, it may be the case that extra information or a new problem definition can arise. It is important to identify new issues and clearly document them. After all, the conclusion of the project (master thesis and the model) is presented.

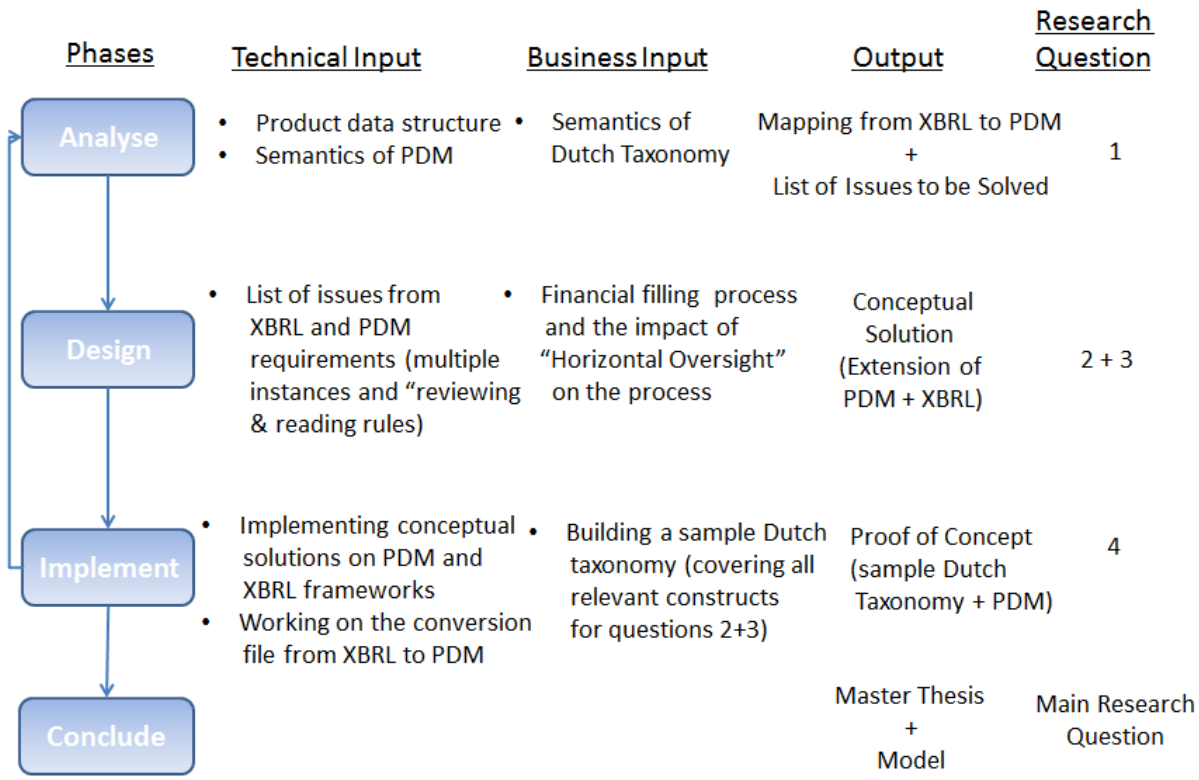


Figure 4- Research Approach

1.7. Thesis Outline

The outline of the thesis is given as follows

- Chapter 2 (Preliminaries)- explains basic mathematical notations, gives the formal definition of PDM, and provides an explanation about the XBRL and the Dutch taxonomy structures
- Chapter 3 (Analysis Phase)- gives a mapping for the multiple instance and the "Reviewing and Reading" rules questions, lists down the requirements for the missing constructs in both frameworks.
- Chapter 4 (Design Phase)- gives conceptual solutions on the formal definition of PDM and XBRL for the multiple instance and the "Reviewing and Reading" rules questions
- Chapter 5 (Implementation Phase) - takes the conceptual solutions found in the "design" phase and implement these on PDM and XBRL frameworks. Then the automatic generation of PDM file from the Dutch taxonomy is presented
- Chapter 6 (Conclusion) - gives a summary of important findings, limitations of the project and gives directions for further research.

2. Preliminaries

In the remainder of this thesis a number of concepts and theories are used, which are first introduced and formalized in this chapter. The chapter starts with the mathematical notations (definition of sets, relation and functions) which facilitate the understandability of the conceptual solutions presented in chapter 4. Second the PDM background section is given to understand the Product Based Workflow Design (PBWD) methodology. Then it is followed by the explanation of the XBRL structure and the Dutch taxonomy.

2.1. Mathematical Notations

While proposing conceptual solutions on the formal definition of PDM in the "design" chapter, the fundamental concepts are used to denote these formal specifications. This section will briefly shed a light on these concepts [15].

Sets

A set is a collection of distinct objects considered as a whole, e.g. the set of natural numbers ($\mathbb{N}$ = {0, 1, 2, 3 …}), or the set of characters in the alphabet ({a, b, c… x, y, z}). A number of notations which is used further in the project are shown as follows

- Let S_1 and S_2 be two elements. Set S is constructed by these two elements by stating S={S_1, S_2}
- $s \in S$ denotes that an element s is a member of S .
- S_1 is a subset of S, $S_1 \subseteq S$, if S_1 is contained in S
- S_1 is a proper subset of S, $S_1 \subset S$, if $S_1 \subseteq S \wedge S_1 \neq S$
- The power set of a set S is the set of all subsets of S, i.e. $\mathcal{P}(S) = \{ S_1 \mid S_1 \subseteq S \}$
- Two sets S_1 and S_2 are equal, $S_1 = S_2$, if $S_1 \subseteq S_2 \wedge S_2 \subseteq S_1$. The order in which the elements of a set appear does not matter, i.e. $\{ s_1 , s_2\} = \{ s_2 , s_1\}$
- Cartesian product, denoted by $S = S_1 \times S_2$, is defined by $S = \{(s_1 , s_2) \mid s \in S_1 \wedge s \in S_2\}$
- $\emptyset$ denotes empty set. For all sets S, it holds that $\emptyset \in S$. If a set contains more elements than just the empty set, it is said to be non-empty.

Partial Order

If S_1 and S_2 are two non-empty sets, then $R \subseteq S_1 \times S_2$ is a relation between S_1 and S_2. The set S_1 is called the "domain" of relation R and S_2 is called the "range" of relation R. And a partial order possesses some of the properties of this relation which are reflexive, antisymmetric, and transitive relation.

- R is reflexive: $\forall x \left[x \in S \Rightarrow (x , x) \Rightarrow \in R\right]$

- R is antisymmetric: $\forall x, y \left[((x, y) \in R \ \wedge (y, x) \in R) \Rightarrow x = y\right]$
- R is transitive: $\forall x, y, z \left[((x, y) \in R \ \wedge (y, z) \in R) \Rightarrow (x, z) \in R\right]$

Functions

A function is a special kind of relation in which every element of the first set is mapped to exactly one element from the second set.

Let S_1 and S_2 be two sets. A (real) function from S_1 to S_2, denoted by f: $S_1 \rightarrow S_2$ is defined as follows

- $f \subseteq S_1 \times S_2$
- $\forall s_1 \in S_1 \left[\exists s_2 \in S_2 \left[(s_1, s_2) \in f\right]\right]$
- $\forall s_1,s_2,s_3 \left[(s_1, s_2) \in f \wedge (s_1, s_3) \in f\right) \Rightarrow s_2 = s_3$

A function $S_1 \rightarrow S_2$ indicates that it assigns one elements of S_2 to each element of S_1.

Let S_1 and S_2 be two sets. A (partial) function f' from S_1 to S_2, denoted by f': $S_1 \nrightarrow S_2$ is defined as follows

- $f' \subseteq S_1 \times S_2$
- $\forall s_1,s_2,s_3 \left[(s_1, s_2) \in f' \wedge (s_1, s_3) \in f') \Rightarrow s_2 = s_3\right]$

A function $S_1 \nrightarrow S_2$ indicates that it assigns zero or one element of S_2 to each element of S_1.

2.2. PDM Background

This section elaborates on the description of the Product Data Model (PDM). First, the notion of PDM is given along with the notations and the explanation of data elements, operations, and operational attributes. Second, the formal definition of PDM is given. Finally the assumptions of PDM are briefly discussed in the last sub-section. In the remainder of the thesis, this formal definition is taken as a basis and extensions are performed on this definition.

2.2.1. Product Data Model (PDM)

The structure of the workflow product is described by the Product Data Model (PDM) and the methodology using PDM in workflow management systems is called the Product Based Workflow Design Methodology (PWBD). PBWD is a revolutionary (re)design methodology, in which the workflow product is the central concept in the design process instead of the activities in the business process [1, 17, 19, 20]. Instead of using existing processes or reference models as a starting point, PBWD starts from scratch and takes clean slate as a starting point. This is a totally different view than the evolutionary design which adopts the product-driven approach and makes the product the central concept rather than the process.

As can be seen from Figure 5, data elements are depicted as circle shapes and for each case, data elements can have different value. Operations, on the other hand, represent the actions that can be taken on data elements and are depicted by (hyper) arcs. The end product is called root element (e.g. data element A is the root element for this example). There are also several performance targets (the execution cost, execution conditions, processing time, failure probability and resource class) which can be specified at the operations in PDM.

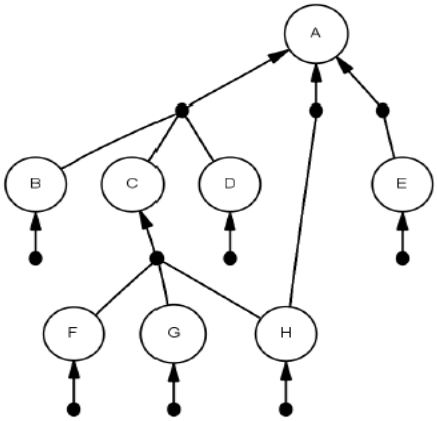


Figure 5-The representation of PDM

2.2.2. Formal Definition of PDM

In the formal definition, a PDM is represented by a 6-tuple, *(D; O; C; W; root; functions),* and functions are represented by a 5-tuple *(cost; time; cond; fprob; res)*. The PDM therefore consists of Data Elements (D), Operations (O), Conditions (C), Resources (W), root element (root), number of functions that specify properties as cost (cost), processing time (time), execution conditions (cond), the failure probability (fprob) and the authorized resource class (res) for each operation [20]. This project takes this formal definition as a starting point and offers extensions on the formal definition of PDM.

- **D:** set of data elements
- $\mathbf{O} \subseteq \mathbf{D} \times \mathcal{P}(\mathbf{D})$**:** the set of operations on the data elements. Each operation, o = (d; ds) has one output element d and a set of zero or more input elements ds
- D and O form a hyper graph H = (D;O) such that its structure graph is connected and acyclic
- **C:** The set of conditions. A condition is a logic expression and consisting of e.g. constants, variables, logical operators, and data element values
- **W:** The set of resource classes (or roles) that is available to the process. A partial order relation is defined on these resource classes
- $\mathbf{Root} \in \mathbf{D}$**:** the root element of the PDM

- **Cost :** O →R: the function that specifies the execution cost for each operation
- **Time :** O →R: the function that specifies the processing time for each operation
- **Cond:** O ⇸ C: the partial function that specifies the condition (if any) for each operation
- **fprob :** O → [0:0; 1:0]: the function that specifies the failure probability of each operation
- **res:** O → W: the function that specifies the authorized resource class for each operation.

Based on this formal specification of PDM, the xsd schema is found in Appendix A. This schema is the representation of the actual formal definition.

2.2.3. Assumptions on PDM

It is important to put an emphasis on the assumptions of PDM for understanding the semantics of the model and the applicability in this field. A set of assumptions about the semantics of PDM is excerpted in [20] and is listed below.

First assumption is that each process has exactly one end product (One Output Element per Operation). In other words there will not be two end-products produced at the end. In the same manner each operation should have one output. Otherwise it is also allowed for more end products in a process since the last operation may produce more than one data element. Second assumption is that operations are unique (uniqueness of operations). It means operations are uniquely identified by their set of input elements and the output elements and it cannot be possible for two operations having exactly the same input elements and the same output element. Third assumption is the execution of operations. Each operation can only be executed once for a specific case (e.g. a case can be considered as a financial report). The outcome of this execution is either successful or unsuccessful. If it is successful, a value for the output element will be produced. If not, no value has been produced for that data element. Fourth assumption is the availability of data elements. In other words, whenever the value of a data element is determined for a case, this value will be available for the entire duration of the case. It is not possible to produce a new value for the same data element. Fifth assumption is about the single instances of data elements. A data element can only have a single instance for a case; multiple instances of one data element in the same case are not allowed.

Listing down these limitations is important for this project for two reasons. First, one of the research questions is to tackle with the multiple instances problem in PDM (assumption 5). Therefore a clear overview of the limitations of PDM is needed. Second, all these limitations should be taken into account while proposing extensions on the formal definition of PDM.

2.3. XBRL Background

This chapter sheds a light on the XBRL structure and the Dutch taxonomy. It starts with an introduction of the XBRL structure and explains more about the XBRL specifications. After investigating the constructs in the XBRL structure, the scope will be narrowed down to the Dutch taxonomy and the relevant constructs for the PDM are elaborated.

2.3.1. XBRL General

XBRL (eXtensible Business Reporting Language) is an open standard business reporting language that facilitates the transmission of financial information by providing a common dictionary of unique identifying tags [22, 23, 24]. In other words, the XBRL framework splits business reporting information into two components: taxonomies and instances [22, 24]. The XBRL taxonomy is the design level information that abstracts the structure, and relationships between different concepts. The XBRL instance contains the instance level information, such as value, time attribute, and monetary attribute, on concepts that are defined in the XBRL taxonomy. By using the concepts defined in taxonomy, a financial report which is the instance file is created.

The current XBRL standard consists of the XBRL 2.1 base specifications and the current recommendations [22, 24]. The constructs in the taxonomy and the instance level are derived from these specifications and recommendations documents. The base specifications provide the technical definition of the constructs and set out the technical rules used in the XBRL standard. These base specifications are the basic rules and define the building-blocks of the XBRL constructs. There are also the recommended specifications which add additional constructs (e.g. the formula linkbase, dimensions, generic links, inline XBRL rendering and as such) to the base specifications [24]. In the scope of this project, the base specifications are investigated. There are two important reasons behind this notion. First, the base specifications contain all the relevant constructs to be used in generating a PDM. Second, the application of the current recommendations can differ from company to company and different constructs can be used.

2.3.1.1. XBRL Structure

This sub-section explains the constructs in the XBRL structure and shows how they are related with each other. These constructs are derived from the XBRL basic specifications. As seen from Figure 6 below, there are two levels in the XBRL structure, the taxonomy and the instance level. The taxonomy level also consists of two levels, the data and the relations level. A PDM is related with all three levels and the constructs from these levels are needed in order to generate a PDM. In Appendix B, further explanation can be found for each construct.

A. Taxonomy Level

A.1. Data Level

Item and tuple constructs are used in the data level. The term “concept” refers to these constructs and also the definition of a data element in PDM. An item represents a single data element and has several attributes (e.g. ID, period, abstract, data type). In XBRL, some data elements are dependent on each other and cannot be independently understood. For such data elements, a tuple is used for proper understanding. A tuple contains several items which are meaningful together (i.e. an “address” tuple contains “street name” and “zip code” items together). Like an item, a tuple has several attributes (e.g. ID, not abstract, nested property) as well. From this line of reasoning, items are “simple” data structures whilst tuples are “complex” data structures.

A.2. Relations Level

In the relations level, the link is created to maintain the relationships between these concepts. This level consists of linkbases which are all used for different purposes. The calculation linkbase, for instance, expresses the basic calculation relationship (e.g. summation) between concepts. The label linkbase is used for creating different labels in different languages and the reference linkbase stores the relationships between elements and the references. The presentation linkbase shows necessary concepts in a hierarchical view which are needed for the financial filing process. The main function of the definition linkbase is to capture the structure of the dimensional construct. The dimensional constructs are complex data structures and can be used as the replacement of tuples for associating concepts; therefore either one of these is chosen to be used. Since the dimensional construct is used to support the use of linkbases to define additional and structured information for the facts (values), it is decided to show dimensions in the relations level.

B. Instance Level

In the instance level, the facts (values) for the corresponding concepts defined in the taxonomy level are indicated. These facts refer to items, and tuples (or dimensions). In other words, the relationship between the taxonomy and the instance level is maintained through data elements. For each fact, context, unit, and footnote constructs are defined. The context is used for specifying the entity and period and has several attributes (e.g. ID, period type, identifier information). A unit (e.g. Euro) is defined for the numerical constructs. The footnote construct is used to associate one or more facts with the same content. It can be used as comments or notations that refer to one or more fact values [24].

The framework of the simplified XBRL structure which is derived from the base specifications is given in Figure 7. In this figure, the data level elements are depicted as circles whereas the constructs

maintaining a relation (link) between these data elements are depicted as rectangular. At the end, a financial report is created by using these constructs and their corresponding values in an instance file.

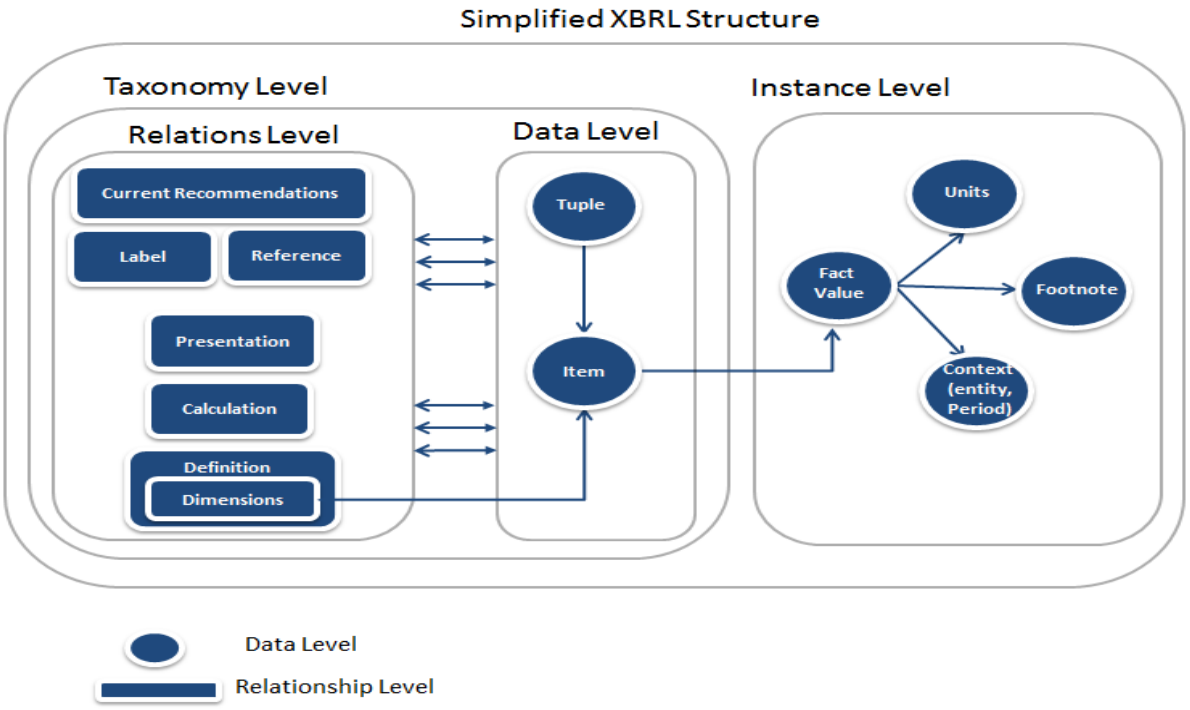


Figure 6- Simplified XBRL Structure

2.3.2. Dutch Taxonomy

Taxonomies are built by using the constructs from XBRL 2.1. base specifications and the current recommendations [24]. Different taxonomies, such as IFRS, US, GAAP, are issued by different foundations and used in XBRL for financial reporting [6, 10, 14]. The Dutch Taxonomy is derived from the Burgerlijk Wetboek (BW), the civil code of the Netherlands [6], and is using different constructs with different quantities (e.g. it contains around 10000 concepts).

2.3.2.1 Dutch Taxonomy Structure

The goal of this sub-section is going to investigate the structure of the Dutch taxonomy and pointing out the relevant constructs for the PDM. As mentioned above the taxonomy contains around 10000 concepts, and not all the concepts are put into one single (large) taxonomy, but in a set of small, easier to maintain taxonomies. In other words, the architecture of the Dutch taxonomy is built up by using a modular approach. Currently the Dutch taxonomy has a 3-layered structure (Basis, Domain, and Report) [12]. The basis layer has an architecture in which shared concepts between reports are captured; the domain layer contains unique concepts for each report and the report layer has multiple entry-points which show the financial report. In Figure 7, the simplified version of the Dutch XBRL structure is shown. This simplified version is used in this project while proposing extensions on

the Dutch XBRL structure and preparing the transformation file (XBRL-PDM) for several reasons. First, the purpose of this section is to identify the relevant constructs (e.g. linkbases, tuples, relations between concepts) for the translation of the XBRL taxonomy to the PDM and this picture serves a clear overview about it. Second, preparing the transformation file (from XBRL-PDM) by using this simplified version will proof that the first step of the research framework can be automatized. In other words, using a real taxonomy architecture will only add complexity to our transformation file since all relevant constructs are already taken into account within this file. In each level, the relevant constructs for PDM are explained.

A. Taxonomy Level

A.1. Data Level

In the data level, items and tuples are used. The characteristics of these constructs are explained in Appendix B. The Dutch taxonomy directly uses the item and tuples constructs from the base specification document and these constructs inherit the same attributes as they are defined in the base specifications.

A.2. Relations Level

As mentioned earlier, the relations between items and tuples are supported by the five linkbases. However not every linkbase is used in the Dutch taxonomy or relevant for the PDM. The function of the calculation linkbase (only summation and subtraction functions) is limited in use, therefore not used in the Dutch taxonomy. The label linkbase is used in the Dutch taxonomy to create an element with labels in different languages or for different purposes (e.g. a short label PPE means its long label property, plant and equipment). Therefore the purpose of this linkbase is not relevant for the PDM. The reference linkbase is also used in the Dutch taxonomy to make citations of some body of authoritative literature (e.g. IAS, para 68). As can be understood, this information is not relevant for the PDM. The main function of the definition linkbase is to capture the structure of the dimensional construct. As mentioned earlier, dimensions can be used as the replacement of tuples. In this sense the definition linkbase is important for the PDM. However it is not relevant for this project because the Dutch taxonomy doesn't use the dimensional constructs. The presentation linkbase shows necessary concepts in a hierarchical view which are needed for the financial filing process. This can be used as a starting point for generating a PDM. More explanation about the characteristics of linkbases can be found in Appendix B.

B. Instance Level

Units and contexts contain necessary information for PDM (for the PDM instance file). Contexts are used to specify the period information (the period in which a certain value is valid) in data elements. Units are defined for the numerical constructs. However the use of the footnote construct is not relevant for a PDM since the term is used for associating text annotations with particular facts.

As mentioned earlier, taxonomies are created by using the simplified XBRL structure (Figure 6). The framework of the simplified Dutch XBRL structure is built by taking this structure into account. In Figure 7, the data level elements are depicted as circles whereas the constructs maintaining a relation (link) between elements are depicted as rectangular. The colors are indicating whether a construct is relevant for the PDM or not.

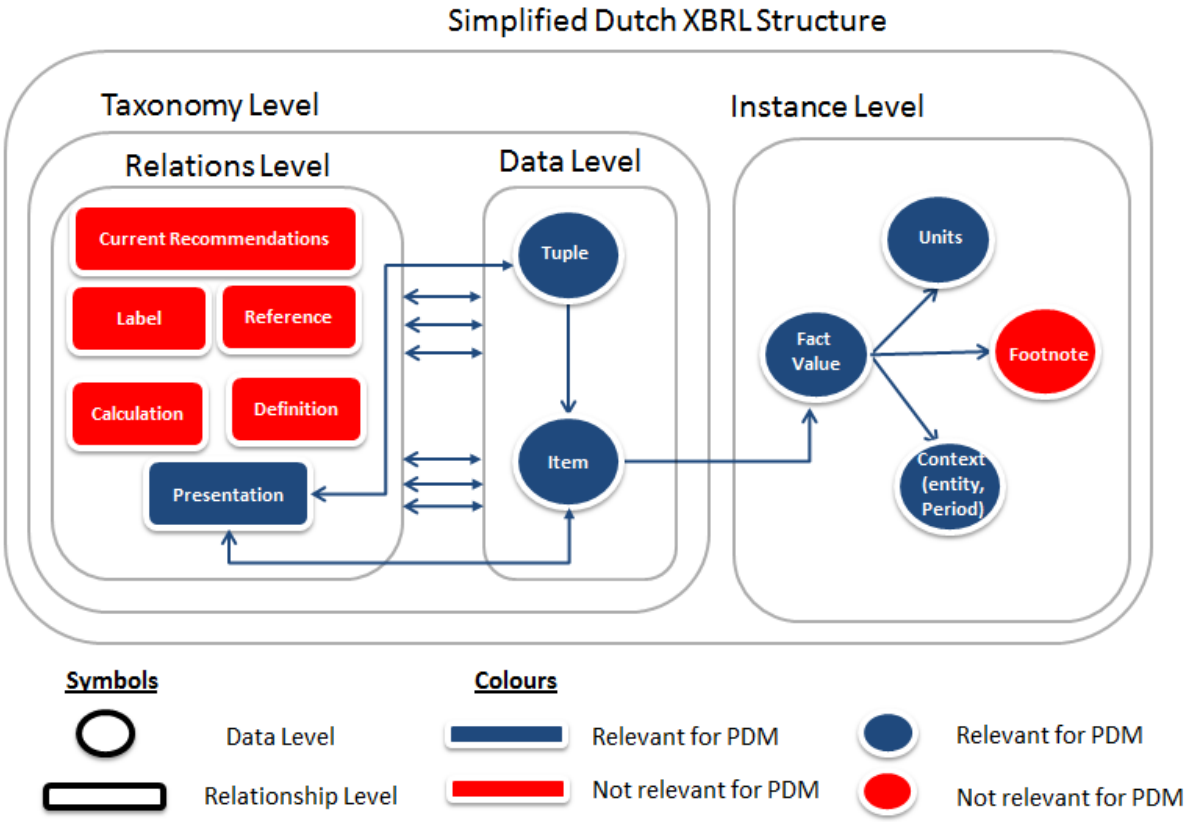


Figure 7- Simplified Dutch Taxonomy

This figure is taken as a starting point for representing the current Dutch taxonomy and extensions on XBRL are made by using this framework.

3. Analysis

This chapter gives a summary of the important “shortcomings” of the PDM and XBRL frameworks with respect to the translation of the XBRL taxonomy to the PDM structure. It is important to stress out that this project is focusing on the upper left part of the framework (see Figure 3, XBRL taxonomy to PDM) and aiming to provide solutions for challenges faced within this part. Therefore the question is whether the PDM and XBRL frameworks, as they were, can incorporate all data needed; if not, which constructs are missing in both frameworks.

First, in Section 3.1, a detailed problem statement is given for the multiple instances question and the mapping is made to match the relevant constructs seen in XBRL with the constructs in PDM. Then the multiple instances patterns seen in process models are taken as a starting point on how to classify the patterns in XBRL. Second, in Section 3.2, a detailed problem statement is given for the financial filing process by explaining the types of rules and “as-is”, “to-be” states of the process models. Afterwards, the mapping is made between the XBRL and PDM constructs for the “Reviewing and Reading” rules. Both questions are carried to the “design” phase and possible extensions are proposed on the formal definition of PDM and XBRL.

3.1. Multiple Instances Problem Statement

In the XBRL-based reports, a data element is created in the data level and generally two or more sets of values are assigned to that data element in the instance level [19, 20]. For example, in order to create a Balance Sheet, a comparison of the financial information between the beginning of the current fiscal year and the beginning of previous fiscal year(s) has to be provided. These set of values are shown in the XBRL instance level by using the context construct. Multiple values of a data element are generally observed in the XBRL-based reports and this problem refers to the multiple instances (MI) of a data element in the PDM. Therefore an extension is needed on the PDM framework.

In the PDM background chapter, the single instance of data elements is mentioned as one of the limitations of PDM. So the multiple instances of the same data element are not allowed yet and as can be understood it may be essential to model these patterns. Similar problem in the area of process modeling is introduced and described in the workflow patterns [21]. From this line of reasoning, first the types of MI patterns in XBRL are described along with the mapping between the relevant XBRL and PDM constructs, and then the MI patterns in process models are taken as a starting point on how to classify such patterns in XBRL.

3.1.1. Mapping between XBRL and PDM constructs for Multiple Instance

In order to understand the types of MI patterns in the Dutch taxonomy, the XBRL structure and the reporting standards specified for the XBRL financial filing have to be investigated first.

In the XBRL structure, two types of Multiple Instance patterns are observed. First type is the time dimension of a concept (an item or a tuple) in XBRL. In other words, the "context" attribute is defined for each fact (value) in order to specify this time dimension. So the maximum number of facts per concept equals to the number of contexts with the same period type (i.e. if there are 3 contexts with the period type instant, then there can be only 3 fact values for a concept with the period type instant). Second type is the tuple representation. An item under the same tuple can have more than one value for the same context (i.e. a "provision" tuple contains several items depending on the situation as "value of the obligation", "risk and uncertainties", "description of the provision", "changes in provision" and for each claim different "provision" tuple is created and an item under this tuple can have several values which are specified by using the context construct). By considering this information, the mapping is made between the XBRL and PDM constructs for the multiple instances problem.

Table 1- Mapping constructs for Multiple Instance

	XBRL Constructs	PDM Constructs	Description	Framework	Extensibility
Multiple Instance Question	A context characteristic in XBRL	Multiple Instances of data elements	XBRL allows you to define as many context as possible for a concept (i.e. several values of the same data element but with different moment of validity)	**PDM**	**Extension is needed in the PDM framework**
	Tuple representation	Multiple Instances of data elements	The same tuple can be created and used more than once in the instance level. Accordingly an item under same tuple can have more than one value for the same context	**PDM**	**Extension is needed in the PDM framework**

To conclude the multiple instance patterns seen in the Dutch taxonomy structure, it is technically possible for an item (or a tuple) to have as many instances as possible.

The reporting standard is specified by governmental bodies for the XBRL-based financial filing and the compliance to this standard is mandatory for businesses. In this sense, the FRIS (Financial Reporting Instance Standards) document is provided by most of the regulators. This document explains the reporting standards which companies must comply with while reporting their financial statements to the regulators. The NL-FRIS document is prepared to be used for the Dutch Taxonomy of XBRL based reports [18]. In this document, the content of the report for the financial filing is explained. For instance, in the KvK (Chamber of Commerce) reports, there have to be two values for all monetary items and there is only one value for textual items. For BD (Belastingdient), one value for each item is allowed for the VAT report and two values for the Income Tax report. For other reports, most of the time there are two contexts defined per item; however it might be the case that there are more than two contexts defined for a certain type of report (e.g. if you have a value for each month, then you have 12 different values). In other words, most of the times in practice two values are assigned to a data element, but theoretically it may be the case that there are more values possible (with using dimensions, more instances can be observed). Therefore the research presented in this thesis aims to provide a general solution to the multiple instances problem.

To sum up, technically XBRL allows you to create as many instances as possible; however there are certain requirements from the NL-FRIS document which all companies are obliged to when reporting their financial statements to the regulators. From this line of reasoning, there are two patterns found in the XBRL structure. First one is the fixed number of instances (provided by the NL-FRIS document); the second one is the multiple occurrences of the tuple representation. Next section describes how these patterns can be classified in terms of the MI patterns seen in the process models.

3.1.2. Multiple Instance Patterns in process models

Based on the different workflow systems, it has seen that some patterns, especially involving multiple instances, are not easy to handle [2]. In the workflow patterns [21] which was established with the aim of delineating the fundamental requirements in the business process modeling, different types of the MI problem are addressed. Mainly four patterns are observed as Multiple Instances without synchronization, Multiple Instances with a Priori Design Time Knowledge, Multiple Instances with a Priori Runtime Knowledge, Multiple Instances without a Priori Runtime Knowledge (more explanation can be found in [21]).

The idea of these four patterns can be taken as a starting point on how to solve multiple instances question in XBRL. As concluded in the previous sub-section, there are two patterns observed in the

Dutch taxonomy which can be matched with the patterns defined in the workflow patterns [21]. The following table gives a proper explanation for each pattern.

Table 2- MI Patterns in XBRL

Pattern	Definition
Multiple Instances With a Priori Design Time Knowledge (Fixed number of instances)	A fixed number of instances with a priori design time knowledge are observed in the XBRL instance file. Most regulators provide a FRIS document in which the data they expect from the XBRL document is described. (i.e. for KvK reports there are two values for all monetary items) For other reports most of the time there are two contexts defined per concept. For example, the Balance Sheet has two instances which are the beginning of the current fiscal year and the beginning of previous fiscal year. For certain types of reports, more than two contexts can be defined. However the number of contexts in each report can be assumed beforehand (i.e. before the process is executed at the design time of the PDM).
Multiple Instances With a Priori Runtime Knowledge (Multiple occurrences of the complex data structures, tuples)	An item under a tuple may have 0 to many multiple instances and there is no way of knowing in advance how many occurrences there will be. Like the example of a “provision” tuple above, the same tuple can be created and used more than once in instance level. Accordingly an item under the same tuple can have more than one value for the same context. However it is assumed that once a company starts to fill the instance document they will know how many tuples they need.

In the “design” chapter, a solution is presented to deal with these two types of multiple instances in the PDM.

3.2. Financial Filing Rules Problem Statement

The financial filing process starts with gathering financial information from a client and ends with the statuary filing. As mentioned in the introduction chapter, XBRL is used in this process for preparing financial reports of clients. Right now a workflow management system is not used and the preparation of the financial reports takes longer. In order to use the XBRL with the PDM, it is an important step to take the financial filing rules into account.

Before classifying the types of the filing rules, a clear overview of the financial filing process is needed. In order to have this overview, meetings are planned with Audit and Consultancy departments to clarify how the financial report is constructed by using Deloitte’s processes, which steps are taken, which types of rules are executed during the process. As a result of this

walkthrough, the current ("as-is") and the future ("to-be") states of the financial filing processes are depicted and the types of rules applied in this process are clarified. Furthermore a detailed investigation is made by giving concrete examples from each type. In order to see differences from the previous research by Y. Sun [19], Table 3 is given below.

Table 3- Overview of Filing Rules

Types of Filing Rules	Previous Research	This Research
Content Related Rules	Content-related rules specify relationships between taxonomy concepts. This rule can be defined using Calculation Linkbases or Formula Linkbases in XBRL taxonomy.	Content-related rules are applied to the XBRL-based reports to make them ready for the statuary filing and have two types. First type is to specify relationships between concepts as the part of the Dutch taxonomy structure specified in the NL-FRIS document. Second type is to express the necessary concepts for the "structure" and "size of the cooperation" for the financial filing.
Work-program Rules	Work-program and Reviewing and Reading rules are classified as one type "Process-related" rules. Process-related rules specify relationships between process-related information, such as data creating time, task originator and etc., of taxonomy concepts.	Work-program rules explain the steps on how the financial values are generated for concepts. In other words, the notes which auditors are taking and working on throughout the filing process are called "Work-program" rules.
Reviewing and Reading Rules	Work-program and Reviewing and Reading rules are classified as one type "Process-related" rules. The previous research proposed an extension of sub-processes in the PDM and gave guidelines and suggestion on how to extend the XBRL structure with workflow information.	After executing the "Work-program" rules and making the concept version of a report ready, the "Reviewing and Reading" phase takes place where a reviewer checks the notes and gives feedback afterwards. This research specifies a list of requirements both for XBRL and PDM. For PDM, conceptual solution is proposed by taking her sub-process extension as a starting point [19]. For XBRL, a conceptual solution is proposed by taking this list of requirements into account.

This research has made a detailed analysis on the "Reviewing and Reading" rules and as a result, a list of requirements has been specified both for XBRL and PDM. Afterwards, conceptual solutions are proposed for both frameworks by taking this list of requirements into account.

3.2.1. Different Types of Filing Rules

Content Rules

"Content-related" rules are applied to the XBRL-based reports to make them ready for the statuary filing and have two classes. First, these rules are used to specify the relations between elements in the Dutch taxonomy.

- An item has a period type (either instant or durational type) attribute.
- Total Liabilities & Equity is equal to the summation of Liabilities and Equity.
- An item has a balance (either credit or debit) attribute

These rules which express relationships between the constructs as the part the Dutch taxonomy structure are specified in the NL-FRIS document [18]. In Appendix D, a mapping is made between XBRL and PDM constructs by considering these rules. The other class of the "content-related" rules is used to specify the necessary concepts for the structure and size of the cooperation for the financial filing.

- Small companies do not have to report the profit & loss statement to the Chamber of Commerce; however larger companies do (the size of the company).
- If a company X does not have "goodwill" concepts, then these do not report (the structure of the company).

In order to implement these rules into the workflow management system, a research should be made in XBRL on how to filter out unnecessary concepts from the report. Afterwards, the constructs between XBRL and PDM can be matched. From this line of reasoning, the "content-related" rules are not dealt within the scope of this project.

Work-program Rules

Work-program rules explain the steps on how the financial values are generated for concepts. In other words, the notes which auditors are taking and working on throughout the filing process are called "Work-program" rules. Currently these notes (rules) are stored in a document management system called AS2 and manually executed. Examples below are given for this type.

- Compare the financial statements with the previous period; analyze the absolute and relative differences regarding balance sheet items
- Explain major changes in the financing structure
- Note that the principal, collateral, repayment and interest terms correctly in the financial statements are explained

These notes are the things which are checked by auditors and largely depended on auditor's experience and decisions. As can be understood, different actions are taken while executing these rules and they also show difference from client to client (instance to instance). In order to use them in a workflow management system, first additional research is needed. As a result of this research, these notes can be decomposed into the specific set of rules which makes easy to interpret and implement in a workflow management system. The rules under this type are mainly expressing the relations between the elements; therefore these rules can be implemented in the XBRL structure. From this line of reasoning, they are not dealt within this project.

Reviewing and Reading Rules

After executing the "Work-program" rules and making the concept version of a report ready, the "Reviewing and Reading" phase takes place where a reviewer checks the notes and gives feedback afterwards. Currently these rules are stored in a document management system called AS2, and are manually executed. Here below some examples from this type are given. In Section 3.2.3, first a detailed analysis of the missing constructs in the "Reviewing and Reading" rules are clarified, and then this analysis will be used as an input for the "design" phase.

- "Create", "Review" and "Sign-off" should be done by different resources.
- A data element should not be "Signed-off" if it is not "Reviewed"
- "Creator", "Reviewer" and "Sign-off" have all specified job-grades (e.g. "Reviewer" has a 9 minimal "job-grade" and a "Reviewer" job-grade should be higher than a "Creator")
- Depending on the structure of a client, different levels of sub-processes are taken (e.g. if a client has "larger than normal risk" level then a detailed sub-process will be taken into account).

Implementing these rules into a workflow management system will play an important role for the financial filing process for several reasons. First, the rules in this type do not show difference from report to report (instance to instance). Second, implementing these rules into the XBRL and PDM frameworks add a control perspective to the process which will improve the efficiency and the quality.

3.2.2. As-Is and To-be Process Models

As we mentioned in the introduction chapter, "Annual Report", "Tax Income", and "Credit Risk Report" are three main reports which companies file during their filing process. By considering these reports, a walkthrough has been made and the "as-is" and "to-be" states are depicted below. Both

process models are drawn in WoPeD (Workflow Petri Net Designer) which is an extended class of Petri Nets and initially introduced by Wil van der Aalst [3].

In some cases, submitting financial statements by using XBRL-based reporting is a mandate [22]. Currently companies who are obliged to this reporting requirement must file their reports in the XBRL-based format. However others can still use the traditional approach (paper-based reporting). The intervention of XBRL in the process of financial filing is now involved while preparing the reports and no workflow management system is used as mentioned in the introduction chapter. Furthermore the construction of reports is occurred in parallel structure and the data information flow is not integrated yet. In future, it is planning to integrate XBRL from the start of the filing process and a workflow management system is utilized. In this way, there is one pool of resources (data elements) and all reports are sharing the same resource. Since reports do share considerable amount of common elements, the main advantage of using XBRL is to reduce rework and make the process more efficient. Then the data classification will be made between departments who are involved in the filing process and as a result the shared elements will only be consumed once and reused from different departments afterwards. Another advantage of using XBRL earlier in the filing process is that companies can have a good reporting system clearly showing the outcomes of their processes and the rules from “Horizontal Oversight” will be implemented easier. By using XBRL early in the process, it would be easier to implement the “Horizontal Oversight” framework and thus improve efficient data reuse. As a result of this, it can be assumed that the throughput time of the process is reduced in the “to-be” state.

As-is State

Currently the filing process starts with gathering the financial data and preparing the specifications for each report. Then the work-program (“Werkprogramma”) is planned. This program guides the process along the way and specifies the approach taken for filing process (i.e. more “work-program” and “Reviewing and Reading” rules take place for organizations which are higher than normal risk level). After completing the “Work-program”, and “Reviewing and Reading” rules, concept version of the report is prepared and shared with a client for the review purpose and then finalized afterwards. Companies who are not required to file in XBRL use the traditional filing (paper-based). If it is necessary, the reporting program (DART) will generate the XBRL instance automatically. Then the “content-related” rules which specify the relations between the concepts in the Dutch taxonomy are checked. After checking these rules, the “verification” task takes place to make sure that the paper-based report and the XBRL instance are showing the same elements and values. After the “verification” step, the report is automatically imported to the portal for a client final sign-off. With

his final review, the XBRL based report is sent to the regulators (In Figure 8 the “as-is” process model can be seen).

To-be State

In the “to-be” state, where XBRL is involved in the process from the start, the process starts with importing the financial data (in XBRL format) from a client. After that the data classification (Classify Data task in Figure 9) has been made in order to specify which departments are responsible for which elements. Therefore the work-program will be now less extensive. By using workflow management system, each department can keep track of the status information of elements (e.g. “created”, “reviewed” or “signed-off”) which are relevant for them. Therefore the parallel flows in XBRL are integrated and depended on each other. Furthermore the “Reviewing and Reading” rules will consume less time since the departments immediately understand which elements are “created-reviewed-signed” by which department. Then the draft version is shared with a client for review purposes and finalized afterwards. Now with using workflow management system, data elements are stored in electronic-based formats and it is easier to generate an instance document (financial report) since the status information of elements has been clearly tracked throughout the filing process. The reporting program (DART) will generate this XBRL instance and the “content-related” rules are applied. Then the “verification” process takes place for the final review. After this step, the report is automatically imported to the portal for a client sign-off. With his final review, the XBRL based report is sent to the regulators (In Figure 9 the “to-be” process model can be seen).

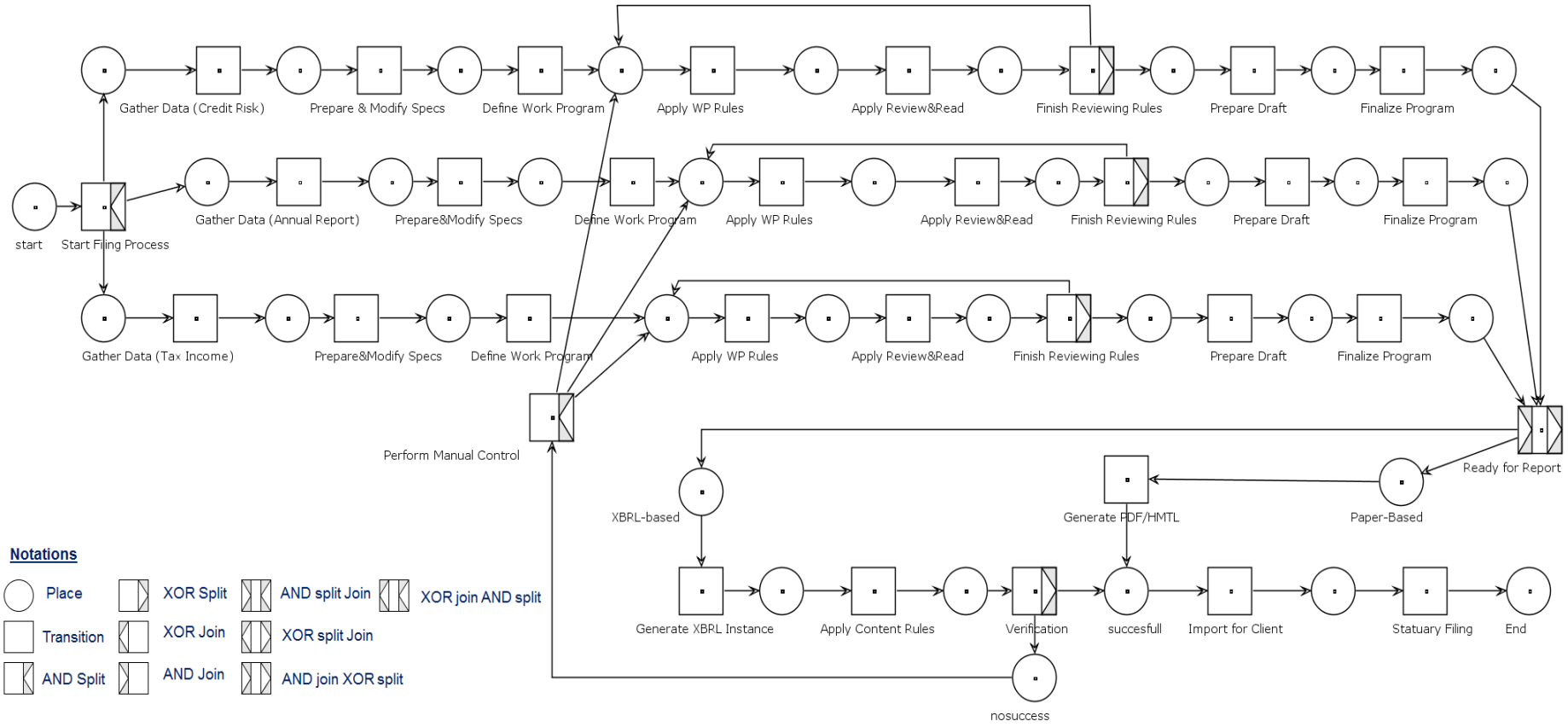


Figure 8- As-is Process Model

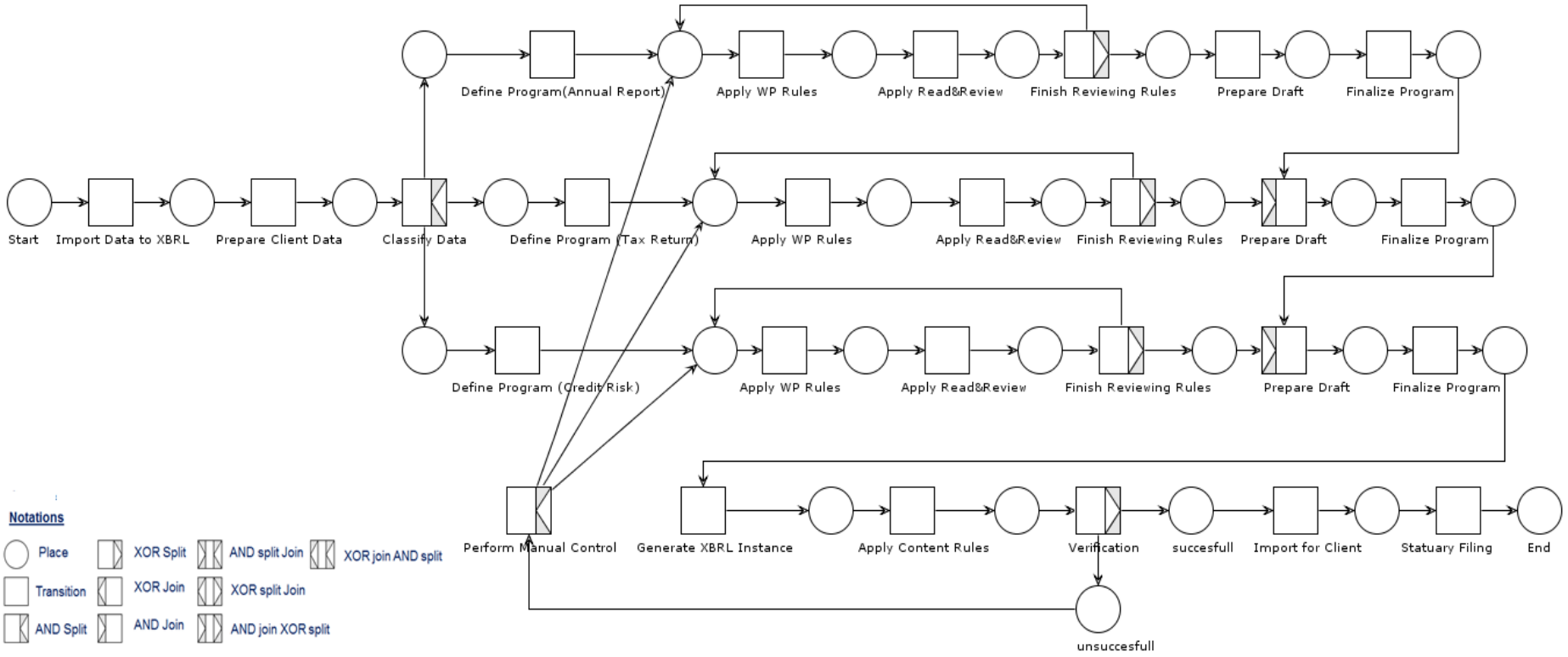


Figure 9- To-be Process Model

3.2.3. Mapping between XBRL and PDM constructs for "Reviewing & Reading" Rules

In order to clarify which constructs are missing for the "Reviewing and Reading" rules, a walkthrough has been made on how the annual and the tax declaration reports are constructed by using Deloitte's processes [9]. As a result of this walkthrough, first the important constructs for the "Reviewing and Reading" rules are clarified. Then, the actual "Reviewing and Reading" rules are specified as follows. While analyzing these constructs, the "to-be" state of the process model is taken into account. Along the way, the Accountancy Approach [8] which provides guidance on the implementation of new clients and projects for Deloitte Nederland has also been checked. Currently the constructs explained below are missing in both frameworks (XBRL and PDM).

Sub-Processes

In the research [19], it is stated that there is one big sub process which is following the sequence of "Create", "Review" and "Sign-off". This research takes this information as a basis and a detailed analysis has been made on the sub-process for the "Reviewing and Reading" rules. While executing the filing process, different levels of sub-processes are performed depending on the client situation. These levels can be seen as the milestones of the filing process which make a change on the status information of a data element. In the following paragraphs, main sub-processes are explained by showing how the financial filing process of Deloitte is implemented on its clients.

The process starts with the "Create" step indicating that the concept is created. After creating a concept, the "Review" sub-process is followed meaning that the created concept is checked. Then the "Read by 1st" sub-process takes in place to assure the correctness of concepts. When the 1st reader finishes, the "Statement" sub-process is made with the client. In other words, a discussion with the client is performed and as a result of this discussion "Statement" is made. Depending on the need of an "independent" reader, another sub-process "Read by 2nd" reader is followed. After agreeing upon the "Statement", the report is "Sign-off" by the responsible managers (assignment manager) from Deloitte and client. When the report is finalized, the "Archived" sub-process is performed to store the financial report. By considering the Accountancy Approach [8], three approaches are developed to provide guidance on the implementation of these rules. As can be understood, the sequence of these sub-processes is different for each approach.

Standard Level: {Create, Review, Made-Statement, Sign-off-1, Sign-off-2, Archive}

Comprehensive Level: {Create, Review, Read by 1st, Made-Statement, Sign-off-1, Sign-off-2, Archive}

Larger than normal risk Level: {Create, Review, Read by 1st, Made-Statement, Read by 2nd, Sign-off-1, Sign-off-2, Archive}

Sign-off 1 and 2 refer to authorized signatures from Deloitte and client. As mentioned earlier, these sub-processes are taken as the milestones which change the status information of an element. In real case, there are other small sub-processes such as “Close & Backup” dossier and “Shipping” report before being “Archive” or “Making Draft for 1st reader” sub-process after “Review”.

Status Information

By taking the names of activities (sub-processes) above, the status information of a data element can be changed as “Created”, “Reviewed”, “Read by 1st”, “Statement Made”, “Read by 2nd”, “Signed-off-1”, “Signed-off-2”, and “Archived”.

Department Information

As can be seen from the “to-be” state of the model, mainly three departments are involved in the process as “Audit”, “Tax” and “Consultancy”. In the XBRL structure, the shared and unique concepts are mapped and clarified for each report. While classifying the data in the “to-be” state, specifying “department” information will clarify the data ownership.

Other than these three, there are also different departments involved in the filing process as “Client Engagement” for gathering the basic elements (e.g. statuary address, legal forms).

Role Information

Like the sub-processes, mainly seven roles are defined as “Creator”, “Reviewer”, “1st Reader”, “2nd Reader”, “Signed-off-Deloitte”, “Signed-off-Client” and “Archival”. These roles can also be used for the other small sub-process (e.g. “Making Draft for 1st reader” is done by “1st Reader”, “Close & Backup” and “Shipping” are done by “Archival”).

Job Grades Information

By considering each role, a specific job-grade is assigned to that role in order to do its function. Job grades are defined from 0 to 14, meaning that 14 is the highest level. Normally “Creator” has no minimal “job-grade” function, “1st Reader” has a 10 for minimal job-grade, and “2nd Reader” or “Signed-off” has a 14 for minimal “job-grade”.

Reviewing and Reading Rules (Relations Information)

As can be understood, there are relations between these constructs and these relations are actually representing the “Reviewing and Reading” rules in practice. Basically three types of relations are observed. First one is the order relation in each construct (e.g. the status information of a data element cannot be “Signed-off” before it is not “Reviewed”, the sub-process “Review” cannot be

executed before the sub-process “Create”). Second type is the relation between constructs (e.g. “Signed-off” sub-process should be done by role “2nd Reader” or “Signed-off”, the status information of an element should be “Reviewed” after “Review” sub-process is performed, the job-grade of the “Reviewer” should be higher than the job-grade of the “Creator”). Third one is the resource relation meaning that “Creator”, “Reviewer” or “Sign-off” activities should be different (or same) resources (the four-eye principle).

To sum up, in the “design” chapter the conceptual solution is proposed for extending the Dutch taxonomy and the PDM with the information needed to include the “Reviewing and Reading” rules. For the PDM framework, the information specified above is used to extend the formal definition. However, for extending the XBRL framework, first the analysis of these requirements is made then the XBRL is extended with the necessary construct(s). In the following section, this analysis can be found.

3.2.4. Analysis of XBRL Requirements

In the research [19], it is concluded that the XBRL taxonomy needs to be extended in a way that the workflow information, especially financial filing rules, can be designed in the taxonomy file. Then several techniques were proposed to serve guidelines and suggestions as “Creating Workflow Linkbase”, “Using XBRL Instance Footnotes”, and “Using XML Signature and XML Encryption”. This research takes this notion as a basis and a conceptual solution on the XBRL taxonomy is proposed by considering the “Reviewing and Reading” rules. First, this section analyses the missing constructs seen in the XBRL framework and elaborates these shortcomings. Then in the “design” chapter, a conceptual solution is proposed by taking these rules into account.

In the current XBRL taxonomy, the status information of concepts is not stored. Considering the “to-be” state of the filing process, having this information can be useful to monitor the current status of the shared concepts. In other words, departments will immediately understand which common concepts are reviewed by which department and which are ready to be used. In the same manner, specifying the department information for concepts will clarify the data ownership. As mentioned earlier, shared & unique concepts are classified in the XBRL structure and assigning the department for common concepts would increase the efficiency of the process in terms of data reusability. The relations between the constructs (e.g. status information cannot be “Reviewed”, if it is not “Created”) will also add a control perspective to the XBRL structure.

The conceptual solution with regards to these shortcomings is shown in the “design” chapter and the current XBRL structure is extended accordingly.

4. Design

In the previous chapter, the "analysis" part of the research questions is completed by doing the mapping between XBRL and PDM constructs. In this chapter, each sub-section refers to the conceptual solution of the research questions. First section is devoted to the multiple instance patterns in XBRL. Based on the patterns identified in Section 3.1.2, the formal definition of PDM is extended accordingly. Second part sheds a light on the financial filing process and the rules derived from this process. Then the conceptual solution is proposed on the "Reviewing and Reading" rules by extending the formal definition of PDM and the current structure of XBRL.

4.1. Multiple Instances

As described in the "analysis" phase (chapter 3), the XBRL structure allows you to define as many instances as possible; however the reporting standards put certain requirements to the number of instances to be submitted in the report (pattern 1, Multiple Instances with a Priori Design Time Knowledge). For complex data structure (tuples), they can be created and used more than once. In other words, an element under the same tuple can have more than one instance for the same context (pattern 2, Multiple Instances with a Priori Runtime Knowledge).

From this line of reasoning, first an abstract point of view is given on how to approach the MI problem and then the conceptual solution is proposed. Finally the solution is discussed by considering the XBRL structure and other multiple instance patterns seen in practice.

4.1.1. Conceptual Solution from an abstract point of view

Currently in the formal definition of PDM, multiple instances of one data element are not allowed for the same case. Like the XBRL structure, similar notion of "context" attribute is needed in PDM to define the sets of values which are assigned to one data element. It is clear that the MI problem is related with the data elements and an extension is needed on the definition of data element. As can be understood, sets of values in XBRL can be converted into a set of instance range [1, ∞] in PDM. Then this range can be taken as a starting point to define the lower and upper bounds of each data element. In this example, the lower bound "1" indicates the single instance of a data element; whereas the upper bound is set free (it is assumed that there is at least one instance for each data element). Therefore the designer can have a freedom to define the maximum occurrences of data elements and the exact number can only be known at the design time. By taking this information into account, next section proposes a conceptual solution on how to deal with the MI patterns in the PDM.

4.1.2. Conceptual Solution

Considering the current formal definition given in the PDM background (chapter 2), the “minVal” and “maxVal” attributes can be used to specify this range. Therefore upper and lower bounds can be set for each element. If a value “0” is assigned for the “minVal” attribute, it means that this data element is optional (i.e. it will have 0 instances at runtime). However, in the PDM, a data element has one instance for the same case. This logic is also same for the XBRL and it can be assumed that an item shown in the presentation linkbase and applicable for the financial filing has at least one value in the instance file. Therefore the “minVal” attribute can be set “1” which indicates the single instance of a data element. In the “minVal” is equal to “maxVal” condition, it refers to pattern 1 (Multiple Instances with a Priori Design Time Knowledge). In other words if the “minVal” is greater than a value “1”, it refers to a fixed number of instances with a priori design time knowledge. If the “maxVal” is greater than the “minVal” condition, this refers to pattern 2 (Multiple Instances without a Priori Run Time Knowledge). From this line of reasoning, *range* construct is developed and put in the formal definition.

- Range function
  - range: D $\rightarrow \mathbb{N}^{+}\mathrm{x}\ \mathbb{N}^{+}$ and denoted as
    - range $\subseteq \mathbb{N}^{+}\mathrm{x}\ \mathbb{N}^{+}$
    - $\forall_{x,y}\left[(x,y) \in \mathrm{N}^{+}) \Lambda\ (x \leq y)\right]$

The PDM, as defined in Section 2.2.2, is now represented by a 6-tuple, *(D; O; C; W; root; functions),* and functions are represented by a 6-tuple *(cost; time; cond; fprob; res, range).* Range function on data elements is used to specify the minimum occurrences and the maximum occurrences of data elements.

4.1.3. Reflection

At the end of this sub-section, the formal definition is extended with the *range* construct. Besides this solution, other extension possibilities can be found in Appendix C. As can be understood, this solution has an effect on the next steps of the project framework. Now the automatic generation of a process model cannot completely work anymore because the current way of deriving a process model from a PDM does not include the *range* construct. For this problem, a new algorithm should be developed to translate this conceptual solution into a process model. Considering the seven algorithms in [20], a similar algorithm to Delta (or adjustment on one of the seven algorithms) can provide a solution. This is mainly because the patterns found in XBRL are matched with the patterns defined in YAWL model and the algorithm Delta creates a YAWL model. Moreover the focus of the algorithm is on the data elements, so is the multiple instances problem. The MI problem should adopt the “top-down” approach (i.e. start with the root element and walk towards to leaf elements)

in order to specify the actual instances of a data element in a PDM (e.g. if the output data element has multiple instances, then its input elements will also have multiple instances).

In future, the dimensional construct can be used as a replacement of tuples. Then there will be more instances of the same data element (e.g. it might even be 100). In this case, first the dimensional construct should be mapped with the PDM structure. Once the dimensional structure is converted to a PDM, the same MI patterns will also be applicable and this proposed solution can be used. It is important to mention that the focus of the MI problem in this thesis is to find a general solution for the patterns seen in the XBRL structure. Therefore the solution proposed can provide a solution for "multiple instances without synchronization" and "multiple instances without a priori runtime knowledge" patterns. In this sense, our solution satisfies the needs of XBRL and additional research is required if one wants to implement other multiple instances patterns [21] into the formal definition of PDM.

4.2. Financial Filing Process

It has been clear with the "analysis" phase that both the PDM and the XBRL frameworks need extensions. This section sheds a light on how the formal definition of PDM and the current XBRL structure can be extended to support the "Reviewing and Reading" rules.

4.2.1. Overview of Extensions

First, the overview of these extensions is clarified for PDM and XBRL frameworks.

Table 4- Overview of Extensions

Missing Constructs	PDM Framework	XBRL Framework
Sub-processes	Sub-processes construct needs an extension in the PDM	This construct is related with the workflow process. Thus does not need an extension in XBRL
Status	Status information of a data element needs an extension in the PDM	Status information of a concept needs an extension in the XBRL
Department	Department information needs an extension in the PDM	Department information needs an extension in the XBRL
Role	Role information needs an extension in the PDM	This construct is related with the workflow process. Thus does not need an extension in XBRL
Job-grade	Job-grade construct needs an extension in the PDM	This construct is related with the workflow process. Thus does not need an extension in XBRL.
Order Relation	Order relations in constructs need an extension in the PDM	Order relations in constructs need an extension in the XBRL
Relation between constructs	Relations between the constructs need an extension in the PDM	Relations between the constructs need an extension in the XBRL

Resource Relation	Resource relation can be mapped in the organizational model therefore no extension is needed.	Resource relation can be mapped in the organizational model therefore no extension is needed.

4.2.2. Conceptual Solution for PDM

In the previous sub-chapter, the extended definition of PDM is represented by a 6-tuple, *(D; O; C; W; root; functions),* and functions are represented by a 6-tuple *(cost; time; cond; fprob; res, range).* Now the following constructs specified in Section 3.2.3 are added accordingly in order to match the requirements from the "reviewing & reading" rules with PDM.

- "Sub-processes": As described in the "analysis" chapter, activities are defined.
  - A: {Create, Review, Read by 1st, Made Statement, Read by 2nd, Sign-off-1, Sign-off-2, Archive} is the set of activities that can be included in an operation.
  - Create $\prec$ Review $\prec$ Read by 1st $\prec$ Made Statement $\prec$ Read by 2nd $\prec$ Sign-off-1 $\prec$ Sign-off-2 $\prec$ Archive denotes an order relation between each activity.
  - SP: O $\nrightarrow \mathcal{P}(\mathrm{A})$, a partial function specifies that an operation can have a "0" or more activities .

- "Status": Data elements can have different status information.
  - S: {Created, Reviewed, Read by 1st, Statement Made, Read by 2nd, Signed-off-1, Signed-off-2, Archived} is the set of possible states a data element can be in
  - Created $\prec$ Reviewed $\prec$ Ready by 1st $\prec$ Statement Made $\prec$ Ready by 2nd $\prec$ Sign-off-1 $\prec$ Sign-off-2 $\prec$ Archived denotes an order relation between each status
  - Status: D $\nrightarrow \mathcal{P}(\mathrm{S})$, a partial function specifies that a data element can have a "0" or more status information.

- "Department" information: "Audit", "Tax" and "Consultancy" departments are involved in the process.
  - DM: {Audit, Tax, Consultancy} is the set of departments in the organization.
  - DepD: D → DM specifies the responsible department for each data element.
  - DepW: W → DM specifies the department which a resource belongs to.

- "Roles" information: Each resource has a specific role
  - R: {Creator, Reviewer, 1st reader, 2nd reader, Signed-off-Deloitte, Signed-off-Client, Archival} is the set of roles a resource can fulfill.

- $\text{Creator} \prec \text{Reviewer} \prec 1^{st}\text{ reader} \prec 2^{nd}\text{ reader} \prec \text{Signed-off-Deloitte} \prec \text{Signed-off-client} \prec \text{Archival}$ denotes an partial order relation, meaning that a person with role "reviewer" is allowed to do all the work "creator" is allowed to do and potentially more.
- $\text{Role}: W \rightarrow \mathcal{P}(R)$ specifies that each resource can have a subset of roles.

- "Job-Grade" information: Each resource has a specific job-grade.
  - Jobgr: $W \rightarrow \mathbb{N}^+$ specifies the job-grade of each resource.

After defining the basic constructs in PDM, the relations (filing rules) are elaborated below

- Order Relation

The formalizations given above specify the set of possible values for each construct and the order relation is shown by partial order relation in these constructs.

- Relations between constructs

As can be understood, there are also relations (links) between the constructs which in turn specifies the "Reviewing and Reading" rules

  - "Activity", "Status" and "Role" have relations between each other (e.g. "Create" sub-process is done by "Creator" and the status of a data element will be "Created"). This relation can be shown as below
    - ARS= A x R x S indicates the Cartesian product of three sets.

This specification gives a general solution for specifying the relations between these constructs; however the actual elements from this relation should be as follows. This set will not be implemented in the PDM schema for the sake of simplicity.

ARS= {(Create, Creator, Created), (Review, Reviewer, Reviewed), (Read by 1st, 1st reader, Read by 1st), (Made Statement, 1st reader, Statement Made), (Made Statement, 2nd reader, Statement Made), (Read by 2nd, 2nd reader, Read by 2nd), (Sign-off-1, Signed-off-Deloitte, Signed-off-1), (Sign-off-2, Signed-off-Client, Signed-off-2), (Archive, Archival, Archived)}

- Resource Relation

With the relation shown above, binding requirements are specified (which role is allowed for each activity and what will be the status). However Resources have also relation with Roles (e.g. "Creator" and "Reviewer" cannot be the same person). This relation can be shown as below

- WR= W x R indicates the Cartesian product of two sets. Since this specific set Resources is not specified in the PDM, first this set should be defined (or an organizational model). After specifying this set (which resources are allowed for the filing), this relation can be applied in the PDM schema.

Now the extended definition of PDM is represented by 10-tuple *(D; O; C; W; root; functions; S; DM; A; R)* and functions are represented by 14-tuple *(cost; time; cond; fprob; res ,range; Status; DepD; DepW; SP; Jobgr; Role; ARS, WR).*

4.2.3. Conceptual Solution for XBRL

This section represents the conceptual solution on the simplified Dutch XBRL structure (Figure 7) by considering analysis at Section 3.2.4 and Table 4. XBRL can be extended with two constructs, the dimensional construct with using formulas and creating a new relationship in the definition linkbase.

1. By using the dimensional construct, XBRL can cover the status information of a data element. This relation can be defined in XBRL by creating hypercube-dimensions. In the hypercube structure, it has at least one axis and exactly one set of line item. In other words, this structure can be seen as a pivot-table function in such spreadsheets programs (for example, Microsoft Excel). A pivot table usually consists of row, column and data fields. In this case, the row which refers to the line items in XBRL is data elements, the column which refers to the axis in XBRL is status and the data fields are the actual values.
2. With this structure, the data fields show the actual status of the data elements and this information can be stored in the instance file (since the dimensional information is part of the "context" construct in the instance level, status information is shown as the part of the context).
3. Formulas will specify the relationships between the status of data elements (e.g. sign-off should be done after creator). In order to apply this structure, a dimension construct is needed to cover the data and the creation time of a data element and also the sequence of the possible status should be specified (e.g. "created" -> "reviewed" -> "Read by 1st" -> "Read by 2nd"-> "Signed-off").
4. If needed, formulas can used to check the required status information of data elements for each approach indicated in the Accountancy Approach document [8].

The conceptual solution of this extension is proposed in this chapter; however the implementation is not dealt within this project. There are two important reasons for this. First, the dimensional construct is considered as one of the limitations of this thesis. Second, implementing the dimensional construct with formulas into the current XBRL structure requires XBRL technical background. As can be understood, the same hypercube structure can also be used to capture the

department information. Then there will be a new column where the department is placed and a data field will show the appropriate department for each data element. However, for the simplicity of the solution, this way is not used and a new relationship in the definition linkbase is defined instead.

5. By using a new "arc role" function in the definition linkbase, this structure can be built. Like the relation between concepts in the presentation linkbase, the "parent-child" arcrole is used to define this relation, a new arc role "ResponsiblePartytoItem" is defined to show the link between concepts and departments.
6. Departments can be created as items in the data level and the relations between concepts and department can be defined in the definition linkbase.

The conceptual solution of the latter extension (numbers 5 and 6) is carried to the "implementation" chapter and the current XBRL structure is extended accordingly. The representation of the all conceptual solutions can be seen in Figure 10. Each number corresponds to the following extension given above.

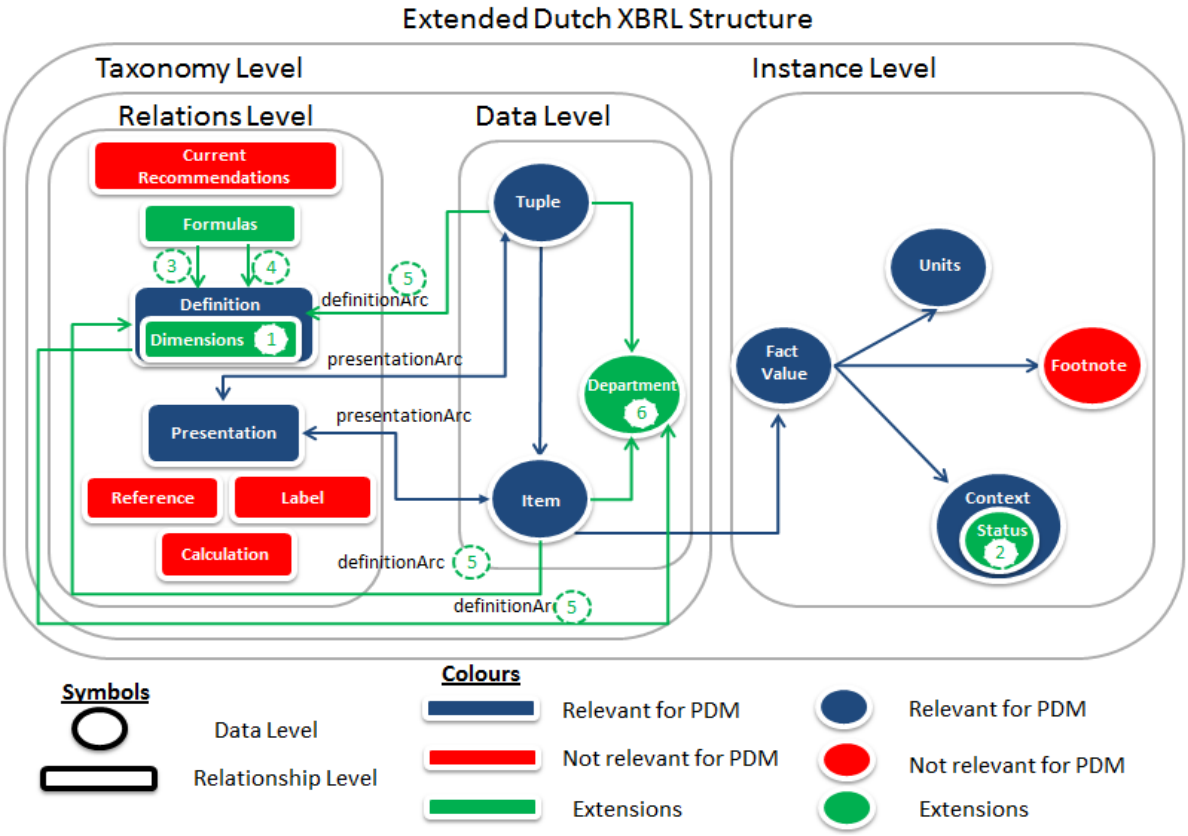


Figure 10- Extended Dutch Taxonomy Structure

4.2.4. Reflection

Now the formal definition of PDM is extended with the “Reviewing and Reading” rules and the XBRL structure is extended with the departmental information. These changes will also affect the research framework of the project. Like the multiple instances problem, the automatic generation of a process model from PDM should be taken into account for the “implementation” chapter. This is because now it will not be possible to generate a complete process model by considering the new definition of PDM. The next step is to specify “Order Relation”, “Relation between Constructs” and “Resource Relation” in the extended PDM schema. In order to specify these restrictions, a different file (e.g. XSLT) can be used. Afterwards, the current algorithms which are used to derive a process model from a PDM can be adopted (updated) by taking these restrictions into account. The starting point would be to create “sub-process” in YAWL model for each set of *Activities* which are now defined under operations in PDM. Then these restrictions can be used as an input for specifying the “Reviewing and Reading” rules in a process model.

It has seen that the current XBRL structure has also important shortcomings and should be extended with the conceptual solution given above in order to support the filing process. Even though the current structure of the Dutch taxonomy uses tuples as a replacement of dimensions, it is evident from our conceptual solution that the dimensions will play an important role for the further extensions of Dutch XBRL structure. Furthermore, an additional research is needed on the “Work-program” rules before specifying them in workflow management systems.

5. Implementation

In the previous chapter, the "design" part of the project is completed by proposing conceptual solutions for PDM and XBRL frameworks. This chapter deals with the implementation of these conceptual solutions as a proof of concept. It starts with extending the current PDM schema (see Appendix A) and the XBRL structure with the multiple instances and the "Reviewing and Reading" rules. Then the transformation procedure (XBRL Sample- XSLT- PDM) of the XML based Dutch taxonomy into the XML based PDM is given as a proof of feasibility.

5.1. Implementation of the Conceptual Solutions into PDM Schema

This sub-section represents the implementation part of the conceptual solutions. Extensions are made on the current PDM schema along with the necessary explanation (in this project we use Altova XMLSPY [4] as xml software to represent the PDM schema; however any XML editor can be used as well). In Appendix I, the extended PDM schema (with multiple instances and "Reviewing and Reading" rules) can be found.

5.1.1. MI patterns

Range function on data elements is used to specify the minimum occurrences and the maximum occurrences of data elements. The implementation of this solution is given as follows

- *Range* construct can be translated into "minVal" and "maxVal" values and added as an attribute in xml to the data elements. Then each construct is of integer type and the use is chosen as optional by considering the "backward compatibility". Then former PDM files in use can still be readable by ProM [16]. A default value "1" is given for the "minVal" indicating the single instance of a data element. A range between "1-100" is given for the "maxVal", assuming that a data element can have values between 1 and 100 for the Dutch taxonomy.

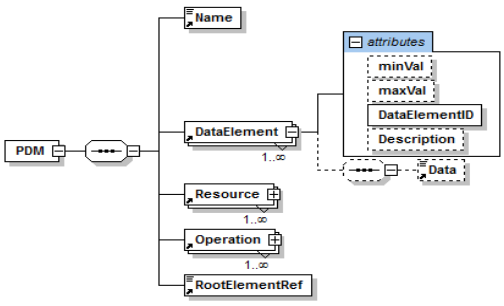


Figure 11- MI Pattern on PDM Schema

5.1.2. Financial Filing Rules

As can be seen from 4.2.2, both the PDM tuple and function tuple are extended with the "Reviewing and Reading" rules. In the original formal definition (see section 2.2.2), the sets (e.g. data elements,

operations) were translated to the PDM schema (.xsd file) as elements and the functions were translated as attributes. First, the implementation of the basic constructs added to the PDM tuple and function tuple is explained (see Figure 12).

- *Activities* is added as a construct to the PDM tuple. This construct is chosen as optional (minimum occurrence is "0") since an operation may not have an activity (sub-process). It has "Create", "Review", "Read by 1st", "Made Statement", "Read by 2nd", "Sign-off-1", "Sign-off-2" and "Archive" states. Enumeration constraint in xml is used to identify the possible values for this. Then the function under operations is specified for *Activities.*
- *Status* is added as a construct to the PDM tuple. This construct is chosen as optional (minimum occurrence is "0") since a data element may not have a status. It has "Created", "Reviewed", "Read by 1st", "Statement Made", "Read by 2nd", "Signed-off-1", "Signed-off-2" and "Archived" states. Enumeration constraint in xml is used to identify the possible values for this. Then this construct is added as an attribute to the definition of data elements.
- *Department* is added as a construct to the PDM tuple. Considering the fact that at least one department is involved in the filing process, this construct is chosen as compulsory (minimum occurrence is "1"). "Audit", "Tax" and "Consultant" departments are listed and enumeration constraint is used to identify this set in xml. Then this construct is added as an attribute to the definition of data elements and resources.
- *Roles* is added as a construct to the PDM tuple. Considering the fact that at least one role is involved in the filing process, this construct is chosen as compulsory (minimum occurrence is "1"). It has "Creator", "Reviewer", "1st Reader", "2nd Reader", "Signed-off-Deloitte", "Signed-off-Client" and "Archival" states. Enumeration constraint in xml is used to identify the possible values for this. Then this construct is added as an attribute to the definition of resources.
- Job-grade is added as an (optional) attribute to the definition of resources. Since the highest level is "14", a range between "0-14" is defined in the PDM schema.

Now the current PDM schema is extended with the basic constructs needed for the "Reviewing and Reading" rules. As we mentioned in the "design" chapter, there are three types of "Reviewing and Reading" rules which are "order relation", "relations between constructs" and "resource relation".

For "order relation", partial ordering is defined for the constructs in the conceptual solution and this relation has to be specified in the PDM schema as well. For "relations between constructs" there is a link between "Activities", "Status" and "Roles" and the relevant elements from this relation are

given in the “design” phase. For “resource relation”, first resources who are allowed for the filing process should be specified. After specifying this set, the restrictions (rules) between “Resource” and “Roles” sets can be implemented. Implementing these relations into the current PDM schema requires a different file (e.g. XSLT or another script for checking these restrictions) and technical knowledge of xml; therefore they are not dealt within the scope of this project. Now the current PDM schema is extended with the basic constructs from the “Reviewing and Reading” rules and the representation of the PDM tuple and function on the PDM schema is given below.

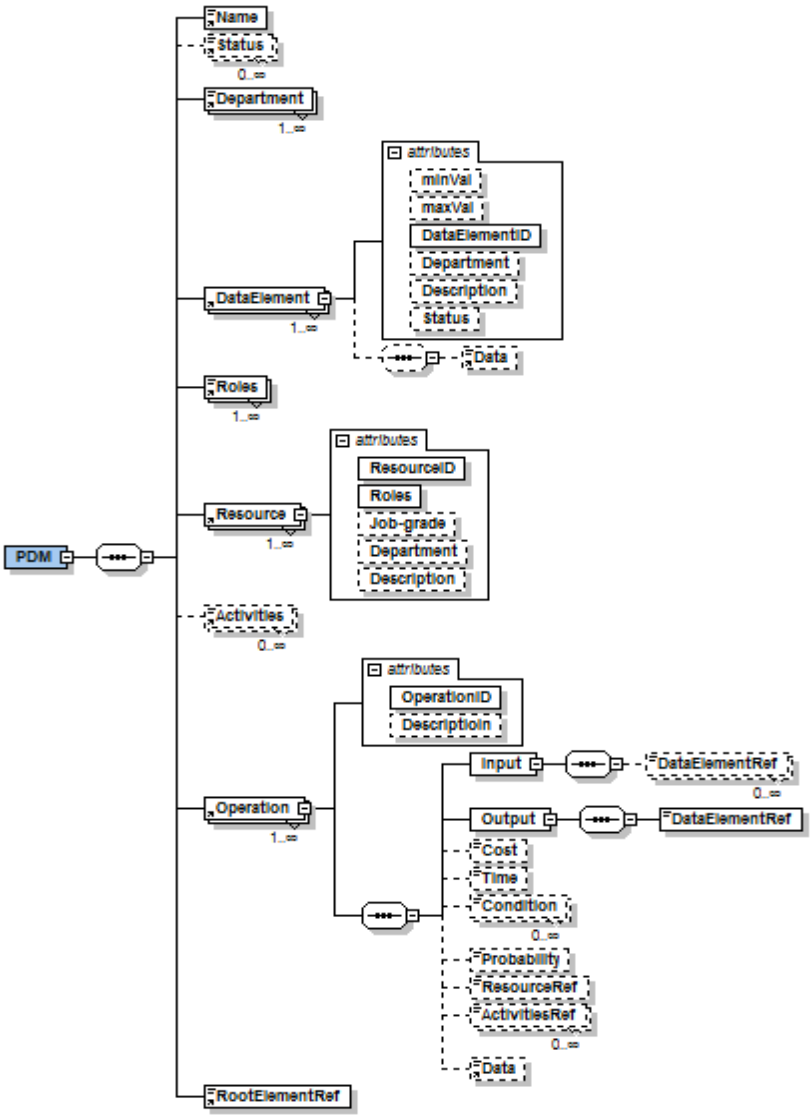


Figure 12- Representation of PDM tuple and function tuple for Financial Filing Rules

By considering these extensions, the new schema definition is given in Appendix E.

5.2. Implementation of the Conceptual Solutions into XBRL

As shown in the “design” chapter, the conceptual solution is proposed for the current XBRL structure and it is extended with the department information. In order to do this extension, steps 5 and 6

given in Figure 10 have to be implemented. This section explains the implementation of this solution by showing the xml clips of the necessary extensions.

First, “Audit”, “Tax” and “Consultancy” departments are created as an element in XBRL. The xml clip is given as follows.

```
<xsd:schema>
        <xsd:import namespace="http://www.xbrl.org/2003/instance"
        <xsd:element name="Audit" id="Audit" type="xbrli:stringItemType"
substitutionGroup="xbrli:item"/>
        <xsd:element name="Tax" id="Tax" type="xbrli:stringItemType"
substitutionGroup="xbrli:item"/>
        <xsd:element name="Consultancy” id="Consultancy" type="xbrli:stringItemType"
substitutionGroup="xbrli:item"/>
</xsd:schema>
```

As can be seen from the xml clip above, an element in XBRL is created for each department. The “type” refers to the string and the “substitutionGroup” is indicating that this element is item. After creating items in XBRL, the relationship between data elements and departments can be defined by specifying a relation in the definition linkbase (“ResponsiblePartytoItem” arc role is created in order to maintain this relationship). The xml clip is given as follows

```
<link:linkbase >
  <link:arcroleRef arcroleURI="http://www.xbrl.org/2003/arcrole/ResponsiblePartytoItem"
xlink:type="simple" xlink:href="default.xsd#TE"/>
  <link:definitionLink xlink:type="extended" xlink:role="http://www.xbrl.org/2003/role/link">
    <link:loc xlink:type="locator" xlink:title="FixedAssets"/>
    <link:loc xlink:type="locator xlink:title="Audit"/>
    <link:definitionArc xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/ResponsiblePartytoItem" xlink:from=" FixedAssets"
xlink:to="Audit" xlink:title="definition: FixedAssets to Audit" order="1.0"/>
    <link:loc xlink:type="locator" xlink:title="CurrentAssets"/>
    <link:definitionArc xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/ResponsiblePartytoItem" xlink:from="CurrentAssets"
xlink:to="Audit" xlink:title="definition: CurrentAssets to Audit" order="1.0"/>
    <link:loc xlink:type="locator" xlink:title="ProfitLossBeforeTax"/>
<link:loc xlink:type="locator" xlink:title="Tax"/>
    <link:definitionArc xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/ResponsiblePartytoItem"
xlink:from="ProfitLossBeforeTax" xlink:to="Tax" xlink:title="definition: ProfitLossBeforeTax to Tax"
order="1.0"/>
  </link:definitionLink>
</link:linkbase>
```

As can be seen from the xml clip above, first locators are specified for departments and items. Then the relations between these are indicated by the “xlink:arcrole” construct. As mentioned earlier, the shared and unique concepts are specified for each report in XBRL and this extension will address the data ownership for the reports.

5.3. Transformation Procedure (XBRL – XSLT- PDM)

As we mentioned earlier, this project focuses on the upper left part of the research framework (Figure 3) and investigates the PDM and XBRL constructs with respect to the translation of the XBRL taxonomy to the PDM structure. Since both (PDM and XBRL) of the frameworks use XML-based language, XSLT (Extensible Stylesheet Language Transformations) is chosen for this transformation procedure. However any advanced computer programming language (e.g. Java, C++) can do this job as well. The magnified view of the upper left part of the research framework is given as follows.

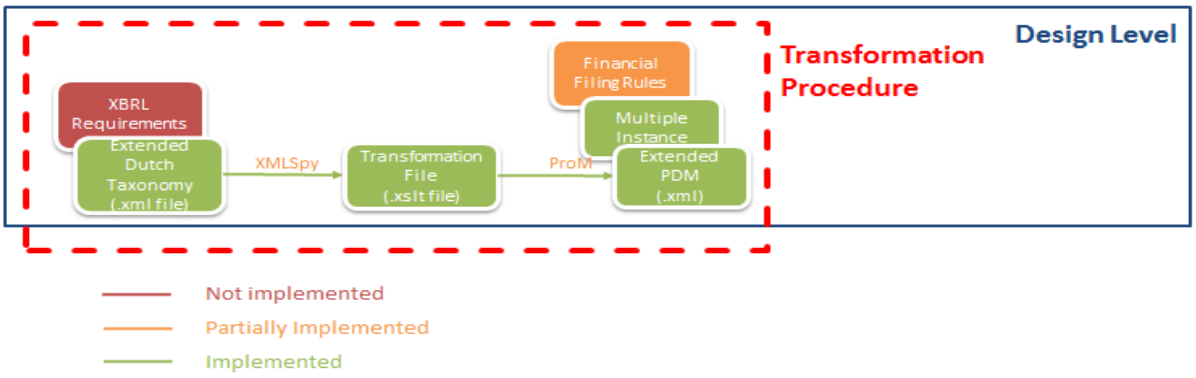


Figure 13- Transformation Procedure

Now the transformation file includes a translation of basic XBRL elements to PDM (e.g. data elements, operations), multiple instances elements to PDM and the extended construct in XBRL for the "Reviewing and Reading" rules. According to Figure 13, the transformation procedure should follow the steps given below

- In Section 5.3.1, characteristics of the sample are explained. The sample also used as a prototype in the previous research [19] will be used in this project for the automatic conversion (the extended XBRL structure, with the new arc role in the definition linkbase, is added to the sample as well).
- In Section 5.3.2, the transformation file prepared for the translation of extended Dutch taxonomy to PDM is explained.
- In Section 5.3.3, PDM is automatically generated by running the XSLT file. Then the output of this conversion can be used in ProM to visualize the PDM structure.

The transformation procedure is described in the following sub-sections by explaining each bulletin given above.

5.3.1. Characteristics of the sample

The representative sample used in [19] is given in Figure 14. The Fujitsu XWand tool [11] is used for creating a sample (Appendix F for more information). The sample can be found in Appendix I.

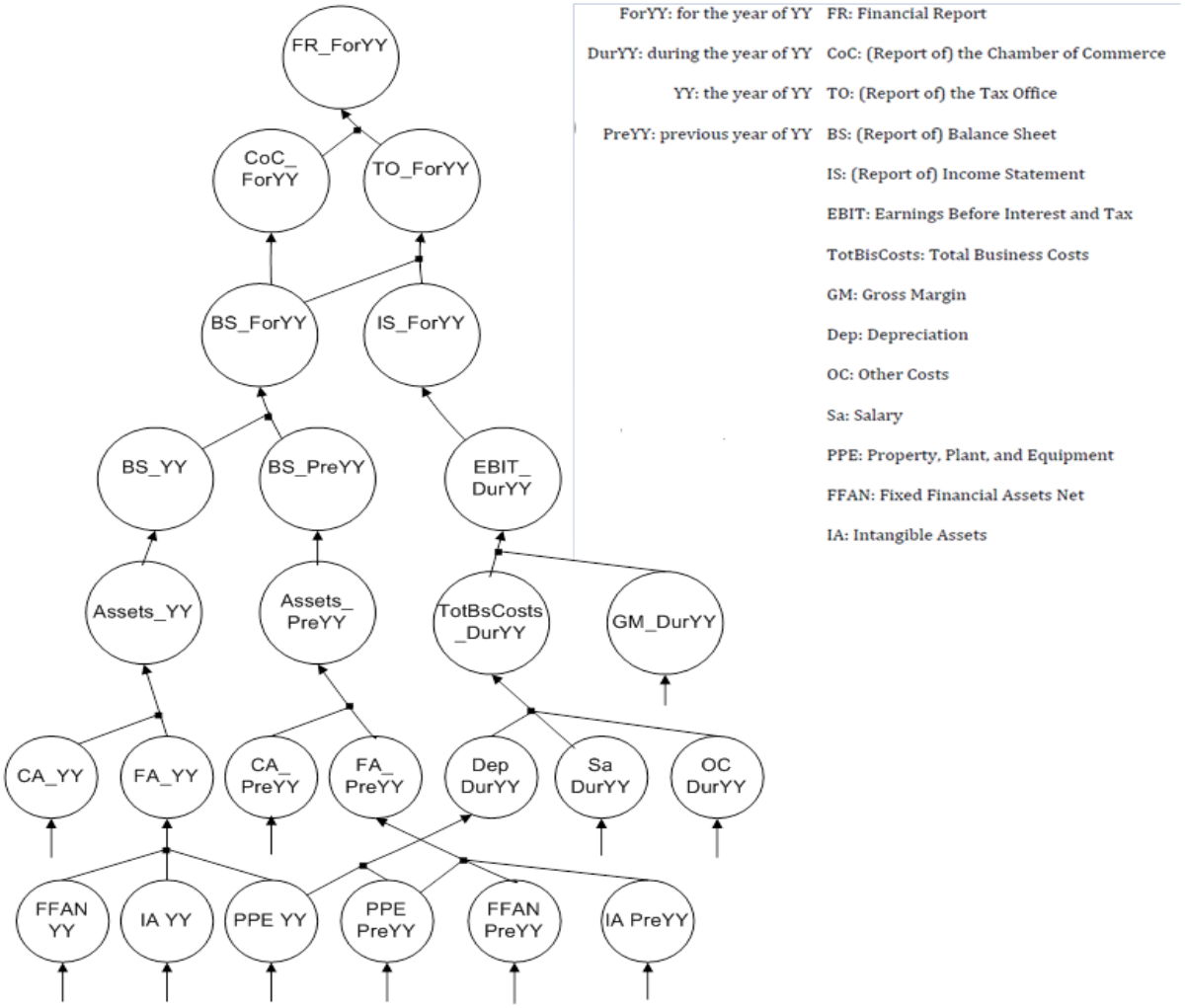


Figure 14- PDM converted from the sample XBRL taxonomy [24]

Since our scope is narrowed down to the Dutch taxonomy, Figure 10 can be used to specify the characteristics of our sample. The report name is “Financial Report” and it consists of the reports of “Chamber of Commerce” and “Tax Office”.

- As can be understood, some data elements (e.g. Assets, Balance Sheet) have multiple instances in the report (for instance, Balance Sheet for the Year of 2011 consists of Balance Sheet for the Year of 2010 and 2011). In Figure 14, these multiple instances are shown as two separate data elements as Balance Sheet YY (for the Year YY) and Balance Sheet PreYY (previous year of YY).
- The presentation linkbase from the relations level is used to represent the relations between data elements. The definition linkbase, the extended XBRL structure, is used to capture the department information for each data element. Audit and Tax departments are responsible for preparing the report of “Balance Sheet”, and “Tax Office” respectively.

Several assumptions are also made while preparing the report.

- For testing purposes, the presentation and definition linkbases are merged into one file. This is mainly because an XSLT script is normally applied to only one file at the time.
- Abstract items (e.g. headers of the report, sub-division titles) are not taken into account.
- As mentioned in Section 2.3.2, the Dutch taxonomy architecture is a 3-layered structure; however the simplified version is used in this project as a proof of concept.
- Conceptual solutions of the other XBRL requirements given in Figure 10 are not used while preparing the sample.

5.3.2. Conversion File (XSLT)

In the previous section, the characteristics of the sample taxonomy are explained along with the assumptions. This section explains how the xml based Dutch taxonomy is converted into the xml based PDM structure. In order to do this transformation, the XSLT file should be prepared first. It starts with explaining the basic translation (set of data elements, set of operations, root element) of the PDM. Afterwards, the implementation of the multiple instances problem and the financial filing rules are shown and the XSLT code is gradually extended. In Appendix G, the description of the whole translation can be found.

Basic Translation

Any XSLT engine can do this transformation, but in this project we use XMLSPY (Appendix F for more information) since it gives a better visualization. First, concepts in XBRL are converted into the data elements in PDM. Then the parent-child relation between the data elements in XBRL is converted into the operations in PDM. Finally, the root item in XBRL is converted into the root element in PDM.

Multiple Instances

As can be seen from the sample in Figure 14, some data elements (e.g. Assets, Balance Sheet) have multiple values for “the year of YY” and “the previous year of YY”. Now these elements can be shown in one data element by using the solution for the multiple instances problem. Since the XSLT file is converting the XBRL taxonomy to the PDM, it is not possible to capture the “context” attribute for the direct translation. Therefore the XSLT script is updated to give a certain range “1-2” for each element in the sample. This range is given for the “design time”, and in the “run-time” exact values are specified in the PDM (e.g. in “run-time”, “Balance Sheet” element will be 2:2 meaning that Balance Sheet has one value for “the year of YY” and one value for “the previous year of YY”).

Financial Filing Rules

Considering the “Reviewing and Reading” rules, a conceptual solution is proposed for XBRL and the current structure is extended with the department information. For PDM, the department is added as an element in xml to the definition of data elements (see Figure 12). Therefore the transformation procedure extracts the department information specified in the definition linkbase in XBRL and matches with the department attribute in PDM. Even though the extended PDM schema supports the constructs from the “Reviewing and Reading” rules, a list of requirements given in Section 4.2.3 (conceptual solution for XBRL) should first be implemented in the current XBRL structure. Then these constructs can be translated into the PDM.

5.3.3. The Automatic Translation

After preparing the XSLT file, the XBRL sample can be used as an input to generate PDM automatically (In Appendix J, the description is given on how to repeat this transformation procedure). In Figure 15 below, the PDM structure converted from the sample can be seen.

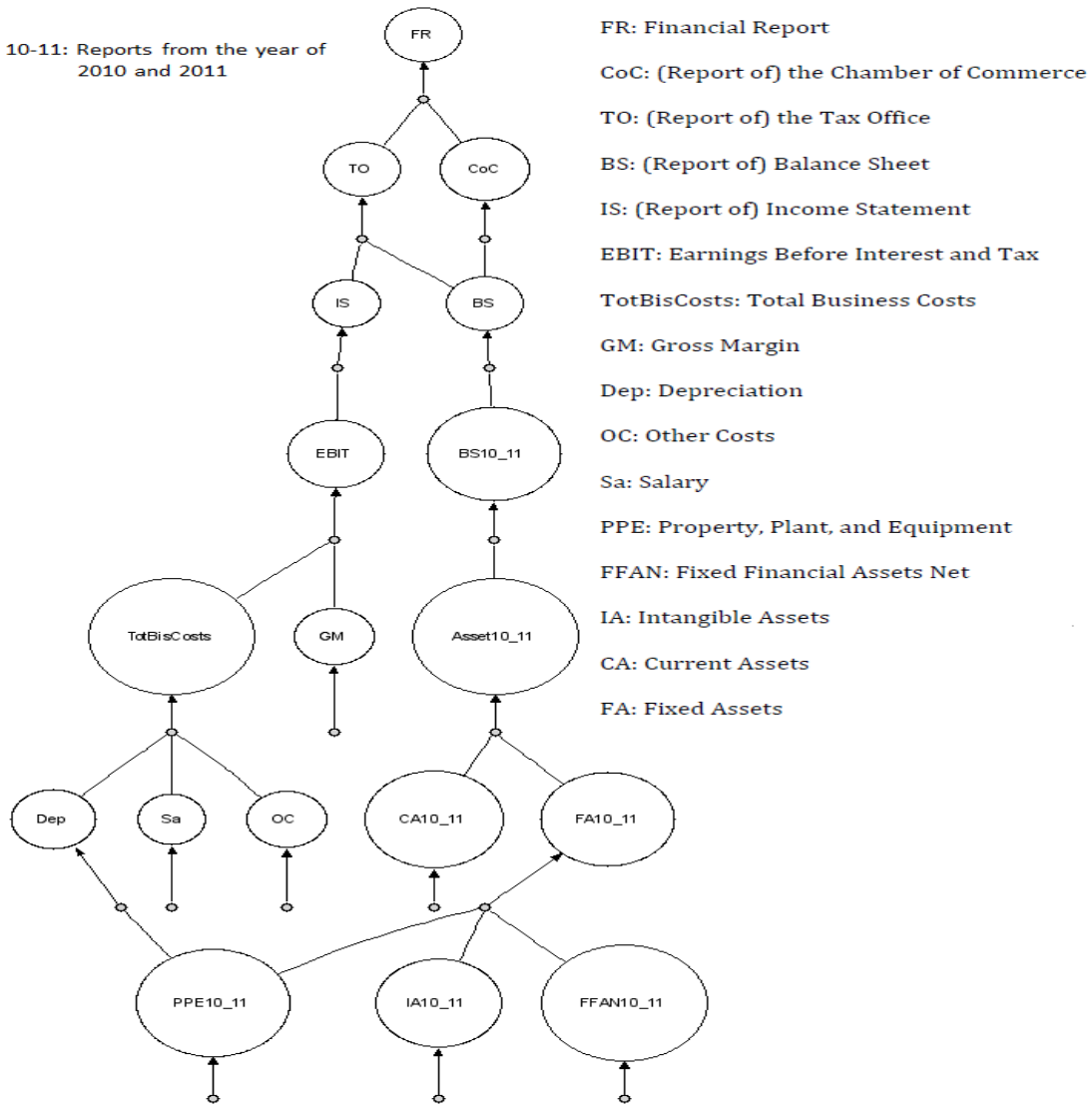


Figure 15- Representation of the Output in ProM

Now the solution of the multiple instances problem is applied to the PDM structure and the duplicate leafs, such as “Balance Sheet”, “Fixed Assets”, are removed from the report. Instead, “minVal” and “maxVal” attributes are used to denote these leafs. As can be understood from the picture above, the “minVal” and “maxVal” constructs and the department information are not yet recognized by ProM. The functionality in ProM that can import the PDM structure from the xml file needs to be updated in order to visualize such constructs. In Appendix I, the XBRL sample, XSLT code and the output xml file (PDM) can be found.

5.4. Reflection

As a reflection of this chapter, first the implementation of the formal definitions is made on the PDM schema and the current XBRL structure is extended with the department information. Afterwards, the transformation procedure takes place where the xml based XBRL taxonomy is automatically converted into the PDM structure. First the basic requirements (data elements, operations, root element) are translated. After this step, an assumption is made on the multiple instances problem. The main reason of this assumption is that even though a conceptual solution is needed for the design level of PDM, the corresponding information in XBRL is stored at the instance level. For the financial filing process, the department information in XBRL is automatically translated to the PDM as the part of the “Reviewing and Reading” rules. Therefore an important step is made to specify the data ownership of the elements.

As the next step of this project, three important directions can be proposed. First one is to investigate the real XBRL architecture for the automatic translation of PDM. This project uses one presentation and one definition linkbases and proofs that the first step of the research framework can be automatized. Considering the Dutch XBRL architecture, the transformation file has to incorporate with the files from different layers in order to generate a PDM. In this sense, an advanced computer programming language (e.g. Java, C++) is needed to make this interface automatic in practice. A first step to use this approach in practice is made by proposing the translation file given as a proof of concept. Extending this file by considering the Dutch XBRL architecture will make it applicable for any Dutch XBRL taxonomy and then it can be used in practice. Second step should be to implement the requirements listed for the “Reviewing and Reading” rules into the current XBRL structure. Afterwards, these constructs can be used for the translation of XBRL-PDM. Third step would be to implement a file in the extended PDM schema to perform “Order Relation”, “Relation between Constructs” and “Resource Relation” restrictions for the “Reviewing and Reading” rules. Then adapting (or improving) algorithms for deriving process models from PDM can be made by using the extended PDM schema and this file as an input.

6. Conclusion

In this project, we have investigated the possibilities on how to extend the formal definition of PDM to support XBRL information in a workflow management system. This chapter is divided into four parts. First, the summary of the most important findings is addressed along with the project framework. Second, general assumptions and limitations made on this project are explained. Third, the implications of these results on the next steps of the research framework and the suggested steps for practice are discussed. Finally, directions for the further researches are elaborated in the last section.

6.1. Summary

This project adopts the research framework from [19] and focuses on the assumptions and limitations of that project. As the outcome of this research, the new framework is given in Figure 16.

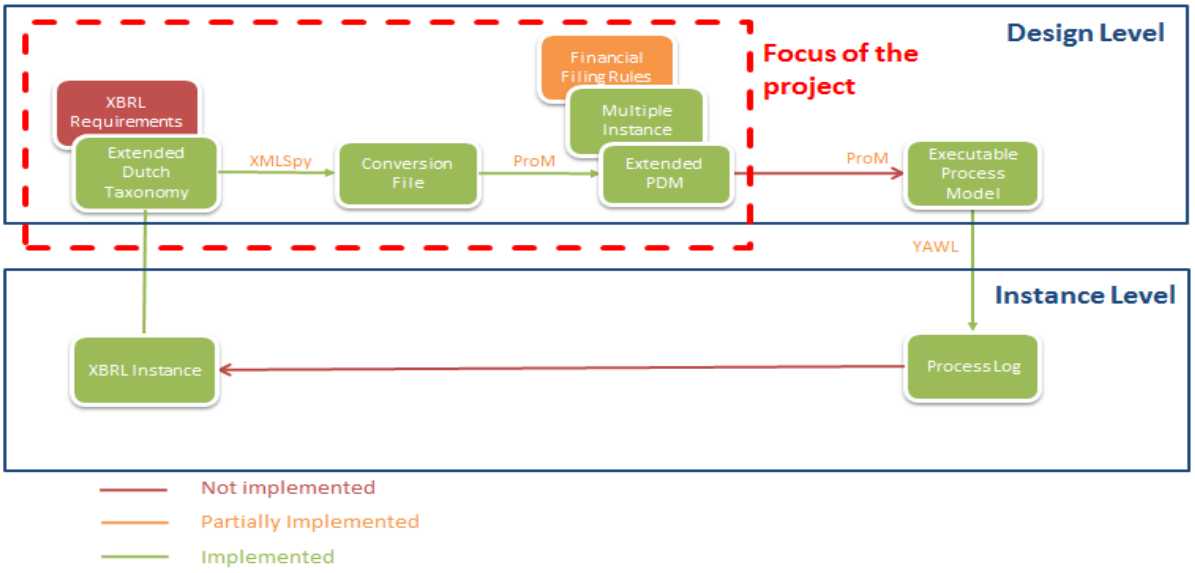


Figure 16- Updated Project Framework

As to answer our first question "Which requirements on the PDM are posed by the XBRL taxonomy?" the XBRL and the Dutch taxonomy structures are investigated. Each structure is partitioned into Relations Level, Data Level, and Instance Level and the relevant constructs are identified for the use of PDM. As a result of this analysis, similarities and differences between XBRL and PDM frameworks have been clarified and this will give an idea on how can the PDM be generated by using the Dutch XBRL structure. The previous research [19] in this sense starts with taking a XBRL sample and uses it to derive the PDM without investigating the both frameworks.

While answering our second question "How can the multiple instances be added to the semantics of PDM?" the analysis made clear from the first question is used to identify the multiple instances patterns seen in XBRL. In the XBRL-based reports, generally two or more sets of values are needed

for the same taxonomy concept; however the current definition of PDM does not support this function. By proposing a solution to this question, the problem of creating duplicate data elements in PDM for the taxonomy concepts having more than one value is solved. In the previous research [19], this problem is solved by duplicating the same elements in PDM by giving different labels and it is listed as one of the limitations of that project.

In our third question "How can the financial process rules be added to the semantics of PDM?" first a walkthrough has been made in order to clarify which steps are in the filing process (considering the Deloitte's process) and as a result of this walkthrough the "as-is" and "to-be" states of the process are clarified. Then three types of rules are classified as "Work-program", "Content-related", and "Reviewing and Reading Rules". As the outcome of this analysis, a list of requirements has been identified for both frameworks with regards to "Reviewing and Reading" rules. By extending the definition of PDM with these rules, control perspective of the filing process is added to the model in order to support data assurance in this process. By proposing a solution on the XBRL, the data ownerships have been specified for the shared taxonomy concepts. It is important to note that this research takes the classification of the filing rules in [19] as a starting point and a detailed analysis is given on the "Reviewing and Reading" rules.

In the last question of the project, "How can the Dutch taxonomy be translated automatically to the PDM?" the extended sample Dutch taxonomy is converted to the extended PDM without any manual interaction as a proof of concept. For this purpose, the conversion file is prepared to make this transformation. In the previous project [19], this conversion was manually performed (by hand) and making this step automatic creates a more reliable process at the end.

To conclude the important results of the project as a reflection on the project framework, the results can be split into conceptual (theoretical) and practical contributions. From the conceptual side, the formal definition of PDM is extended with the multiple instances and "Reviewing and Reading" rules and the current XBRL structure is extended with the necessary construct from the "Reviewing and Reading" rules. From the practical side, a list of requirements is proposed for XBRL which are important to support the filing process and the step from XBRL to PDM is automated showing that XBRL can be used with PDM in workflow management systems.

6.2. Assumptions & Limitations

While putting an emphasis on the challenges tackled within this project, there are also several assumptions & limitations made along the way.

The first assumption is that this project only takes the XBRL 2.1 base specifications into account while specifying the relevant constructs for the PDM. However the current recommendations (e.g. the formula linkbase, dimensions, generic links, inline XBRL rendering) can also play an important role in extending the XBRL structure. In practice, current recommendations are commonly used and give flexibility to the XBRL designers. Second assumption is made on the PDM structure for the multiple instances of data elements. This is because sets of fact values in XBRL are specified and distinguished in the instance level; however the multiple instances problem needs a solution in the design level. Therefore it is not possible to capture this information at the design level. Third assumption is that a list of important shortcomings of the XBRL (e.g. dimensions and formulas) is given for the "Reviewing and Reading" rules. However the current XBRL structure is not extended with such constructs and they are taken as one of the limitations of this project. The fourth assumption is that the current PDM schema is extended with the necessary constructs from the "Reviewing and Reading" rules. However showing the relations between constructs in the PDM schema requires technical knowledge of xml (e.g. writing a XSLT code, or another script). Therefore they are not dealt within the scope of this project. The last assumption is that only one presentation and one definition linkbase are used as a proof of concept while showing the XBRL-PDM step is repeatable and automatic. However the architecture of the Dutch taxonomy is built up using a modular approach and there are different layers which are interrelated with each other in the taxonomy structure.

6.3. Implications

This section gives a reflection on the implications of the project for practice and explains the suggested steps considering the next steps of the project framework.

First implication is that solving the list of requirements in XBRL by using the dimensions and formulas. With the use of XBRL, the data classification for reports has been made and specifying the status information of data elements along with the other requirements can support the filing process in a better way. Second implication is to work on the extended PDM schema by considering the next steps of the project framework. One way is to write a XSLT code (or another script) to apply restrictions in the PDM schema for the "Reviewing and Reading" rules. After preparing this file, it will be used as an input for deriving a process model from a PDM. Current algorithms will not work completely while deriving a process model since the formal definition is now changed. Therefore a research on adapting these algorithms is needed by using the extended PDM definition and this file as an input. Third implication is to work on the interface (XBRL-PDM) and make this step automatic in practice. Currently we use one presentation and one definition linkbase to show that this step can be made automatically. Since the Dutch taxonomy is built up using a modular approach, the

transformation file has to incorporate with the files from different layers in order to generate a PDM. In this sense, an advanced computer programming language (e.g. Java, C++) is needed to make this process automatic in practice.

6.4. Future Research

Based on the outcome of our analysis, this section proposes different future researches for the XBRL and the PDM framework.

For the XBRL framework, a list of requirements has been specified and the importance of these requirements is stressed out throughout the thesis. Implementing the dimensions and formulas constructs with regards to the "Reviewing and Reading" rules and mapping them with the PDM structure will be an important step to use XBRL in workflow management systems for practice. Second future direction would be to investigate the Dutch XBRL structure for the automatic translation of XBRL to PDM (investigation of the XBRL architecture, how each layer is interrelated with each other and which concepts in each layer are constructing which part of the reporting-tree PDM). Then it will be applicable for any XBRL taxonomy. Third research direction would be to work on the "Work-program" rules. In this direction, meetings with Audit and Tax should be made to specify these rules on how to use them in workflow management systems. Currently these rules are largely depending on auditor's experience and changing instance to instance however, based on their comments, a research can be made on this area to structure them in a way that workflow model can interpret and understand.

For the PDM framework, in order to include all the relations from "Reviewing and Reading" rules (e.g. partial ordering relation in constructs such as "Status" information, or relations between different constructs as "Status", "Activity" and "Role"), a file next to the PDM schema is needed. Writing a XSLT file (or another script) for applying these relations on the schema will be a starting point. Afterwards, a research can be done on the current algorithms [20] which are used for deriving process models from PDM by considering the extensions made in this project (multiple instances and "Reviewing and Reading" rules). Another research direction would be to work on a tool to directly execute PDM. Then it will skip the automatic generation of the process model and create an efficient process at the end. With this tool, the possible implications of the extended PDM on deriving a process model will be circumvented. In future, if a PDM engine is decided to be used for the direct execution of the PDM, the PDM instance schema is also needed to show the actual PDM design which will be used as an input for the engine.

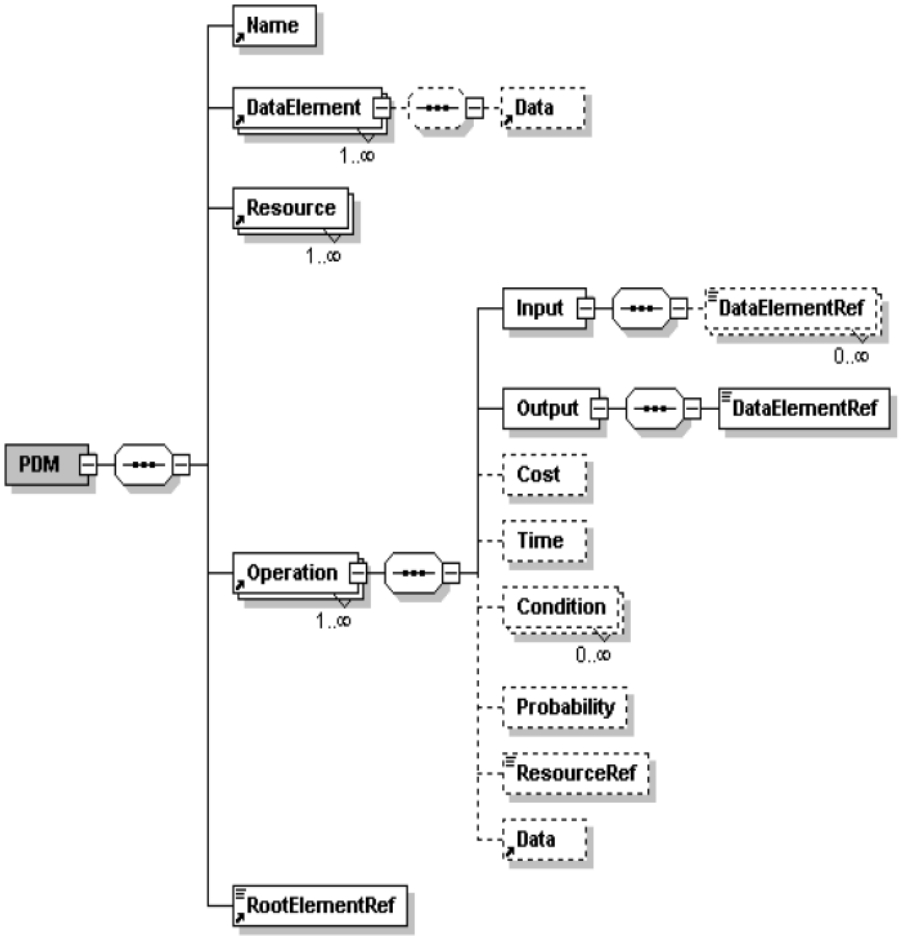
References

1. W.M.P. van der Aalst, On the Automatic Generation of Workflow Processes based on Product Structures, Computers in Industry, 39:97–111, 1999.
2. W.M.P. van der Aalst, A.H.M., ter Hofstede, YAWL: yet another workflow language, Information Systems, 30:245–275, 2005.
3. W.M.P. van der Aalst, A.H.M. ter Hofstede, M. Weske, Business Process Management: A Survey, Berlin: Springer-Verlag, International Conference on Business Process Management, Vol 2678, 2003.
4. Altova XMLSPY, Home page, http://www.altova.com/xmlspy.html
5. Belastingdienst, Horizontal Oversight, http://www.belastingdienst.nl/variabel/toezicht/toezicht-01.html
6. Burgerlijk Wetboek, Home page, http://www.wetboek-online.nl/site/home.html
7. Deloitte Innovation Services Netherlands, Home page, http://www.deloitte.com/view/nl_NL/nl/diensten/innovation-services/index.htm
8. Deloitte Nederland, MKB Accountancy Approach, Het uitvoeren van samenstellings- en beoordelingsopdrachten in de MKB praktijk, 2010
9. Deloitte Nederland Consultancy, Audit and Tax Process Models, 2011
10. Ernst & Young, US GAAP vs IFRS The basics 2009, http://www.ey.com/Publication/vwLUAssets/IFRS_v_GAAP_basics_Jan09/$file/IFRS_v_GAAP_basics_Jan09.pdf
11. Fujitsu XWand XBRL, Home page, http://www.fujitsu.com/global/services/software/interstage/solutions/xbrl/
12. B. Groenveld, The Dutch Taxonomy Structure 5.1., Deloitte Nederland
13. A. R. Hevner, Design Science in information systems research, Mis Quaterly, 2004.
14. IFRS, International Standard Accounting Board, http://www.ifrs.org/Home.htm
15. N. Nissanke. Introductory Logic and Sets for Computer Scientists. Addison-Wesley, Harlow, England, 1999.
16. ProM, Home page, http://prom.win.tue.nl/tools/prom/.
17. H.A. Reijers, S. Limam Mansar, and W.M.P. van der Aalst. Product-Based Workflow Design. Journal of Management Information systems, 20(1):229–262, 2003.
18. Standard Business Reporting Programma, NL-FRIS Eisen aan en toelichting op gebruik Nederlandse taxonomie XBRL instance documenten gebaseerd op de NT2011, http://www.sbr-nl.nl/fileadmin/SBR/documenten/NT_2011/NL-FRIS_NT2011.pdf
19. Y. Sun, How to use XBRL in Workflow Management Systems: Eindhoven University of Technology, Master Thesis, 2010.
20. I. Vanderfeesten, Product-Based Design and Support of Workflow Processes, Eindhoven University of Technology, PhD Thesis, 2008.
21. Workflow Pattern Net, Multiple Instances, Home page, http://www.workflowpatterns.com/patterns/control/index.php

22. XBRL, International Transforming Business Reporting, http://www.xbrl-nederland.nl/upload/21/pub/20020217%20Final%20XBRL%20Brochure.pdf
23. XBRL, Organization, http://www.xbrl.org/AboutTheOrganisation/
24. XBRL, XBRL 2.1. Specifications, http://www.xbrl.org/Specification/ BRL-RECOMMENDATION-003-12-31+Corrected-Errata-2008-07-02.htm

Appendix A- The XSD Schema for PDM

A graphical representation of the XSD schema for a PDM is given below. The schema describes the structure of the XML file containing the PDM. Note that in the current definition, an operation contains exactly one output element and zero or more input elements. The attributes specifying the execution cost, execution time, etc. of the operation are optional.



Appendix B- Characteristics of the Relevant Constructs in XBRL

In this section, the characteristics of the constructs defined in Figure 7 and Figure 8 are described. It starts with the data level and followed by relations and instance level.

Data level

Item

- An item has an ID
- An item has either an instant or a durational period type. An instant period type corresponds to a single value at a certain point in time whereas a durational period type indicates a value for the period between two points in time.
- An item has a nillable attribute (either true or false). Nillable attribute appears on all taxonomy elements and have the default value as “true”, and there is no need for any extension to define an element with nillable “false”. In other words this attribute is not used in the XBRL structure.
- An item has a data type (e.g. string, monetary)
- For monetary items, a balance attribute can be set. The balance attribute is a sign (plus or minus) used for the monetary type.
- An item can be abstract attribute. An abstract item is for visualization purposes and does not have a value in the instance level.
- An item can be a child or parent of another item or tuple

An xml clip from the Dutch XBRL taxonomy for an item construct is given below. In this example the name of the concept is “Assets”. “Assets” is with the type of “xbrli:monetaryItemType” in the substitutionGroup an “xbrli:item”. And it is with the periodType of “instant”.

```
<element
id="fr_Assets"
name="Assets"
type="xbrli:monetaryItemType"
substitutionGroup="xbrli:item"
xbrli:periodType="instant"
nillable="true" />
```

An xml clip for an abstract item is given below. In this example the name of concept is “balancesheet”. “Balancesheet” has the abstract property “true” and is with the type of "xbrli:stringItemType" in the substitutionGroup an “xbrli:item”. And it is with the periodType of “instant”.

```
<element
id="ci_balancesheet"
name="balancesheet"
type="xbrli:stringItemType"
substitutionGroup="xbrli:item"
abstract="true"
xbrli:periodType="instant"/>
```

Tuple

- A tuple has an ID.
- A tuple is a container and it contains items.
- A tuple has no value and no data type information by itself.
- A tuple is not abstract; however it does not have a value by itself in the instance level. It has values from the items it contains.
- A tuple can be nested (i.e. a tuple in a tuple in a tuple…)
- A tuple has a nillable attribute (either true or false) but it is not used in the XBRL structure

An xml clip for an abstract item is given below. In this example, the name of the tuple is "managementInformation" and it is in the substitutionGroup an “xbrli:item”. "ManagementInformation" tuple contains a set of items "s:managementName", "s:managementTitle",and "s:managementAge".

```
<element name="managementInformation" substitutionGroup="xbrli:tuple">
 <complexType>
  <complexContent>
   <restriction base="anyType">
    <sequence>
     <element ref="s:managementName"/>
     <element ref="s:managementTitle"/>
     <element ref="s:managementAge" minOccurs="0"/>
    </sequence>
    <attribute name="id" type="ID" use="optional"/>
   </restriction>
  </complexContent>
 </complexType>
</element>
```

The relations level

Current Recommendations

- Besides basic specifications, new specifications are developed from the requirements statements and discussed in the XBRL community. If the work is agreed upon and declared to be complete, then it is released as an official XBRL Recommendation.
- It is optional to use the current recommendations and one can customize the XBRL architecture by using additional constructs from these recommendations.

Reference Linkbase

- It associates concepts with citations of some body of authoritative literature (e.g. IAS, para 68, Common Law; book IV; 156-32;b).
- This linkbase is used in the Dutch taxonomy however the purpose is not relevant for a PDM.

An xml clip for the label linkbase is given below. In this example, concept "IAS_CashCashEquivalents" has references to name “IAS”, number “7”, paragraph “Cash Flow”, subparagraph “7”.

```
<link:reference xlink:type="resource" xlink:role="http://www.xbrl.org/2003/role/reference"
xlink:label="IAS_CashCashEquivalents">
<ref:Name>IAS</ref:Name>
<ref:Number>7</ref:Number>
<ref:Paragraph>Cash Flow Statements</ref:Paragraph>
<ref:Subparagraph>7</ref:Subparagraph>
</link:reference>
```

Label Linkbase

- In this linkbase, a concept in the taxonomy with labels in different languages or for different purposes (e.g. a short label PPE compared to its long label Property, plant and equipment).
- This linkbase is used in the Dutch taxonomy however the purpose is not relevant for a PDM.

An xml clip for the label linkbase is given below. In this example, one locator, one arc and one resource are defined. This example gives the relationship that the "fr_AssetsDeadline" concept is related to the concept "fr_AssetsDeadline_lbl". And concept "fr_AssetsDeadline_lbl" is with the type of xml:lang="en".

```
<labelLink xlink:type="extended"
   xlink:role="http://www.xbrl.org/2003/role/link">
   <loc xlink:type="locator"
    xlink:label="fr_AssetsDeadline"
    xlink:href="Financial%20Report.xsd#fr_AssetsDeadline" />
   <labelArc xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/concept-label"
    xlink:from="fr_AssetsDeadline" xlink:to="fr_AssetsDeadline_lbl" />
   <label xlink:type="resource"
    xlink:label="fr_AssetsDeadline_lbl" xml:lang="en"
    xlink:role="http://www.xbrl.org/2003/role/label">Assets Deadline</label>
</labelLink>
```

Calculation Linkbase

- The calculation linkbase defines basic calculation rules (e.g. summation and subtraction), which must apply for all instances of the taxonomy.
- This linkbase associates concepts with other concepts so that values appearing in an instance document may be checked for consistency. For example two concepts A, B can be summed up to a third concept C, such that C = A + B.
- Since the function of this linkbase is limited in use, it is not used in the Dutch taxonomy.

An xml clip for the calculation linkbase is given below. In this example, two locators and one arc are defined. This example gives the relationship that the first input of concept “Assets (fr_Assets)” is concept “Current Assets (fr_CurrentAssets)” with the *weight* of one.

```
<loc xlink:type="locator"
xlink:label="fr_Assets"
xlink:href="Financial%20Report.xsd#fr_Assets" />
```

```
<loc xlink:type="locator"
xlink:label="fr_CurrentAssets" 17 | P a g e
   xlink:href="Financial%20Report.xsd#fr_CurrentAssets" />
   <calculationArc xlink:type="arc"
   xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
   xlink:from="fr_Assets" xlink:to="fr_CurrentAssets"
   order="1"
   weight="1"
   use="optional" />
```

Presentation Linkbase

- Presentation linkbase stores relations between concepts for the proper presentation and this structure shows similarities with PDM.
- All necessary concepts (items and tuples) are shown in presentation linkbase
- Necessary concept is a needed concept for financial filling process.
- If applicable, a company has to fill all necessary concepts shown in presentation linkbase. In other words, a concept shown in presentation linkbase is not always applicable for each company.
- The same concept can be shown more than once in presentation linkbase.
- For each report there is only one end (root) concept (item or tuple) in presentation linkbase.
- A concept which is not shown in presentation linkbase can have a value in instance document however the rule by governmental parties (e.g. Tax Office) indicates that only the concepts which are shown in presentation linkbase may have values in instance level.
- As a general rule in presentation linkbase either instant or durational items are used together in places, not to cause inconsistency between period types.
- Each of these five linkbases shows the relations between concepts in a hierarchical order however presentation linkbase is the only one which shows all the necessary concepts in a given report. In other words, it shows the hierarchical relations between all necessary concepts needed for financial filling process.

An xml clip for the presentation linkbase is given below. In this example, first the locators for “fr_Assests” and “fr_FixedAssets” are defined. Then the presentation arc is defined by pointing out “from” and “to” locators. This means that “fr_Assests” must be presented as the parent of “fr_FixedAssets”.

```
<presentationLink xlink:type="extended"
xlink:role="http://www.xbrl.org/2003/role/link">
  <loc xlink:type="locator"
    xlink:label="fr_FixedAssets"
    xlink:href="Financial%20Report.xsd#fr_FixedAssets" />
   <presentationArc xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="fr_Assets" xlink:to="fr_FixedAssets"
    order="2"
    use="optional" />
   <loc xlink:type="locator"
    xlink:label="fr_Assets"
    xlink:href="Financial%20Report.xsd#fr_Assets" />
```

Definition Linkbase

- The definition linkbase provides taxonomy developers with a means of define additional relationships between concepts.
- The main function of this linkbase is to define the dimensional constructs in taxonomy. In other words the presentation linkbase is used to express non-dimensional information whereas the definition linkbase is used to express dimensional information (e.g. hypercube).
- The function of the definition linkbase is relevant when generating a PDM. However the definition linkbase is not relevant for the Dutch taxonomy because tuples are used as the replacement of dimensions for associating concepts.

Dimensions

- The dimensional constructs are defined in the definition linkbase. This construct gives more flexibility to the semantics of the XBRL standard.
- The visualization of dimensions is different from other concepts. Therefore an example is given in order to illustrate the dimensional constructs and how they are structured in the definition linkbase.
- For instance, a reporting Sales value can be a dimensional construct. There can be one dimensions defined as products (e.g. pharmaceutical) and regions (e.g. Europe). With these dimensions a cross-table can be built and each cell in this table refers to the corresponding Sales value. More than two dimensions can be defined (three dimensions refer to a hypercube) and there is no upper bound for the quantity of dimensions used in a report. A visualization of the dimensional construct is given for this example as below

Element	arcrole
Definition Link	
http://www.xbrl.org/2003/role/link	
Sales	
Company Hypercube	all (*)
By Region (Placeholder)	hypercube-dimension (*)
All Regions	dimension-domain (*)
Europe Region	domain-member (*)
By Product (Placeholder)	hypercube-dimension (*)
All Products	dimension-domain (*)
Pharmaceuticals Segment	domain-member (*)

Dimensional Construct

- Four types of concepts are used to model dimensions.
- First, primary items are concepts which are reporting (e.g. Sale concept).
- Second, hypercube refers to set of dimensions (set of axis providing segmentation)
- Third, dimension refers to axis (e.g. X,Y axis) of the hypercube (e.g. product, region)
- Fourth, domain member refers to a single value under each dimension (e.g. Pharmaceutical segment, Europe Region)
- The structure of a dimensional construct is different from items and tuples. In the instance level, now the facts for the corresponding domain members are not meaningful (e.g. 1000€ for Europe Region). Only the primary item (e.g. Sale) is the meaningful construct which is reported and will get a value in instance level (e.g. 1000€ for Sales in Europe Region). Other constructs defined are for visualization purposes.
- One can create as many dimensions and domains members as possible.
- The dimensional constructs can be complicated depending on how many dimensions and domain members are used. And they are structured constructs comparing to tuples.
- The Dutch taxonomy is currently using tuples as a replacement of dimensional constructs; however in future this situation can be changed and the dimensional constructs can start using in the Dutch taxonomy.

An xml clip of the example for the definition linkbase is given below. In this example, primary item, hypbercube, dimensions and domains members are first defined by locators and their relations are shown by arcs.

```
<definitionLink xlink:type="extended" xlink:role="http://www.xbrl.org/2003/role/link">

   <loc xlink:type="locator" xlink:href="Company.xsd#company_Sales"
xlink:label="company_Sales"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_CompanyHypercube"
xlink:label="company_CompanyHypercube"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/all"
xlink:from="company_Sales" xlink:to="company_CompanyHypercube" use="optional" order="1.0"
xbrldt:targetRole="http://www.xbrl.org/2003/role/link" xbrldt:contextElement="scenario"
xbrldta:summable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_ByRegionPlaceholder"
```

```
xlink:label="company_ByRegionPlaceholder"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension" xlink:from="company_CompanyHypercube" xlink:to="company_ByRegionPlaceholder" use="optional" order="1.0" xbrldta:summable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_AllRegions" xlink:label="company_AllRegions"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/dimension-domain" xlink:from="company_ByRegionPlaceholder" xlink:to="company_AllRegions" use="optional" order="1.0" xbrldt:targetRole="http://www.xbrl.org/2003/role/link" xbrldt:usable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_EuropeRegion" xlink:label="company_EuropeRegion"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/domain-member" xlink:from="company_AllRegions" xlink:to="company_EuropeRegion" use="optional" order="2.0" xbrldt:usable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_ByProductPlaceholder" xlink:label="company_ByProductPlaceholder"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension" xlink:from="company_CompanyHypercube" xlink:to="company_ByProductPlaceholder" use="optional" order="2.0" xbrldta:summable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_AllProducts" xlink:label="company_AllProducts"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/dimension-domain" xlink:from="company_ByProductPlaceholder" xlink:to="company_AllProducts" use="optional" order="1.0" xbrldt:targetRole="http://www.xbrl.org/2003/role/link" xbrldt:usable="true"/>

   <loc xlink:type="locator" xlink:href="Company.xsd#company_PharmaceuticalsSegment" xlink:label="company_PharmaceuticalsSegment"/>

   <definitionArc xlink:type="arc" xlink:arcrole="http://xbrl.org/int/dim/arcrole/domain-member" xlink:from="company_AllProducts" xlink:to="company_PharmaceuticalsSegment" use="optional" order="1.0" xbrldt:usable="true"/>

  </definitionLink>
```

Instance Level

Facts

- The term facts refer to concepts in instance level.

- A fact is a concrete value for a data item and these values are contained in the instance level. For each fact, context, period and unit properties have to be defined.
- Abstract elements are for visualization purposes and not appeared in the instance file. So they neither have an instance nor values in the instance level.
- A tuple itself has an instance in the instance file but does not contain a value by itself.
- All constructs (besides abstract ones) may (not) have values in the instance level. In other words some concepts may not appear in the instance file even though they are not abstract.

An xml clip of the example for a fact is given below. In this example, a fact “Net Income (fr:netIncome)” with the value of 1083000. The attribute of this fact is defined in a grouped set with the id of “Current_AsOf”. The node of < numericContext > specifies the parameters used on the fact “Net Income”, i.e. the entry, period, and unit property.

```
< fr:netIncome numericContext='Current_AsOf'>1083000</fr:netIncome >
<numericContext id='Current_AsOf' precision='18' cwa='true'>
<entity>
<identifier scheme='http://www.sampleCompany.com'>Sample Company</identifier>
</entity>
<period>
<instant>2002-12-31</instant>
</period>
<unit>
<measure>iso4217:EUR</measure>
</unit>
</numericContext>
```

Units

- The unit specifies the units in which a numeric item has been measured. Therefore unit is only relevant for a numerical concept.
- The content of the unit can be either a simple or a complex type. Some examples of simple units of measure are EUR (Euros), meters, kilograms and FTE (Full Time Equivalents). Some examples of complex units of measures are Earnings per Share and Square Feet. Only one unit can be defined for each fact.

In the xml clip above, a unit attribute is defined for the fact “Net Income (fr:netIncome)” in terms of Euros.

Context

- For each fact, at least one context has to be defined in instance level. And only one value can be given under the same context (if four contexts are defined for an item, it means this item has four different values).
- A context has entity and period properties. Entity is the identifier of cooperation (e.g. Deloitte); a period can be instant or duration.
- A context has ID
- A context has period type information (either instant or duration) which is different from other contexts.
- Contexts can be differentiated from each other by these properties.
- A context has the identifier information which can be same for other contexts.
- A context has either instant or duration indication as a period type.
- Instant or duration period type information in a context specifies the time in which a data is valid.
- One can define as many contexts as possible for an instant value
- One can define as many contexts as possible for a durational value
- For tuples, multiple usage of the same tuple (with the same items inside) in instance level is arbitrary and depends on how a company structures its financial processes.
- In other words the same tuple can be used more than once in instance level.
- An item under the same tuple can have more than one value for the same context.

In the xml clip above, an instance period type is defined for “2002-12-31”. The same xml clip but for a durational period type is shown below

```
< fr:netIncome numericContext='Current_AsOf'>1083000</fr:netIncome >
<numericContext id='Current_AsOf' precision='18' cwa='true'>
<entity>
<identifier scheme='http://www.sampleCompany.com'>Sample Company</identifier>
</entity>
<period>
    <startDate>2009-06-01</startDate>
    <endDate>2010-05-31</endDate>
</period>
<unit>
<measure>iso4217:EUR</measure>
</unit>
</numericContext>
```

Footnotes

- To express these irregular linkages, XBRL uses the footnote construct to describe these irregularly structured associations between facts in an XBRL instance.
- It can be used as comments or notations that refer to one or more fact values.
- Use of the footnote construct is not relevant for a PDM since the term is used for associating text annotations with particular facts.

An xml clip for the footnote construct is shown below. First a locator for the fact “Assets” is created. Then the element footnote contains the text of a footnote and the footnoteLink connects the element with this reference.

```
<Assets id="Assets"
 decimals="0" contextRef="Current_AsOf" unitRef="GBP">20000</Assets>

<link:loc xlink:type="locator" xlink:href="#Assets" xlink:label="Assets"/>

<link:footnoteArc xlink:type="arc"
 xlink:arcrole="http://www.xbrl.org/2003/arcrole/fact-footnote"
 xlink:from="Assets" xlink:to="AssetsFootnote" order="1.0"/>

<link:footnote xlink:type="resource" xlink:label="AssetsFootnote"
 xlink:role="http://www.xbrl.org/2003/role/footnote"
 xml:lang="en">For more information see Disclosures on Assets</link:footnote>
```

Appendix C- Other Proposals for Multiple Instance Problem

Besides extending the PDM with the “range” function, other proposals are given for the MI problem as well.

Proposal 1

Impact on Formal Definition: The formal definition of PDM is extended on data elements by differentiating single and multiple data elements types.

Description: By adding a choice construct to the current definition of data elements, one can distinguish single and multiple instances in the current set of data elements. “SingleData” element inherits the same attributes (“DataElementID” and “Description”) as the data elements from the original definition. “MultipleData” element possesses the same attributes as “SingleData” element and the “minVal”, “maxVal” attributes additionally.

Focus: Data Elements

Proposal 2

Impact on Formal Definition: The formal definition of operations is extended by adding the “minVal” and “maxVal” attributes.

Description: Extend the definition of operations by adding the “minVal” and “maxVal” conditions. These conditions are with integer type and required for each operation. If a value “1” is assigned both, this indicates the single execution of that operation. In the “minVal” is equal to “maxVal” condition, it refers to pattern 1. In other words if “minVal” is greater than a value “1”, it refers to a fixed number of instances with a priori design time knowledge. If the “maxVal” is greater than “minVal” condition, this refers to pattern 2.

Focus: Operations

Proposal 3

Impact on Formal Definition: The formal definition of PDM is extended on operations by adding the “minVal” and “maxVal” conditions and the MI attribute is added on the data elements.

Description: Extend the definition of data elements by adding the MI attribute. And extend the definition of operations by adding the “minVal” and “maxVal” conditions. This proposal is a combination of two approaches. For each data element, the MI attribute is with Boolean type, and the use is required. For each operation the “minVal” and “maxVal” conditions are added. These conditions are with integer type and the use is required. If MI attribute is “False” for any data element then both the “minVal” and “maxVal” conditions have a value “1” at the incoming arc(s) of that data element. If MI attribute is “True” for any data element then the “maxVal” should be greater than or equal to “minVal” condition at the incoming arc(s) of that data element. In the “minVal” is equal to “maxVal” condition, it refers to pattern 1 (Multiple Instances with Priori Design Time Knowledge) whereas the “maxVal” is greater than “minVal” condition refers to pattern 2 (Multiple Instances without Priori Run Time Knowledge)

Focus: Data Elements and Operations

Appendix D- The Mapping between XBRL and PDM for "Content related" Rules

As a result of the analysis of the mapping between XBRL and PDM for the multiple instances issue and the financial filing rules issue, also a number of general shortcomings of the PDM were identified as 'additional benefits' from the "identification" phase. To explain these additional requirements, a mapping has been made by taking the simplified Dutch taxonomy structure into account. It starts with the data level and followed by the relations and the instance level information accordingly. In other words, this mapping can be used to specify the relations between elements in the Dutch taxonomy ("content-related" rules of the financial filing process).

	XBRL Constructs	PDM Constructs	Description	Framework	Extensibility
Data Level	An item concept	A data element	-	-	✓
	Each item is unique and has ID	Each data element is unique and has ID	-	-	✓
	An item has a data type attribute (e.g. integer, string)	A data type attribute is not specified	Data type can be added as an attribute to the data element in the PDM	**PDM**	**X**
	An item has a balance attribute (plus or minus sign)	A balance attribute is not specified	Balance attribute can be mapped in the PDM instance (actual PDM design)	**PDM**	**X**
	An item has a period attribute (instance or durational type)	A period attribute is not specified in PDM	Period attribute can be mapped in the PDM instance (actual PDM design)	**PDM**	**X**
	An item (abstract) concept	A data element	A data element in the PDM can be mapped with an abstract item	-	✓
	A tuple concept	A data Element	A data element in the PDM can be mapped with a tuple	-	✓
Relation Level	Hierarchy in the presentation linkbase	Starting a PDM model	Presentation linkbase has tree-like structure and	-	✓

			shows necessary data elements for the financial filing		
	Relations between the concepts in the presentation linkbase (mentioned as arcs in the XML)	Operations in a PDM	-	-	✓
	Relations between the same concepts (same input and output) can be occurred more than once in the presentation linkbase	Operations are unique in PDM	In the presentation linkbase, the same relation can be shown more than once (however there is only one occurrence in the instance file).	**XBRL**	**X**
	For each concept shown in the presentation linkbase, it is not necessary to have a corresponding value (and instance) in the instance file	Each data element has an instance but not necessary to have a value. Also instance level information is not recorded yet (PDM instance file)	In XBRL, if a concept does not have a value (and instance) then it is not necessary for financial filing. Those ones should first be filtered out In PDM, instance level information should be recorded	**XBRL**	**X**
	Each item can be shown more than once in the presentation linkbase	Each data element is shown only once but can be used more than once	Data elements in PDM can be used several times	-	✓
	As a general rule in the presentation linkbase either instant or durational items	Data elements are used in places	Data elements are not causing inconsistency in each other	-	✓

	are used together in places, not to cause inconsistency between period types				
	In the presentation linkbase, there might be more than one root (end) item however they are connected with linkroles and only one root item in the Dutch taxonomy reports	One root element	-	-	✓
Instance Level	An abstract item does not have an value (and an instance) in the instance file	Each data element has an instance but not necessary to have a value	Abstract and concrete items have to be differentiated in the PDM design. In the PDM instance, each data element does not necessarily need to have a value	**PDM**	**X**
	A tuple has an instance but does not have a value by itself	Each data element can just have an instance but not necessary to have a value	In the PDM instance, it is not necessary for a data element to have a value	-	✓
	A context characteristic (There is only one value corresponding to each context)	Multiple instances of data elements	A PDM should handle the multiple occurrences of the same data element	**PDM**	**X**

As can be seen from the table above, the constructs between XBRL and PDM are shown similarities as well as differences between each other. Some extensions can be directly applied to the PDM (e.g. data type attributes); however some (e.g. multiple occurrences of operations in XBRL) may need additional research. As a reflection of this mapping, the structure ("content-related" rules) of the constructs in the Dutch taxonomy has been clarified and additional research directions are proposed in the limitations and future work seciton.

Appendix E- Extended XML Schema Definition for a PDM

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2011 rel. 2 sp1 (http://www.altova.com) by a (a) -->
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by TM (Technische Universiteit Eindhoven)
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="PDM">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Name"/>
				<xs:element ref="Status" minOccurs="0" maxOccurs="unbounded"/>
				<xs:element ref="Department" maxOccurs="unbounded"/>
				<xs:element ref="DataElement" maxOccurs="unbounded"/>
				<xs:element ref="Roles" maxOccurs="unbounded"/>
				<xs:element ref="Resource" maxOccurs="unbounded"/>
				<xs:element ref="Activities" minOccurs="0"
maxOccurs="unbounded"/>
				<xs:element ref="Operation" maxOccurs="unbounded"/>
				<xs:element ref="RootElementRef"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Status">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:enumeration value="Created"/>
				<xs:enumeration value="Reviewed"/>
				<xs:enumeration value="Read by 1st"/>
				<xs:enumeration value="Statement Made"/>
				<xs:enumeration value="Read by 2nd"/>
				<xs:enumeration value="Signed-off-1"/>
				<xs:enumeration value="Signed-off-2"/>
				<xs:enumeration value="Archieved"/>
			</xs:restriction>
		</xs:simpleType>
	</xs:element>
	<xs:element name="Department">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:enumeration value="Audit"/>
				<xs:enumeration value="Tax"/>
				<xs:enumeration value="Consultancy"/>
			</xs:restriction>
		</xs:simpleType>
	</xs:element>
	<xs:element name="DataElement">
		<xs:complexType>
			<xs:sequence minOccurs="0">
				<xs:element ref="Data" minOccurs="0"/>
			</xs:sequence>
			<xs:attribute name="minVal" type="xs:integer" use="optional" default="1"/>
			<xs:attribute name="maxVal" use="optional">
				<xs:simpleType>
					<xs:restriction base="xs:integer">
						<xs:minInclusive value="1"/>
						<xs:maxInclusive value="100"/>
```

```
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:attribute>
                                <xs:attribute name="DataElementID" type="xs:ID" use="required"/>
                                <xs:attribute name="Department" type="xs:string" use="optional"/>
                                <xs:attribute name="Description" type="xs:string" use="optional"/>
                                <xs:attribute name="Status" type="xs:string" use="optional"/>
                        </xs:complexType>
                </xs:element>
                <xs:element name="Roles">
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="Creator"/>
                                        <xs:enumeration value="Reviewer"/>
                                        <xs:enumeration value="1st Reader"/>
                                        <xs:enumeration value="2nd Reader"/>
                                        <xs:enumeration value="Signed-off-Deloitte"/>
                                        <xs:enumeration value="Signed-off-Client"/>
                                        <xs:enumeration value="Archival"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:element>
                <xs:element name="Resource">
                        <xs:complexType>
                                <xs:attribute name="ResourceID" type="xs:ID" use="required"/>
                                <xs:attribute name="Roles" type="xs:string" use="required"/>
                                <xs:attribute name="Job-grade">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:integer">
                                                        <xs:minInclusive value="0"/>
                                                        <xs:maxInclusive value="14"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:attribute>
                                <xs:attribute name="Department" type="xs:string" use="optional"/>
                                <xs:attribute name="Description" type="xs:string" use="optional"/>
                        </xs:complexType>
                </xs:element>
                <xs:element name="Activities">
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="Create"/>
                                        <xs:enumeration value="Review"/>
                                        <xs:enumeration value="Read by 1st"/>
                                        <xs:enumeration value="Made Statement"/>
                                        <xs:enumeration value="Read by 2nd"/>
                                        <xs:enumeration value="Sign-off-1"/>
                                        <xs:enumeration value="Sign-off-2"/>
                                        <xs:enumeration value="Archieve"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:element>
                <xs:element name="Operation">
                        <xs:complexType>
                                <xs:sequence>
                                        <xs:element name="Input">
                                                <xs:complexType>
                                                        <xs:sequence>
                                                                <xs:element name="DataElementRef"
type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
```

```
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="Output">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="DataElementRef"
type="xs:IDREF"/>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="Cost" minOccurs="0"/>
                                <xs:element name="Time" minOccurs="0"/>
                                <xs:element name="Condition" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element name="Probability" minOccurs="0"/>
                                <xs:element name="ResourceRef" type="xs:IDREF"
minOccurs="0"/>
                                <xs:element name="ActivitiesRef" type="xs:IDREF" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element ref="Data" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attribute name="OperationID" type="xs:ID" use="required"/>
                            <xs:attribute name="Descriptioin" type="xs:string" use="optional"/>
                        </xs:complexType>
                </xs:element>
                <xs:element name="Data"/>
                <xs:element name="RootElementRef" type="xs:IDREF"/>
                <xs:element name="Name"/>
</xs:schema>
```

Appendix F- Demonstration Tools

Many XBRL tools, e.g. UBmatrix, Altova MissionKit, and Fujitsu, are available in the market to support the XBRL taxonomy design, XBRL instance file creation, and XBRL validation. For this project, we used XWand Taxonomy Designer from Fujitsu because the starting point of the research framework is to create XBRL taxonomy; in addition, XWand XBRL product is a licensed tool and commonly used within Deloitte for XBRL projects. Then XMLSpy is used to create the XSLT file for XBRL-PDM conversion. The output of the XSLT file is used in ProM to graphically depict the PDM structure.

XWand Tool

Fujitsu XWand[1] offers comprehensive and complete software to cater for all stages of eXtensible Business Reporting Language (XBRL) implementations. It covers the full lifecycle of XBRL development, such as taxonomy design, reporting preparer, reports and procedural audit, processing engine and deployment of XBRL applications. XWand tool is a desktop application designed for building, extending and maintaining XBRL taxonomies. In this project, XWand is used to create sample Dutch taxonomy which is later used as an input for the XBRL to PDM transformation.

Altova XMLSpy

Altova XMLSpy is an XML editor and development environment for modeling, editing, transforming, and debugging XML-related technologies. It offers a graphical schema designer, a code generator, file converters, debuggers, profilers, full database integration, and support for XSLT, XPath, XQuery, WSDL, SOAP, XBRL, and Office Open XML (OOXML) documents, plus Visual Studio and Eclipse plug-ins, and more. Recently added support for chart creation in XMLSpy brings a whole new dimension to working with - and analyzing - XML data. In this project, XMLSpy is used for the implementation of constructs in the PDM schema. XSLT function in XMLSPY is also used to transform the Dutch taxonomy (xml) to PDM (xml). XSLT file is prepared by performing a tag matching between the XBRL and PDM constructs.

ProM

ProM[2] is a free process mining tool developed by Eindhoven University of Technology. It is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins. There are two types of plug-ins in ProM, i.e. mining plug-ins, and analysis plug-ins. In addition, ProM is able to import process models constructed in many languages, e.g. PDM, Petri Nets, EPC,

[1] http://www.fujitsu.com/global/services/software/interstage/solutions/xbrl/
[2] http://prom.win.tue.nl/tools/prom/.

BPMN, YAWL, and etc. In this project, the output of XSLT file is used as an input for ProM to visualize the PDM structure.

Appendix G- Description of the XSLT Code

We first divide the XSLT code into three parts and each part corresponds to the different construct (data elements, operations and root element) in the PDM.

In the first part, data elements of the PDM are matched with the “link:loc” tags. We use <xsl:apply-templates> with a select attribute indicating “link:loc” tag. Therefore it will only process the child element that matches the value of the attribute. When applying the template match, <xsl:value of> element is used for extracting the data element information from “xlink:label”.

In the second part, operations of the PDM are matched with the constructs in XBRL. As can be understood, there is a link between each elements and direct tag matching (“xlink:from” for “output” and “xlink:to” for “input”) will not work. This is because the information from all child elements is needed together in order to make a decision upon the parent element. Therefore a different algorithm for this translation is considered. Since every element in PDM is once an output element, <xsl:value of> is used for extracting the information from “xlink:label” for the output elements. For the input elements, first <xsl:apply-templates> is used with a select attribute “xlink:label”. This will process the records that match the value of the attribute. Then the condition of “xlink:label” equals to “xlink:from” is tested where returns the parent elements in PDM. If the condition is satisfied, “xlink:to” information is extracted as input elements in the PDM.

In the third part, the root element of the PDM is found in the presentation linkbase. The root element is not referred by any other element in XBRL, therefore it cannot be seen in the “xlink:to” tag. First a list (“list” in XSLT) is created by considering the “xlink:to” information in the presentation linkbase. As can be understood, “xlink:to” information refers to the child elements. Then < xsl:key> function is used to match the “xlink:to” with the “xlink:label” information in the presentation linkbase. Then the “count” function counts the number of “xlink:to” in the given list. When this number is zero, meaning this element is not a child of any other elements, it returns the root element of the PDM.

For multiple instances, an attribute is defined as “minVal” and a default value “1” is set. In the same manner, a “maxVal” attribute is defined and a default value “100” is given. In order to capture the department information, first a list (“department” in XSLT) is created by considering the “xlink:from” information in the definition linkbase. As can be understood, “xlink:from” information refers to the data elements. Then <xsl:key> function is used to match the “xlink:label” information in the presentation linkbase with the “xlink:from” information in the definition linkbase. When there is a match, the function returns the “xlink:to” information from the definition linkbase which will show the responsible party for each element.

Appendix H- XSLT File

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
        xmlns:link="http://www.xbrl.org/2003/linkbase"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:output method="xml" encoding="UTF-8" indent="yes"/>

        <xsl:key name="list" match="/link:linkbase/link:presentationLink/link:presentationArc"
use="@xlink:to" />
   <xsl:key name="department" match="/link:linkbase/link:definitionLink/link:definitionArc"
use="@xlink:from" />

        <xsl:template match="link:linkbase">

                <PDM xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="pdm.xsd" >

                <Name>My sample Dutch Taxonomy</Name>


                <xsl:apply-templates select="link:presentationLink/link:loc" >
                        <xsl:with-param name="part" select="1" />
                </xsl:apply-templates>

                <xsl:apply-templates select="link:presentationLink/link:loc" >
                        <xsl:with-param name="part" select="2" />
                </xsl:apply-templates>

                <xsl:apply-templates select="link:presentationLink/link:loc" >
                        <xsl:with-param name="part" select="3" />
                </xsl:apply-templates>

                </PDM>
        </xsl:template>

        <xsl:template match="link:presentationLink/link:loc">
                <xsl:param name="part"/>

                <xsl:choose>
                        <xsl:when test="$part = 1" >
                                <DataElement>
                                        <xsl:attribute name="DataElementID"> <xsl:value-of
select="@xlink:label"/> </xsl:attribute>
                                        <xsl:attribute name="Department"> <xsl:value-of
select="key('department',@xlink:label)/@xlink:to" /> </xsl:attribute>
                                        <Range>1-100</Range>
                                </DataElement>
                        </xsl:when>

                        <xsl:when test="$part = 2" >
                                <Operation>
                                        <xsl:attribute name="OperationID"> <xsl:value-of
select="generate-id()"/> </xsl:attribute>


                                <!-- input velden -->
```

```
                                        <Input>
                                                <xsl:apply-templates
select="/link:linkbase/link:presentationLink/link:presentationArc" >
                                                        <xsl:with-param name="parent"
select="@xlink:label" />
                                                </xsl:apply-templates>
                                        </Input>
                                <!-- output veld -->

                                        <Output>
                                                <DataElementRef><xsl:value-of
select="@xlink:label" /></DataElementRef>
                                        </Output>
                                <!-- resource veld -->

                                        <ResourceRef>W1</ResourceRef>

                                </Operation>
                        </xsl:when>
                        <xsl:when test="$part = 3" >
                                <xsl:if test="count(key('list',@xlink:label)) = 0" >
                                        <RootElementRef>
                                                <xsl:value-of select="@xlink:label" />
                                        </RootElementRef>
                                </xsl:if>
                        </xsl:when>
                </xsl:choose>
        </xsl:template>

        <xsl:template match="/link:linkbase/link:presentationLink/link:presentationArc" >
                <xsl:param name="parent" />

<!--            <xx>
                        <xsl:value-of select="$parent" />
                        <xsl:value-of select="@xlink:title" />
                </xx>
-->
                <xsl:choose>
                        <xsl:when test="@xlink:from = $parent" >
                                <DataElementRef><xsl:value-of
select="@xlink:to"/></DataElementRef>
<!--                                    <Ref><xsl:value-of select="@xlink:title"/></Ref>  -->
                        </xsl:when>
                </xsl:choose>
        </xsl:template>




</xsl:stylesheet>
```

Appendix I- Sample

I.1. Presentation and Definition Linkbase (xml) – see CD

I.2. XSLT Code—see CD

I.3. Generated PDM Example—see CD

I.4. Extended PDM Schema—see CD

Appendix J- Description of the Transformation Procedure

In order to repeat this process, first a sample should be created in the presentation linkbase in XBRL. After creating the sample, we open it in XMLSPY. By clicking the “XSL/XQuery” button from the toolbar, and selecting “XSL Transformation”, we can specify the location of our XSLT file. After executing the XSLT on our sample, the xml output is automatically generated. One can use this file in ProM to visualize the PDM structure. In order to do this, you should find “Open PDM file” under the “File” section in toolbar and select the automatically generated xml output from your XMLSPY. Then the PDM file can be seen. In the figure below, the screenshot of the XMLSPY is given in order to facilitate the understandability of the transformation procedure.