How to use XBRL in workflow management systems?

Sun, Y.

Award date:
2010

TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Mathematics and Computer Science

Master Thesis

How to use XBRL in Workflow Management Systems?

Yaqing Sun

Supervisors

Dave van den Ende (Deloitte Innovation)
Boudewijn van Dongen (TU/e)
Irene Vanderfeesten (TU/e)

Eindhoven, November 2010

Acknowledgement

This master thesis is the outcome of the graduation project for my master study in the Business Information Systems Program of the Eindhoven University of Technology (TU/e). This master project is carried out in Deloitte Innovation B.V. and the Architecture of Information Systems (AIS) group of the TU/e. Deloitte Innovation has provided an excellent professional environment for me to explore business opportunities of workflow management systems in practice, while TU/e prepared and supported me with solid theoretical background for this research. I'd like to thank several people for their support during my master study and my master project.

First of all, I'd like to thank my supervisors, Dave van den Ende, Boudewijn van Dongen and Irene Vanderfeesten. It has been a very nice experience to work under your supervision. Dave, thanks for being patient in explaining XBRL to me and helping me clear up so much vagueness during the project. Thank you for funding me with the great opportunity to attend the 21st XBRL International Conference at Beijing, China. It was a really nice experience for me to gain new thoughts about XBRL. Boudewijn, thanks for supporting me when I made an extremely tight schedule after my mid-term presentation. Your valuable opinions always guided and motivated me into the right direction during the whole project. And Irene, I really appreciate your critical feedbacks on my thesis. I like the way we exchanged our opinions on PDM. I learned a lot about PDM by discussing with you. The research on converting algorithms for PDM was a good learning process for me. It has brought many new thoughts to me, regarding workflow management both in theory and in practice.

I'd also like to thank Bas and Paul from Deloitte, who generously helped me with their XBRL knowledge during this project. Moe Thandar Wynn from Queensland University of Technology, Australia, thank you for providing me with the YAWL logging application. Arthur ter Hofstede, also from Queensland University of Technology, thank you for sharing your knowledge with me about the progress of BPM researches in the world, especially in China.

Many thanks to all my friends here in the Netherlands, Vanessa, Stefania, Amalia, Fernando, Cristian, Cem, Zhao Rui, Sun Tong, Seung-yub, Yin Si, Latif, Oziel, Anton, Ehsan, Chu, Andreea and Jorn, and two of my best friends back in China, Yaoyi and Li Yan. Thank you for sharing every little detail with me about my master project and my master study. You have made my life colorful.

Finally, I'd like to thank my family, especially my parents and my grandparents, for always keeping your faith in me and encouraging me when I was down. You made me understand what love really means. Thanks to Chih-wei, Wei for being part of my life in the last two years. And special thanks to Chris Chang for trusting me and bringing new hopes in my life. You have the ability to sweep away my worries and make me feel confident in everything I do.

Love and peace,

Yaqing Sun

November 2010

Summary

XBRL (eXtensible Business Reporting Language) is an open standard business reporting language that facilitates the transmission of financial information. For years, Deloitte has been promoting the use of XBRL among its clients for the purpose of reducing financial costs and human input errors as well as improving data integration within organizations. In this master project, we investigate the possibilities to facilitate the XBRL based financial filing processes by extending XBRL with workflow management information as to assure the data validity, and thus to reuse available data for multiple purposes.

Two problems are identified in the XBRL based financial filing process within Deloitte, i.e. data assurance and data reuse. Data assurance is to assure the validity of data, i.e. data correctness (using right inputs and right formulas) and data audit trail (created by the right person at the right time). As long as data validity is assured before hand, available data can be reused without authority issues for other reports. In order to standardize these two problems, we defined a sample set of financial filing rules that can be used to guide financial filing processes.

In the first part of the thesis, we started by investigating how workflow management systems can help to provide data assurance information and what workflow technique can be used to design the XBRL based financial filing process. Considering the two parts of data assurance, i.e. data correctness and data audit trail, we decided to pay more attention to the question of how to provide and store data audit trail information, since data correctness is more related to the job of taxonomy designers. In fact, with the logging function provided by most workflow management systems, data audit trail information can be easily obtained by retrieving process logs from the workflow engine. And we chose to use PDM (Product Data Model) as a design method for XBRL based financial filing rules because it is objective, flexible and can be automatically generated from XBRL taxonomy.

In the middle part of the thesis, we explored the possibilities to extend XBRL as well as PDM schema in order to store financial filing rules. We suggest create a Workflow Linkbase for XBRL using Generic Links Specification. And we propose to use separate files to include financial filing rules and complex organizational structure in PDM. Moreover, we propose to extend the PDM schema as to include sub processes for data elements that need special treatment.

In the last part of the thesis, we combine the solutions described above in a prototype as a proof of concept. The key step in creating the prototype is to find the best converting algorithm to translate PDM into an executable process model. We set a groups of requirements based on the requirements we need for XBRL financial filing processes along with the classification framework of the algorithms. By evaluating the algorithms against this set of requirements, we concluded that three algorithms, i.e. Charlie, Delta, and Echo are all suitable for our prototype. However, because of time limit, the prototype is built based on the process model converted from Echo.

In the end, we prove that the conceptual solutions we propose to solve data assurance and data reuse problems are possible to apply in practice. Though with limitations, such as using abstract rather than a real XBRL taxonomy, simplifying sub processes, ignoring wrong and missing data, we still think this master thesis shows a positive result of the possibility to facilitate XBRL based financial filing process using workflow management systems.

Table of Contents

List of Figures

List of Tables

1 Introduction

This thesis is the outcome of the graduation project in the Master Program of Business Information Systems of the Eindhoven University of Technology (TU/e). The master project is carried in Deloitte Innovation B.V. and the Architecture of Information Systems (AIS) group of the TU/e. Throughout this master project, we investigate the possibilities to facilitate the financial filing processes by extending XBRL (eXtensible Business Reporting Language) with Workflow Management information and to assure the quality and trustability of data created during these processes.

1.1 Deloitte

Deloitte is one of the professional financial consultancy companies in the world. It provides outstanding services including tax filing, auditing, consulting, and financial advisory to its clients. For many years, Deloitte has been a leader in encouraging innovative ideas and promoting new technologies in the financial industry. Deloitte Innovation B.V., where this master project is carried out, is a main part within Deloitte Netherlands that is responsible for such innovative activities.

1.2 eXtensible Business Reporting Language

eXtensible Business Reporting Language (XBRL) is an open standard business reporting language that facilitates the transmission of financial information by providing a common dictionary of uniquely identified tags that are applied to the individual elements of financial statements and the notes to financial statements (1) . It has a wide use in U.S., European and Asian countries for the purpose of financial data collecting and reporting as well as data consumption and analysis. It is widely used among companies, governments, banks and stock exchange centers. An international XBRL organization was formed in 1998 in order to support the need of building standardized XBRL language for digitizing business reports in accordance with accounting and banking regulations in each country and to promote the adoption of XBRL in different industries (2) .

Organizations that adapt to XBRL need two types of files, i.e. XBRL taxonomy and XBRL instance file, to generate a financial report. An XBRL taxonomy file defines concepts (reporting terms), while an XBRL instance file gives values to those concepts. Relationships between concepts can be linked by five XBRL Linkbases, namely Calculation Linkbase, Presentation Linkbase, Reference Linkbase, Lable Linkbase and Definition Linkbase.

The benefits of adapting XBRL include:

- It minimizes information asymmetry during financial filing process.
- It reduces human errors and associated costs in financial filing process.
- It increases transparency of financial filing process for credit checking.
- It improves the efficiency and effectiveness of financial filing process controls.
- It provides persistent connectivity between financial data elements.

- It enables dynamic financial filing in which different presentational layouts can be prepared for internal systems, third-party systems, trading partners, etc.

Unfortunately, one main disadvantage of XBRL is that it has no information about the validity of data. Hence it is very problematic to reuse available data. So in most cases, data reusing is rarely done in XBRL based financial filing process in practice.

1.3 Workflow Management Systems

A Workflow Management System is a generic software package that can be used to support the definition, management and execution of workflow processes. It abstracts the real work processes with the order of tasks that need to be done as to achieve predefined goals within an organization. The goal of workflow management is to control the flow of work such that the work is done at the right time by the proper person (3). A workflow management system contains information of three aspects, i.e. control flow, data elements and organizational structure.

Workflow Management Systems can be used in various kinds of industries and organizations, for example factories, financial organizations, governments and health care organizations. Many big companies, such as Deloitte, ING, and Philips have adapted to use Workflow Management Systems to control their daily business operations. Big software vendors, such as IBM, SAP, and TIBCO (current owner of Staffware) also see a great opportunity in this field and are providing professional products or services to support customers' workflow management need. Compared with other information systems, one important advantage of workflow management systems is that changes to the process, such as splitting or combining tasks, and rearranging resource classes are much easier (3) . Also, a good design of workflow process reduces operation cost and time, and improves quality of the output products or services.

Improvement on process efficiency has always been a key concern among companies that are using workflow management systems. As for Deloitte, who uses workflow systems to facilitate financial filing processes, it will be a great enhancement if its workflow management systems can be more automated and thus save time, costs and resources for its services. Moreover, since workflow management systems allow us to store audit trail information, we can see how data values were produced from what inputs, at what time, and by whom. Therefore, we propose to combine workflow management systems with XBRL to assure data quality and enable data reuse.

In order to solve the two main problems (described in the next sub-section), i.e. data assurance and data reuse, we decided to study more about the possibilities to connect workflow management systems with XBRL engine, in such a way that the XBRL based financial filing process will be faster and easier to manage.

1.4 Problem Description

The aim of this project is to investigate possibilities to extend XBRL with a workflow management system that facilitates XBRL based financial filing processes. Two of the main problems that are concerned by the XBRL adaptors are XBRL data assurance and XBRL data reusing. In order to standardize these two problems more specifically, a sample set of financial filing rules are defined and used in this project.

1.4.1 Data Assurance

One main issue is the assurance of data quality during the XBRL based financial filing. In general, quality of data means the validity of data, i.e. data should be created by the right person at the right time, using right inputs and right formulas. Taken from this understanding, data assurance consists of two parts, i.e. data correctness (using right inputs and right formulas) and data audit trail (created by the right person at the right time). In this project, more attentions will be given to the data audit trail part than the data correctness part, because data correctness is more related with how the XBRL taxonomy is designed by the financial filing experts.

The problem of how to assure data has become an urgent issue when companies and organizations adapt to XBRL. There are some solution proposals for XBRL data assurance, but no standard is set up among organizations yet. The following list shows some reasons of why data assurance is so important with XRBL based financial filing process (4) :

- XBRL is a digital format that permits a different consumption of data from traditional financial filing.
- Different consumption of data means different representations of the same data.
- If data is represented in any other ways by a third party, the process of representing the data need to be guided by a set of so called “Financial Filing Rules” to assure the quality of the represented data, especially the audit trail part.
- The current auditing standards provide no guidance on how to provide assurance on information in electronic format (audit approach).
- Stakeholders have no views yet about the opportunities of electronic financial filing and the necessity on assurance (internal and third party).

Currently, audit trail information on collecting and updating data values during the financial filing process is stored separately in workflow systems from XBRL instance file. In this case, in order to find data assurance information, e.g. what data is created at what time by which employee, one has to look into both the workflow system and XBRL processing engine. By linking XBRL and workflow management systems, it will become possible to include design level workflow information, such as organizational structure, resource allocation rules, and data collecting deadlines, in the XBRL taxonomy. As a result, the instance level of this audit trail information can be stored in the XBRL instance file as to assure data quality.

In addition, workflow systems can also guide data flow in financial filing process by interpreting information from XBRL taxonomy and by checking data availability in XBRL instance files. So again, it is no more necessary to look into both workflow management systems and XBRL process engine to find out the availability of a certain data value. This can help reduce the overall time spent in financial filing process.

1.4.2 Data Reusing

One of the main advantages to adapt XBRL is that it can improve the efficiency of financial filing process. During the financial filing process, it is common that one data element or document is needed multiple times for different stakeholders or different end products. For example, the document of Balance Sheet is needed when the final report is submitted to both the Chamber of Commerce and the Tax Office. Unlike the traditional way of financial filing, every document becomes digital when using XBRL, this means that data and documents are reusable after they are created. However, this also means that data representations have to be assured before data can be reused for other purposes. So another problem that needs to be solved in this project is to provide a solution that can reuse available data created during the financial filing process with quality assurance. This is also why we use workflow management system to facilitate financial filing process. If a workflow management system is used to guide the financial filing process, it is possible to include the data audit trail information to ensure data quality during XBRL based financial filing process.

1.4.3 Financial Filing Rules

In order to assure the quality of the data created, companies and organizations follow some sets of self-defined rules during the financial filing process. These financial filing rules are used to specify how data values are related between different financial reports and what procedures should be taken before a financial report is ready to be submitted to the client, its external investors, and financial departments.

There are two ways of using financial filing rules. One way is to consider the rules during the process. This can be done by specifying the rules in the process model, so that the process will be guided by the rules while being executed. Another way is to consider the rules after the process is finished. This can be done by performing a conformance check on the recorded process log compared with the original process model. This project will set its focus to use those rules during the financial filing process. Financial filing rules that will be used in this project can be classified into two categories, i.e. content-related rule and process-related rule. More details on financial filing rules can be found in Chapter 3.

1.5 Research goal

The goal of the project is to answer the following research question:

> How to enable **data assurance** and **data reuse** to control data gathering in XBRL based financial filing process, while respecting **financial filing rules**?

As explained in the previous sub-section, this research question has to be answered by satisfying the two requirements in this project, i.e. providing **data assurance** information and make available **data reusable** by following the **financial filing rules**.

In order to answer this main research questions, several sub questions are needed.

First of all, on the technology level, we need to know

1) **How can data quality be assured using workflow management systems?**

This question will be answered in Chapter 2, where the workflow management reference model is explained.

2) **What workflow technology is most suitable for XBRL based financial filing process?**

   This question will be answered in Chapter 2, where Product Data Model (PDM) is introduced.

Then, on the content level, we need to know what should be included in financial filing rules and where should these rules be stored.

3) **What financial filing rules can be specified in a financial filing process?**

   This question will be answered in Chapter 3, section 2.

4) **How to extend XBRL with workflow information?**

   This question will be answered in Chapter 3, section 3.

5) **How should PDM be extended to include all necessary information?**

   This question will also be answered in Chapter 3, section 4.

Finally, on the execution level, we need to answer the following questions in order to prove the concept.

6) **How to generate PDM from XBRL?**

   This question will be answered in Chapter 3, section 4.

7) **How to execute the process based on PDM?**

   This question will be answer in Chapter 4, where the PDM to Petri Nets or YAWL model conversion algorithms are explained and evaluated against chosen requirements.

8) **How should financial filing rules be applied during execution of financial filing process, using the process model generated from PDM?**

   This question will be answered with a detailed demonstration in Chapter 5.

1.6 Project Framework

Based on the questions listed in the previous section, we made a framework for this project (See Figure 1). The structure of this project is a cycle that starts from XBRL taxonomy and ends with XBRL instance file. It can be grouped into two levels, i.e. Design Level and Instance Level. On the Design Level, information on data structure and data relationships are defined, while on the Instance Level, data values are given based on the structure and relationships defined in the Design Level. To be more specific, the process will start with a given XBRL taxonomy file. Then the XRBL taxonomy will be converted to a workflow process model reserving data structure, organizational model and financial filing rules defined in the XBRL taxonomy. This step will be followed by the

execution of the workflow process model in a workflow engine. When the execution is finished, process logs that contain audit trail information for XBRL instance will be retrieved. Finally, XBRL engine will take needed information for the XBRL instance file and generate a financial report.

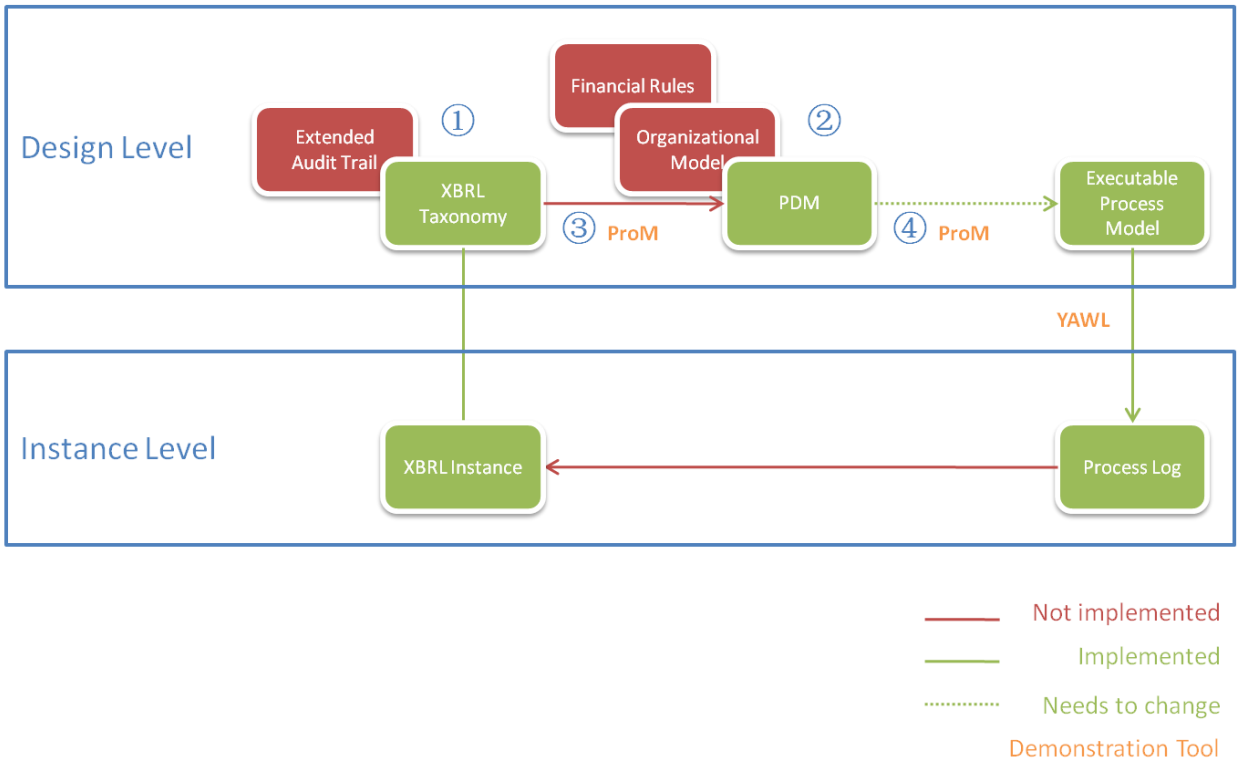


Figure 1 Project framework

In this project, the workflow process will be modeled using the concept of Product Data Model (PDM). As listed as one of the sub questions in this project, why to choose PDM is answered in Chapter 2. Once PDM is built, it will be converted to an executable process model, i.e. YAWL model in this case. And how to convert PDM to YAWL model will be answered in Chapter 4.

The following **challenges** have to be solved in order to make the complete circle shown above.

1. Extend XBRL taxonomy with design level workflow management information
    a. Create "Workflow Linkbase" specification
    b. Include financial filing rules in XBRL taxonomy using "Workflow Linkbase"

2. Extend PDM schema to reserve information that is stored in the XBRL taxonomy
    a. Extend PDM with organizational structure
    b. Extend PDM with financial filing rules
    c. Extend PDM with sub process of data elements

3. Convert XBRL taxonomy to a PDM
    a. Include XBRL taxonomy structure in PDM
    b. Include organizational structure in PDM

c. Include financial filing rules in PDM
d. Automate this step

4. Convert PDM model to an executable process model
    a. Make requirements to suggest algorithms to convert PDM
    b. Include organizational structure in the executable process model
    c. Include financial filing rules in the executable process model
    d. Revise the suggested algorithms to automate this step

Because of time limit, the whole circle cannot be complete during this master project. A prototype will be presented in Chapter 5 as a proof of concept showing it is possible to complete the circle in the project framework. In this prototype, Step 1, 2 are out of scope for this project, but solution proposals for these two steps can be found in section 3.3. Since PDM only contains data structure, only the first task in Step 3 is solved in section 3.4. The concepts of how to solve Step 4 can be found in Chapter 4. However the suggested algorithms were not adjusted to automate the last task in Step 4. More details on this prototype as well as reasons for important solution decisions can be found in Chapter 5.

1.7 Thesis outline

Chapter 2 of the thesis will explain more about Workflow Management Systems and the business case for choosing PDM as the product model in this project. Chapter 3 of the thesis will give a detailed explanation on XBRL. It will list a group of sample financial filing rules, and it will also discuss how to extend XBRL to suit the need for this project and how to convert XBRL to PDM. The result of chapter 4 will be a suggestion of all possible conversion algorithms (from PDM to an executable model) based on a set of requirements. Chapter 5 will describe a prototype made for this project. This prototype combines everything discussed in the previous chapters and adapts all financial filing rules defined in section 3.2. Chapter 6 will conclude the thesis, discuss the limitations of this project, and give recommendation for future studies.

2 Workflow Management System

Workflow management systems are needed to answer the main research question. It can be used to control data gathering and applying financial filing rules. This chapter will explain the first objective of this project, i.e. workflow management. The first sub section will introduce workflow management systems, a reference model of workflow management systems will be given in this sub section. The second sub section will explain workflow patterns. The third sub section will describe the XBRL based financial filing process and discuss requirements for workflow technique that can be used for XBRL based financial filing process. This will be followed by a detailed introduction of Product Data Model (PDM), which will be used as the design method in this project. This last sub section of this chapter will also discuss the advantages and disadvantages of PDM.

2.1 Introduction of Workflow Management Systems

Workflow management, also referred as business process management, is an approach that includes methods, techniques, and software to support the design, enactment, management, and analysis of operational processes involving humans, organizations, applications, documents and other sources of information (5). Key concepts in this definition are the workflow management lifecycle (design, enact, control and analyze) and the interaction between computer applications, information systems and human resources. The workflow management concept, which started as early as the second industrial revolution, has become a keen method for companies to improve the quality of their products and services and the efficiency of daily operations. A generic software package for managing workflow processes is called workflow management systems (WFMS). The following figure shows a reference model of Workflow Management System. This reference model is a description of the architecture of a workflow management system, in which the main components and the associated interfaces are summarized (3).

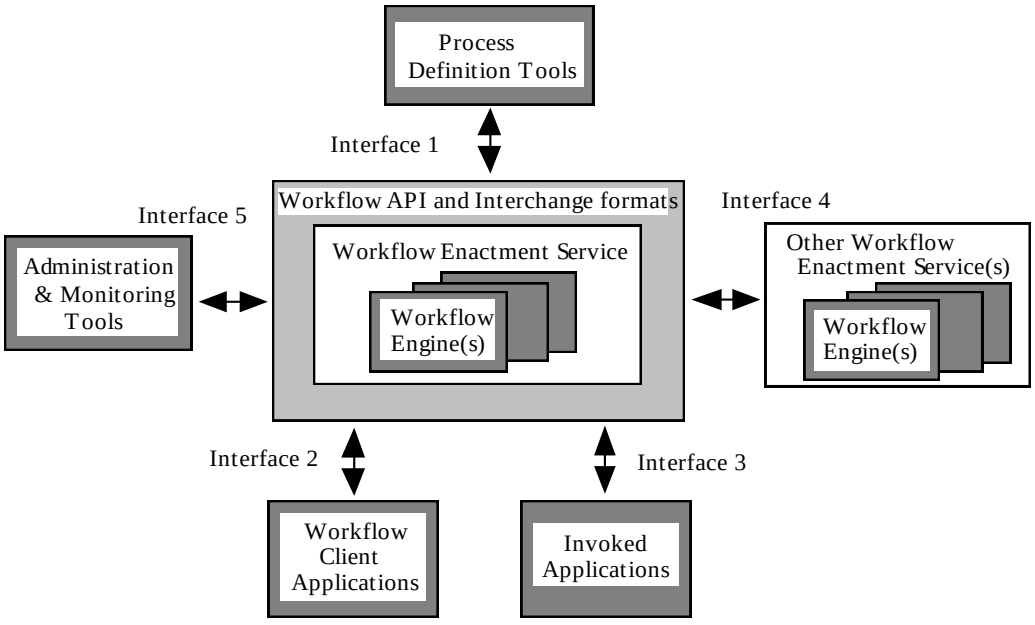


Figure 2 The Workflow Management Coalition's reference model (© WFMC).

The main part of a workflow system is the *workflow enactment service*. It ensures that the right activities are carried out in the right order and by the right people or right computer application. The five interfaces support this center service to work properly. *Process definition tools* define process control flow and resource structure. Work items are offered to the employees through *workflow client applications*. All the application software that can be started from the workflow system is known as *invoked applications*. Workflow tracking, case control, and staff management are supported by the so-called *administration and monitoring tools*.

Take this project for example, the Process Definition Tools (Interface 1) stores the organizational structure as well as the process control flow that is converted from XBRL taxonomy. The workflow enactment service engine assigns work items to corresponding resources via Workflow Client Applications (Interface 2). These resources will then invoke XBRL applications (Interface 3) to calculate needed financial filing values and indicate to the workflow engine when this work item is done. The process will continue this cycle until it reaches the end of the control flow model. All the performances are logged in the Administration and Monitoring Tools (Interface 5) for later use. For this project, Interface 5 is very important, because it answers the first sub question, i.e. how to ensure data quality during the financial filing process. To ensure the quality of the data, audit trail is needed to provide information on what was done at what hour by which resource. This information can be found in the Interface 5 of a workflow management system.

2.2 Workflow Patterns

A workflow process normally consists of three parts, i.e. control-flow part, organizational structure part, and data information part.

There are many languages available to design control flow models, for example, EPC (Event-Driven Process Chain), BPMN (Business Process Modeling Notation), and Petri Nets. However, no matter using what languages, tasks are often constructed following four basic routings (3). Definitions of these four routings are summarized in the table below.

Routing Name	Definition
Sequence Routing	When tasks have to be carried out one after another.
Parallel Routing	If more than one task can be carried out at the same time or in any order. These tasks can be initiated using an AND-split and later resynchronized using an AND-join.
Selective Routing	When there is a choice between two or more tasks that depend upon the specific properties of the case. These tasks can be notated with OR-join, OR-split, XOR-join, or XOR-split.
Iterative Routing	A repeated execution of a particular task.

Table 1 Summary of four basic construction routings

After the process model is constructed, resources need to be allocated to tasks. In some workflow engines, it is possible to define specific rules as resource allocation principles, such as Four Eye Principle, Deadline Rules, and etc. In this project, eight process related rules (See 3.2) for the financial filing process will be used as a proof of concept of the project solution. Also it is necessary

to include data information in the process model, since it is important to indicate conditions for tasks to be performed or to edit data values that are transferred from one task to another.

In order to provide a better insight into workflow process modeling, research on workflow patterns has been carried out at Eindhoven University of Technology and Queensland University of Technology. This research aims at providing a thorough examination of the various perspectives (control flow, data, resource, and exception handling) which need to be supported by a workflow language or a business process modeling language. (6) The result of this research categorizes workflow patterns into three types, i.e. control-flow patterns, data patterns, and resource patterns. Control-flow patterns have eight classes, namely Branching Patterns, Synchronization Patterns, Repetition Patterns, Multiple Instances (MI) Patterns, Concurrency Patterns, Trigger Patterns, Cancellation and Completion Patterns, and Termination Patterns. Data patterns have five classes, i.e. Data Visibility Patterns, Internal Data Interaction Patterns, External Data Interaction Patterns, Data Transfer Patterns, and Data Routing Patterns. Resource patterns have seven patterns, i.e. Creation patterns, Push patterns, Pull patterns, Detour patterns, Auto-start patterns, Visibility patterns, and Multiple Resource patterns. An overview of workflow patterns that are supported by different workflow management system vendors can be found in the Appendix.

2.3 Requirements for this project

A workflow process can be generic, however, in order to solve the two particular problems mentioned in Chapter 1, i.e. data assurance and data reusing, special requirements are needed for workflow process used in this project. Considering the three perspectives that can be included in a workflow process, the requirements are classified in these three groups as well, i.e. requirements on control flow, requirement on data information, and requirements on organizational model.

Before specifying the requirements on control flow, it is useful to first understand how the financial filing process looks like for Deloitte. The process given here is just an example of how Deloitte deals with financial filing process, but ideas and concepts can be generalized from this example.

Usually, the final product of the financial filing process is a well edited annual report of its client that needs to be handed in to the client's stakeholders, external investors, and tax officers. The annual report describes the client's future strategy and its performance in the previous year by showing its financial statement.

1. Static Report and Dynamic Report

The financial statement normally consists of two parts, i.e. Balance Sheet and Income Statement. A Balance Sheet is a static report. It contains financial values that are recorded at an instant time spot, for example, at the end of the fiscal year. By reviewing and comparing the Balance Sheet from two different time spots, you can analyze the financial structure, organizational performance and profitability of the company. An Income Statement is a dynamic report. It records profits and costs of the company during a certain period of time. By reviewing the Income Statement, you can find out indexes about the company's business scale and growth rate.

2. Multiple Use of Data

At the end of each fiscal year, Deloitte starts collecting data from its clients to prepare for the annual report. At the highest level, we can see that one annual report is needed for multiple end users. Actually this is also true for lower level report, for example Balance Sheet is needed for Tax Office as well as for the Chamber of Commerce. Another example is that the value of Property Plant and Equipment is needed as one of the inputs to calculate Fixed Assets in Balance Sheet, but it can be also used to calculate Depreciation in Income Statement.

3. Sub process

After data is collected, auditors and tax officers from Deloitte can start creating the actual financial statement report and its lower level report, such as Balance Sheet and Income Statement. In order to ensure the validity of data in the reports, two or three persons are asked to perform three tasks on the report, in the order of Create, Review and Sign Off. The Create step collects all necessary data for the report, and format them in a good order. The Review step checks these data values against financial filing rules and sees if these values are valid. The Signoff step assures that the report has been created under authorization. Normally an allocation rule called "Four-Eye Principle" is used here, in order to prevent a signal authorization over one report.

2.3.1 Control Flow

From the process description above, we can see there are several challenges in constructing the control flow model for financial filing processes.

1. The model has to be flexible. Since there will be multiple ways of obtaining data values (obtained from the client directly or calculated by other available data), it will be much easier to handle the case of having more than one possible inputs if the control flow model is flexible. This is related to **data reuse** in this project.

2. The model has to be rational. Because some data values will be needed for more than once during the process, the control flow model should allows using the same data as many times as it is needed. This is related to **data reuse** in this project.

3. The model has to be extendable with sub processes. From the description above, we can see that to create a financial report is not a simple step as to just create. It contains many more steps, such as Create, Review and Signoff. This is related to **data assurance** in this project.

4. The model has to reserve the relationships between data values and reports that are used for the process. The financial filing process has a clear hierarchical structure of the data used. The control flow process has to be able to indicate what data are used as input and what data will be produced as output. This is related to both **data reuse** and **data assurance** in this project.

2.3.2 Data Aspect

From the description above, we can see two different types of data are used in financial filing process, i.e. static data and dynamic data. The data used in the financial filing process has to be considered for the following requirements.

1. The data value should contain time aspect information. During the financial filing process, one important step to check whether a data value is valid or not is to check the date of the data. For example, when preparing Balance Sheet of 2008, if the fiscal year ended at 31st December 2008, a data value with the date of 10th January 2009 will be considered as invalid. In order to perform this check, it is important to include time aspect information.

2. The same data can be duplicated with the same meaning but with different time attributes. As described previously, one financial report may need more than one set of static data. For example, when preparing for the Balance Sheet of 2008, all needed data has to have two records of values, i.e. with the date at the beginning of the fiscal year (31st December 2007) and the end of the fiscal year (31st December 2008).

2.3.3 Organizational Model

During the sub processes, an allocation rule called "Four-Eye Principle" is used to prevent the single authorization for security purposes during the financial filing process. By obeying this allocation rule, two different resources have to be assigned to the needed tasks. The organizational model in the financial filing process has to be designed to make use of this rule.

2.3.4 Conclusion

In conclusion, considering the three aspects in a workflow model, it seems difficult to construct a process model out of nothing. Fortunately, XBRL taxonomy provides a starting point for this project, as it looks similar to a PDM, which we are going to introduce in the next section.

2.4 Product Data Model

Product Based Workflow Design is a revolutionary workflow design methodology, in which the workflow product is the central concept in the design process instead of the activities in the business process (7). Like the Bill of Material (BoM) for the manufacturing industry, a Product Data Model (PDM) is used as the product model to describe the structure of the product being produced. One main difference between BoM and PDM is that the products in BoM are un-reusable physical objects, but products in PDM are information which can be used as many times as needed after it is available.

To illustrate how PDM works, we take the process of calculating Profit before Tax for example. In this process, values of Advertising, Insurance and Salary determine the value of Expenses, values of expenses and Revenue determine the value of Net Income, and value of Revenue determines the value of Tax. Finally, values of Revenue and Tax determine the value of Profit before Tax. Values of Advertising, Insurance, Salary and Revenue can be obtained from client. However, it is also possible to determine the value of Revenue by the value of Advertising. From this example PDM, we can see a clear structure of how the final product, Profit before Tax, is produced, what its inputs are and where those inputs come from. Figure 3 shows the PDM of this process (See Appendix for the PDM XML file) .

In Figure 3, we can see that *Data elements* are depicted as circles in the PDM. The actions that are taken on the data element values are called *operations* and are represented by arcs. Each operation can have zero or more input data elements and produces exactly one output data element. An

operation can have a number of *attributes*, such as execution cost, processing time, execution conditions, failure probability and resource class, to describe the characteristics of the operation in more details (7).

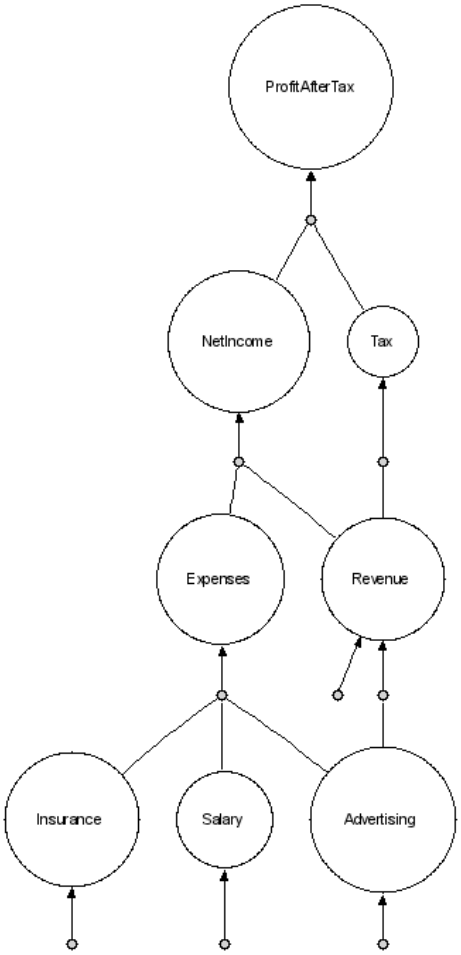


Figure 3 An example of PDM

Considering the requirements mentioned in the previous sub section, the following advantages can be obtained by using PDM as the workflow process model in the financial filing processes.

1. PDM is rational. The biggest advantage of using PDM is that it is possible to reuse available data, because PDM is defined in a way that enables multiple uses of data elements in the process. This is quite crucial, since to make available data reusable is also one of the two main problems that need to be solved in this master project.

2. PDM is objective. Because the product specification is taken as the basis for the design of PDM, each recognized data element and each production rule can be justified and verified with this specification. As a consequence, there are no unnecessary tasks in the resulting workflow. This can improve the efficiency of the financial filing process.

3. PDM is flexible. Not only is reusing data important, but also the way how the data is reused. Since PDM is data-oriented process model, it is more flexible and thus easier in handling

dynamic situations when more than one possible input is available for one data, compared with traditional workflow model.

4. PDM reduces time and cost for financial filing processes. Since it is easy to duplicate values of information, especially when it is stored in digital form, PDM saves time and cost when reusing data.

5. PDM provides insight into the process. Because of the data-oriented hierarchical structure of PDM, it makes the understanding of a financial filing process much easier for people who are not very familiar with workflow management concepts. Researchers have shown that people tend to understand better when looking at a PDM than looking at a traditional workflow process model.

6. PDM can be generated automatically from XBRL. Based on the similarity of the data structure between PDM and XBRL taxonomy. It is possible to convert XBRL taxonomy into PDM automatically. This can save a lot of designing time for PDM. More details on how PDM can be generated from XBRL can be found in section 3.4.

Though Product Based Workflow Design have many advantages and solves one of the main problems in this project, there are also drawbacks. For example the application of Product Based Workflow Design is an intensive effort because of the thorough analysis of the product specification that is required (7). The XBRL taxonomy we used for this project is comparatively simple, but in practice, an XBRL taxonomy can contain as many as 2,000 concepts with more than 2,000 relationships between those concepts. With all different crossing content links in the XBRL taxonomy, the quality of the converted PDM from XBRL taxonomy is hard to decide for now.

Also, adapting to product based workflow management asks for a clear understanding from internal process experts, strategy decision maker and IT supporters on the new roles and new responsibilities that are going to take in the new process.

Moreover, currently the only tool available to directly execute PDM is in the form of a prototype. There are no commercial tools yet which makes PDM more difficult to apply in practice. Research is being carried on regarding this subject, but the software is not mature enough to deliver a satisfactory result for this project. So, in order to use PDM as a design of workflow process model as well as to prove the concept in a running workflow engine, we decided to convert the PDM to an executable process model, i.e. YAWL model. This will be explained more in Chapter 4.

3 eXtensible Business Reporting Language

This chapter will explain another important objective in this project, i.e. eXtensible Business Reporting Language (XBRL). The first sub section will give the definitions of terminologies in XBRL and the structure of XBRL. The second sub section will explain financial filing rules in more details. The third sub section discusses possible solutions to extend XBRL with workflow information. The last section explains in details how to generate PDM from XBRL taxonomy.

3.1 Introduction of XBRL

eXtensible Business Reporting Language (XBRL) is an open standard business reporting language that facilitates the transmission of financial information by providing a common dictionary of unique identifying tags that are applied to the individual elements of financial statements and the notes to financial statements (1). It is commonly used by stock exchange centers, regulators and governments, investment analysts, banks, financial data companies and accountants to fasten financial filing process, reduce human errors, and improve data integration. Many countries, such as U.S., most of European countries, China, Japan, and etc, have already noticed the importance and business value of adapting XBRL. Many researches for example data assurance on XBRL, data mapping between XBRL and databases, and efficiency in XBRL taxonomy design, are being carried on to refine and improve the structure and functionality of XBRL as well as to seek for more business opportunities to adapt XBRL.

The XBRL framework splits business reporting information into two components: taxonomies and instances (8). The XBRL taxonomy is design level information that abstracts the structure, and relationships between different concepts. The XBRL instance contains instance level information, such as value, time attribute, and monetary attribute, on concepts that are defined in the XBRL taxonomy. In most cases, because financial filing regulatory is the same within one country, XBRL taxonomy development projects are carried out within country-size scale. As a result, when a company wants to prepare a financial report, instead of creating a complete new XBRL taxonomy, it will directly use the national XBRL taxonomy or extend the national XBRL taxonomy according to the company's specific industry. According to its XBRL taxonomy structure, the company will then create XBRL instance file. Finally, an XBRL engine will combine two files together and generate a well edited financial report. The following two sub sections will explain XBRL taxonomy and XBRL instance in more details.

3.1.1 XBRL taxonomy

XBRL taxonomy is comprised of an XML Schema and all of the linkbases contained in that schema or directly referenced by that schema (8). The taxonomy schema defines each *concept*, i.e. a reporting term, with a concrete name, type, balance, and period type. The *linkbases* in taxonomy further document the meaning of the concepts by expressing relationships between concepts (inter-concept relationships) and by relating concepts to their documentations (8).

There are five kinds of linkbases used in XBRL taxonomies, i.e. calculation, definition, presentation, label and reference. Relation links (calculation, definition, and presentation) manage the relations between taxonomy elements. Label links manage the text associated with taxonomy elements in various languages. Reference links manage the references to authoritative literature (either online

or paper) (8) . The first three types of extended link express inter-concept relationships, and the last two express relationships between concept and their documentation.

Formal definitions for terms that are often used in XBRL taxonomy are listed below(8).

Taxonomy A taxonomy is an XML schema and the set of XBRL linkbases that it references using linkbaseRef elements and the linkbases that are nested within it.

Item An item is an element in the substitution group for the XBRL item element. It contains the value of the simple fact and a reference to the context (and unit for numeric items) needed to correctly interpret that fact.

Concept Concepts are defined in two equivalent ways. In a *syntactic* sense, a concept is an XML Schema element definition, defining the element to be in the *item* element substitution group or in the *tuple* element substitution group. At a *semantic* level, a concept is a definition of kind of fact that can be reported about the activities or nature of a business activity.

Extended links An extended link is an element identified as an extended link using the syntax defined in the XML Linking Language [XLINK]. Extended links represent a set of relationships between information that they contain and information contained in third party documents.

Linkbase A linkbase is a collection of XML Linking Language [XLINK] extended links that document the semantics of concepts in a taxonomy.

The following xml clips give a better understanding on the terms and definitions in XBRL taxonomy and XBRL taxonomy linkbases.

- Concept

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

This example shows a definition of a *concept* in XBRL taxonomy. In this case, the *name* of the *concept* is “Assets”. “Assets” is with the *type* of “xbrli:monetaryItemType” in the *substitutionGroup* an “xbrli:item”. It is with the *periodType* of “instant”, which means it is a static value as described in Section 2.3.

- Linkbase

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

 <loc xlink:type="locator"
    xlink:label="fr_CurrentAssets"
```

```
    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" />
```

This example shows a Calculation Linkbase. Similar to the other four linkbases, this calculation linkbase is defined using two <loc> nodes and one <Arc> node. As its name indicates, the <loc> node locates the concept in the taxonomy, while the <Arc> node connects the two located concepts by the relationship defined in the <Arc> node. This example Calculation Linkbase gives the relationship that the *first* input of concept "Assets (fr_Assets)" is concept "Current Assets (fr_CurrentAssets)" with the *weight* of one.

3.1.2 XBRL instance file

XBRL instance file contains fact values for concepts that are defined in the XBRL taxonomy. Both instance file and taxonomy file are needed in order to create a financial report. During the financial filing process, new values will be created and added into the initial instance file until all needed fact values are available. Terms that are often used in XBRL instance files are defined below.

XBRL Instance — XBRL instances are XML fragments with root element, *xbrl*. XBRL instances contain business report facts, with each fact corresponding to a concept defined in their supporting DTS. XBRL instances also contain contexts and units that provide additional information needed to interpret the facts in the instance.

Fact — Facts can be simple, in which case their values must be expressed as simple content (except in the case of simple facts whose values are expressed as a ratio), and facts can be compound, in which case their value is made up from other simple and/or compound facts. Simple facts are expressed using items (and are referred to as items in this specification) and compound facts are expressed using tuples (and are referred to as tuples in this specification).

The following xml clip shows a part of example instance file.

- Fact

```
< 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>
```

This example defines 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.

An important technique that is often used in preparing for a XBRL instance file is data mapping. Since data are often stored in the company's database systems, Excel files or other kinds of formats, it is handy to automate the step of mapping digitally stored data with fact values in XBRL instances. A lot of XBRL vendors, for example Altova, UBMatrix, and Fujitsu support data mapping functions.

3.2 Financial Filing Rules

In Chapter 1, we introduced financial filing rules as a way to standardize financial filing process considering data reusing and data assurance. This sub section is going to explain financial filing rules in more details. It will give a specific set of rules as an abstract of real case financial filing rules. The financial filing rules given in this sub section can be categorized into two types, i.e. content-related rules and process-related rules. It is important to note that the financial filing rules given in this sub section are not a complete set of rules that are generally used among financial companies, they are shown here just as an example.

3.2.1 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. As explained in 2.3, a financial report contains static report and dynamic report. Static reports are reports with fact values (also called instant values) which are collected at a time spot; while dynamic reports are reports that contain fact values (also called durational values) which are collected during a period of time. In a XBRL taxonomy linkbase, concept with different period types cannot be defined in one linkbase. As a result, when defining content-related rules, we also separate rules between the same period type and rules cross different period types. Example content-related rules can be found as below.

Rule 1 (Between Instant Values)

Assets = Current Assets + Fixed Assets

Rule 2 (Between Instant Value and Durational Value)

Depreciation from 2008/1/1 to 2008/12/31 = PP&E 2008/1/1 – PP&E 2008/12/31

3.2.2 Process-related Rules

Process-related rules specify relationships between process-related information, such as data creating time, task originator and etc., of taxonomy concepts. The current version of XBRL specification does not allow taxonomy schema to include process related information. Considering one of the main problems described in 1.4, i.e. XBRL data assurance, it is important to include process-related information in XBRL taxonomy and instance. By including process-related information in XBRL taxonomy, it makes it possible to trace changes of fact values in XBRL instance file. This is the main idea in this project to solve the data assurance problem for XBRL financial filing. However, in order to implement this idea, XBRL schema has to be extended in order to

include this process-related information. There are many ways of extending XBRL taxonomy with process-related information. For example, it can be done by creating a Workflow Linkbase, using Instance File footnote, and etc. More details on how to extend XBRL taxonomy with process-related information will be explained in the next sub section. The following list shows some example process-related rules.

Rule 3 "Create" and "Review" should be done by different resources

Rule 4 "Sign Off" should be done by resource with the position of either "Director" or "Client"

Rule 5 A higher level concept in the XBRL taxonomy should not be "Signed off" if a lower level concept in the XBRL taxonomy is not "Signed off"

Rule 6 "Current Assets" should be available two weeks before the end of the reporting period for the corresponding fiscal year

Rule 7 "PP&E" should have two instance values, i.e. at the beginning of the fiscal year and at the end of the fiscal year

Rule 8 The duration for data element "Depreciation" should be from the beginning of the fiscal year to the end of the fiscal year

Rule 9 "Depreciation" for a certain period should not be available before the end of that period

Rule 10 "PP&E" at a certain time should not be available before that time

3.2.3 Storing of Financial Filing Rules

In the previous sub section, we gave a general idea of financial filing rules. Now we are going to answer the question of where should these rules be stored.

First of all, these rules should be included in workflow process, so that workflow engine can recognize these rules and restrict the financial filing process for data assurance checking and data reusing according to these rules. Thus, these rules should be stored in workflow process models.

Secondly, in order to fasten the data assurance, it is wise to store workflow information in XBRL instance file. This can save time by looking into both the workflow engine and XBRL engine for data assurance checking. However, workflow information is an instance level information, while financial filing rules are design level information. Thus, in order to store workflow information in XBRL instance file, it is necessary to store financial filing rules in XBRL taxonomy.

Thirdly, since these rules are highly related to concepts and the relationships between concepts in the XBRL taxonomy, these rules should be stored in the XBRL taxonomy so that financial filing experts can easily define new rules without bothering workflow process model designers.

To summarize, financial filing rules should be stored in both workflow model and XBRL taxonomy. More details on how XBRL can be extended with financial filing rules can be found in Section 3.3. How financial filing rules are stored in workflow model, i.e. PDM, can be found in Section 3.4. A prototype of how financial filing rules can be used can be found in Chapter 5.

3.3 Extension to XBRL with workflow information

As shown in the framework of the project (See Figure 1), one important part in this project is to extend XBRL with workflow information (Step 1 in Figure 1). To be more specific, the XBRL taxonomy needs to be extended in a way that workflow information, such as organizational structure and financial filing rules, especially process-related financial filing rules, can be designed in the taxonomy file.

There are several ways to extend XBRL with workflow information. The following techniques can be used as guidelines and suggestions.

1. Create Workflow Linkbase

Under the XBRL specification, it is possible to create new types of Linkbases, such as Versioning Linkbase, Formula Linkbase, and Rendoring Linkbase, using Generic Links(9). By using Workflow Linkbase, it will be possible to not only define workflow concept but also link the workflow concept to other concepts, either workflow concept or financial data concept, in the XBRL taxonomy file.

2. Use XBRL Instance File footnote

Footnote is designed to deal with irregular structured associations between facts in XBRL Instance file. For instance, several facts may all be linked to the sentence "Created by Yaqing Sun, at October 15th, 2010." In this way, workflow information can be added to XBRL Instance file. However, since footnote deals with irregular structured information in XBRL Instance file, it has several drawbacks to apply this method. First, workflow information is not structured in footnote, since footnote only stores irregular structured information. This makes it difficult for data assurance check because workflow information will be hard to retrieve. Second, it is not possible to specify design level information in XBRL Taxonomy. In this case, only XBRL Instance file knows what kind of workflow information it contains, but on design level, there is no overview of this information in XBRL Taxonomy.

3. XML Signature and XML Encryption

The aim of adding workflow information to XBRL taxonomy is to provide a new way of assuring the financial filing process. In this case, XML Signature and XML Encryption techniques can be used to perform the assurance task. However, this technique can only be used for data assurance. In this project, we also need the workflow information in XBRL to check the availability of data values being produced during the financial filing process.

In conclusion, considering the project framework (see Figure 1), we think the first proposal to create Workflow Linkbase would be the best choice to extend XBRL with workflow information as to ensure data validity. As we mentioned before, most companies do not have a hard copy of a pre defined financial filing rules. Most financial filing tasks depend largely on the experience of the financial filer. Thus the creation of a workflow linkbase can help the financial filers to specify their own way of perform their tasks. Also, by using workflow linkbase, different information can be sorted in the XBRL taxonomy and XBRL instance file. This would make it easier for financial filers to manage their XBRL based financial reports.

3.4 Generating PDM from XBRL

In section 2.4, we mentioned that one of the advantages of choosing PDM as the workflow design method is that it can be automatically generated from XBRL taxonomy. This section will explain in more details of how to generate PDM from XBRL regarding three aspects, i.e. data structure, organizational model and financial filing rules. This section also shows how Step 2 and 4 in the project framework (See Figure 1) can be solved.

3.4.1 Data Structure

From the example PDM shown in Figure 3, we can see a clear hierarchical structure shown in PDM. Since PDM focuses on products instead of tasks during one process, data elements in PDM have high dependencies on each other. Taking the advantage of this, it is not difficult to convert an XBRL taxonomy into a PDM following the guidelines described below.

1. Interpret XBRL taxonomy concepts as data elements in PDM.

Facts values for taxonomy concepts can be seen as products created during a financial filing process. Since PDM focuses on products rather than tasks in a process, it is reasonable to treat taxonomy concept as data elements in PDM.

2. Duplicate data elements in PDM if the taxonomy concepts have more than one entry of instance fact figures with different recording dates.

One difference between XBRL taxonomy and PDM is that PDM does not consider time aspect for data elements, while XBRL taxonomy does. For example, in order to create a Balance Sheet, normally two or more sets of fact values are needed for one same taxonomy concept. This is because Balance Sheet has to provide a comparison of financial information between the beginning of the **current** fiscal year with the beginning of **previous** fiscal year or years. In XBRL taxonomy, the concept is defined only once, and this concept can have multiple entries of fact values in XBRL instance file according to different needs. Since current version of PDM does not support time aspect for data elements, we have to distinguish this same concept with different recording dates by duplicating the same data elements with different labels indicating their recording time. The data element is duplicated as many times as the fact values are needed for the corresponding concept.

3. Use Content-related rules to build up PDM tree.

As explained in section 3.2, Content-related financial filing rules specify relationships between taxonomy concepts. If one taxonomy concept is interpreted as one data element in PDM, these content-related financial filing rules can be used to indicate relationships between data elements. In this way, we can build up a PDM tree.

To generalize the use of content-related rules, taxonomy concepts on the left side of the content-related rule equation will be treated as output data elements, while taxonomy concepts on the right side of the content-related rule equation will be treated as input data elements.

3.4.2 Organizational Model

In workflow management, three aspects need to be considered, i.e. control flow, data information, and organizational model. When using PDM as the product, the control flow and data information are already taken into account in the PDM when the hierarchical data structure was built. Thus what is left to be considered is the organizational model in PDM.

Many process-related financial filing rules used in our project are associated with organizational information. For example, Rule 4 states that "Sign Off" should be done by resource with the position of either "Director" or "Partner". In this process-related rule, the name of the position for "Sign Off" task will not be known if the organizational model is not provided for this process. In order to properly define process-related rules, it is important to include organizational model in PDM.

Currently it is not possible to define organizational structure in PDM except for a list of roles. And because of the limited implementation on resource structure in PDM, this list of roles is designed to deal with simple cases. In addition, the resource information can only be specified in the PDM file itself rather than any other formats, e.g. a reference xml file. All these limitations make it very difficult to manage the PDM file itself and its organizational model.

As a result, we propose to use a separate file to store organizational model for PDM. In this way, companies can define their own XML schema to store organizational structure. And the change of organizational functions and structure does not require a change to the PDM file.

3.4.3 Sub process

As explained in section 3.2, nearly all reports generated during the financial filing process are needed to undertake a sub process following the sequence of "Create", "Review" and "Sign Off". This sub process is performed to assure the correctness of data values produced, and to provide auditing authorization on reports that are created. Though information on the sub process is not stored in XBRL taxonomy, it is stored in traditional financial filing process models (Petri Nets, BPMN, EPC, and etc). If we want to use PDM as product model in financial filing process, it is important to include this sub process in it.

Operations (noted as *dot*) in PDM, can be seen as tasks as in traditional workflow process model. If sub process is needed in PDM, the operation element can be the one to be extended with such information. Since PDM is data oriented model, extension on sub processes in operations are not considered when PDM schema was designed. In this project, we propose to change the PDM scheme as follow.

```
<xs:element name="Task">
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element name="ResourceRef" type="xs:IDREF" minOccurs="0"/ >
            <xs:element name="Condition" minOccurs="0" maxOccurs="unbounded"/ >
            <xs:element name="Cost" minOccurs="0"/ >
```

```
            <xs:element name="Time" minOccurs="0"/ >
            <xs:element ref="Data" minOccurs="0"/ >
        </ xs:sequence>
        <xs:attribute name="TaskID" type="xs:ID" use="required"/ >
        <xs:attribute name="TaskName" type="xs:string" use="optional"/ >
    </ xs:complexType>
</ xs:element>

<xs:element name="Operation">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Input">…</ xs:element>
            <xs:element name="Output">…</ xs:element>
            <xs:element name="TaskList">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="TaskRef"  type="xs:IDREF" / >
                    </ xs:sequence>
                </ xs:complexType>
            </ xs:element>
        </ xs:sequence>
        <xs:attribute name="OperationID" type="xs:ID" use="required"/ >
        <xs:attribute name="Descriptioin" type="xs:string" use="optional"/ >
    </ xs:complexType>
</ xs:element>
```

The following PDM clip gives a glance of how this structure can be used in a PDM to indicate sub process.

…

```
<Task TaskID="Create_Final_Report">
    <ResourceRef>Creator</ ResourceRef>
</ Task>
<Task TaskID="Review_Final_Report">
    <ResourceRef>Reviewer</ ResourceRef>
</ Task>
<Task TaskID="SignOff_Final_Report">
    <ResourceRef>SignOff</ ResourceRef>
</ Task>
…
```

```
< Operation OperationID="Generate_Final_Report">
    < Input>IncomeStatement</ Input>
    < Input>BalanceSheet</ Input>
    <Output>FinalReport</ Output>
    <TaskList>
        <TaskRef>Create_Final_Report</ TaskRef>
        <TaskRef>Review_Final_Report</ TaskRef>
        <TaskRef>SignOff_Final_Report</ TaskRef>
    </ TaskList>
</ Operation>
…
```

3.4.4 Financial Filing Rules

Being part of the financial filing rules, content-related rules have already been applied during the conversion from XBRL taxonomy to PDM. In addition, we can also see that Rule 7 and 8 in the process-related rules can also be solved by generating a duplicate tree for all data elements of Balance Sheet report with different time labels. Nevertheless, there are still financial filing rules that are not specified or stored anywhere in the PDM.

Similar to organizational model, it is not possible to store the process-related financial filing rules in PDM. Thus, we propose to have a separate file to store unsolved process-related rules as we did for organizational model. We can refer the financial filing rules in the PDM in the "condition" attribute for the needed task. By making a separate file, it will be easier for product model owner to manage, edit and extend the existing set of process-related rules. Also, some financial filing rules can be used multiple times in the process by giving different values to its parameters. By separating the financial filing rules file with the PDM file, it is possible to make reuse of the created financial filing rules.

3.4.5 Conclusion

This chapter introduced some basic knowledge on XBRL based financial filing processes. It listed a set of financial filing rules that are commonly followed by financial filers. Moreover, it explained a conceptual solution of how to extend XBRL with audit trial information. This is followed by the section of how to convert PDM from XBRL and how to include all necessary information, i.e. sub processes, financial filing rules, and organizational model in PDM.

As mentioned in the previous chapter (see section 2.4), the tool to execute PDM is still at a prototype stage. It does not provide a complete set of functions that a workflow engine include, e.g. it does not produce log files. For this reason, we decided to convert PDM into an executable process model as to prove our proposed concepts and solutions for this project.

4 Generating process model

As explained in section 1.4, we decided to use PDM as the product model as a design method in this project in order to deal with data reusing problem in financial fining processes. The tree-shape hierarchal structure of PDM can make advantage of data points that need to be used multiple times during the process. However, tools for executing PDM are still under implementation, we need to convert PDM into an executable process model (Step 4 in project framework, see Figure 1). In this sub section, we are going to discuss the seven algorithms that can be used to convert a PDM to an executable process model (7). Sub section 4.1 first explains the seven algorithms based on its classification framework. Sub section 4.2 gives a set of requirements on these seven algorithms as criteria for selecting the best algorithm for general purpose and as well as for this project in particular. Finally, sub section 4.3 evaluates the seven algorithms against the chosen requirements. This sub section also compares the best two algorithms and selects one algorithm for this project.

4.1 Introduction of PDM converting algorithms

The seven algorithms are created and classified based on a classification framework. This framework contains two perspectives, i.e. construction perspective and process execution perspective. Construction perspective includes **representation**, **order**, and **focus**, while process execution perspective includes **moment of choice**, **eagerness**, and **concurrency**. The list below summarizes the meaning of these six perspectives (7).

Construction Perspectives:

Representation The modeling language in which the process model is represented. All the seven algorithms convert the PDM either to a *Petri Nets* model or a *YAWL* model.

Order The order in which the algorithm translates the PDM to a process model. There are three orders that a PDM can be translated, i.e. *top-down*, *bottom-up*, and *middle-out*. The top-down order translates the PDM tree from its root element to the leaf elements; the bottom-up order translates the PDM tree from the leaf elements to the root element; the middle-out order translates the elements in the PDM tree in an arbitrary order.

Focus The focus of the translation can either be *data elements* or *operations* in the PDM.

Process Execution Perspectives:

Moment of choice If there is an alternative operation to produce the value for a specific data, there will be two moments when the choice for an alternative route can be made. For example, in Figure 3, two alternatives to get value of Revenue is direct input or determined by the value of Advertising. If the choice is made before any data value is available, the moment of choice is *early*; if the choice is made as defer choice, the moment of choice is *late*.

Eagerness The process model can have a structure in which it is encouraged to produce as many data element values as possible, but it can also be structured to only

produce values that are strictly needed to determine the value of the end product. There are three degrees of eagerness, i.e. *strict*, *overcomplete, but restricted*, and *overcomplet and unrestricted*.

Concurrency If there is concurrency in a process model, there is a possibility to simultaneously execute several activities of the process model for the same case. Concurrent activity executions in a process model can either be possible or impossible.

The figure below gives an overview of the seven algorithms based on this classification framework (7).

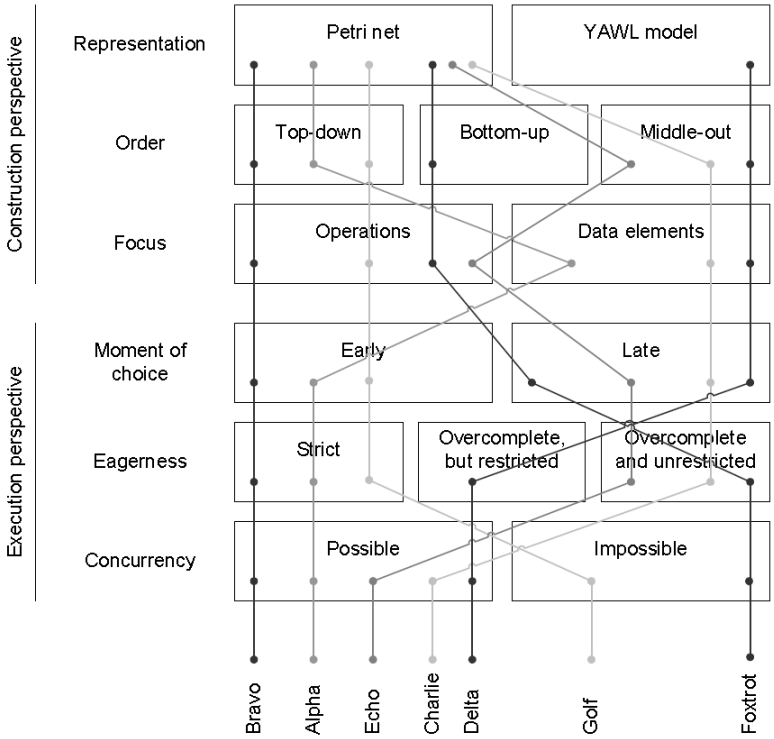


Figure 4 Classification of the seven algorithms

4.2 Requirements for algorithms

The previous section explained the seven algorithms that can convert a PDM to an executable process model. As we can see, each of them uses different structures and is created for different purposes. Some of them are created to deal with choices presented in PDM, while others are created to reuse data. In order to choose the best algorithm for any project that uses these algorithms, a general set of requirements is defined as a guideline for algorithm selection.

4.2.1 General set of requirements

This set of requirements is made considering two aspects of the algorithms, namely its outcome and its performance. Most of the requirements are related with the execution perspective in the classification framework. The following list will describe each requirement in more details with a name, the reasons to be included, and its business impact.

Requirements based on outcome of the algorithm*:*

1. ***Is able to reuse data***

   Technical description: One reason to use PDM is that it considers the process as an information flow in which produced information can be used multiple times. Having the goal of improving the process efficiency, it is very necessary to keep this advantage in the converted process model.

   This is related with the eagerness property of the algorithms. Since strict eagerness means only data elements that are strictly needed will be produced, most likely an algorithm with a strict eagerness property will not be able to reuse data. In this case, we will need to look for algorithms that have unrestricted eagerness property.

   Business impact: Available information can be used as many times as they are needed.

2. ***Soundness***

   Definition of Soundness:

   > *The process starts from one token in the only one initial place, and end with one token in the only one final place with no remaining tokens in other places.*

   The soundness property of a process model is one of the criteria to check the correctness of the process model, since an unsound model may lead to deadlocks or unwanted remaining tokens in the process. Depending on where the token is when the process is finished, this soundness requirement can be divided into three sub requirements.

   a. ***Is able to reach the end place***

      Technical description: This requirement makes sure that the converted process can produce the wanted information under any circumstances. If a process model cannot produce an end product, it means it includes deadlocks and will easily get stuck during execution.

      This requirement needs to be strictly obeyed because this is the basic property of a sound process. Without meeting this requirement, the process model cannot be considered as a correct and applicable process model.

      Business impact: The converted process model should be able to produce an end product. The process should not get stuck somewhere while it is under execution.

   b. ***Do not have multiple tokens in end place***

      Technical description: As mentioned before, having the goal of improving process efficiency, the process model should meet this requirement. Because if the converted process produces multiple tokens in the end place, it would indicate that the end products have been produced multiple times.

Another problem with multiple tokens in the end place is with the data consistency. It does not matter if the input data is the same, otherwise two tokens with two different values will be produced for one data element. In a process deals with information flow, this will cause the problem of which data value should be selected. As to avoid this kind of situation, the process model should not allow producing multiple tokens in the end place.

Business impact: The converted process model does not produce more than one end product with different values.

c. Do not have remaining tokens

Technical description: The reason for a remaining token in the process can be that two sets of input values are available for one data point (referred as data point "A"). So two tokens are produced in the place of data point "A" and only one token is consumed in the process. The problem with the remaining token is that if the two sets of input lead to different values for data point "A", the end product may contain inconsistent values.

Whether to include this requirement or not depends on the strictness of the soundness definition. The soundness definition given above is very strict, which is also hard to apply in practice. Normally, less strict soundness definitions, e.g. relax soundness or lazy soundness, are used to ensure the correctness of the process model. Relaxed soundness means that there exist enough executions which terminate properly (i.e. without spare tokens). It does not force the modeler to think about all possible executions and then to care for proper termination in all cases (10). By the definition of relax soundness, it requires that at least all intended behavior has been described correctly. Lazy soundness allows remaining tokens in the process as long as the end product is produced only once.

Considering remaining tokens, this requirement is highly related to the lazy soundness property. If we use lazy soundness as desired process property, this requirement can be excluded from the requirements set. However, in order to see how remaining tokens can influence the financial filing process, we chose to use the strict soundness definition in this project.

Business impact: No activities can take place after the process finishes execution.

3. Is able to handle alternative choices

Technical description: PDM allows one data point to be produced in many different ways using different sets of input data. For example, data point "A" can be produced by input "B" and "C" together or "D" only. In case all three values for input "B", "C" and "D" are available, the process needs to decide a better stream to proceed.

This requirement is related with moment of choice property in the classification framework. Like described above, if the stream is made before the data availability is known, the process

model cannot provide alternative choices; otherwise, if the process takes the data availability into consideration, it also allows the process to include deferred choice.

One big aspect to consider whether a control flow model is able to handle alternative choices or not is to see whether the alternative choice structure effect the soundness property of the control flow model or not. If the alternative choice structure causes deadlocks or remaining tokens in the process, it will be considered as unable to handle alternative choice.

Business impact: The converted executable process model should be flexible to handle the situation where two possible sets of inputs are available for one output element.

Requirements based on performance of the algorithm:

4. Time complexity

Technical description: Some large PDMs can have a large number of data points, so it is very important that an algorithm is able to handle this large set of data within a certain amount of time. The algorithm is not desirable if it takes too long for an algorithm to generate a process model from a PDM.

Business impact: PDM can be converted to an executable model within a required period of time.

5. Space complexity

Technical description: No matter how advanced a computer is, there is a limit to its computing ability. As mentioned before, some PDMs may contain a large number of data points. This requirement ensures that the chosen algorithm is able to generate a process model within the computer's computing capacity.

Business impact: PDM can be converted to an executable model within the computer's computing capacity.

These two requirements based on performance of the algorithm are related with the concurrency property in the classification framework. Not allowing simultaneous execution means every situation has to be taken into consideration while building up the process model. As a result, it takes more time and computing space to build up such process models that include all possible steps the process may take.

4.2.2 Specific requirements for this project

The set of requirements described above is made for general purpose, thus some of them may not apply for this project. Now we are going to explain which requirements are considered as relevant, which are not, and why.

1. Is able to reuse data **Relevant**

Since data reusing is one of the main problems for this project. This requirement is relevant and important for this project.

2. Soundness

    a. Is able to reach the end place **Relevant**

Since the goal of financial filing process is to produce financial reports, which are the end products of a process, it is important that the process model can reach the end place.

    b. No multiple tokens in end place **Relevant**

In order to ensure the efficiency of the financial filing process, it is not desirable to produce multiple tokens in the end place. Thus this requirement is considered as relevant.

    c. No remaining tokens **Relevant**

One goal of improving financial filing process is to improve the process efficiency. If a remaining token in the process can trigger unnecessary tasks to be undertaken, the idea of combing XBRL and workflow management systems will be less significant. Thus, this requirement is considered as relevant.

3. Ability to handle alternative choices **Relevant**

In the financial filing process, normally it is not sure what data will be available at the initial stage. So it is important the process model can handle alternative choices. This gives the process more flexibility when executing the process. Thus this requirement is relevant.

4. Time complexity Irrelevant

Since the creation of an executable process model will be a one-time job only, it does not affect that much whether the converting process takes 30 minutes or 2 days as long as it is within a reasonable amount of time.

5. Space complexity **Relevant**

In real case, an XBRL taxonomy can contain as many as 2000 concepts, which means there will be 2000 data points in PDM as well. So the chosen algorithm should be able to convert a process model within computers' computing capacity.

In summary, except time complexity, all requirements listed in the requirements set are related to this project. The next section will evaluate the seven algorithms against the chosen requirements.

4.3 Evaluation on algorithms

In this section, we are going to evaluate the seven algorithms based on the requirements described above. We will take the sample PDM (see Figure 3) as an example and convert it into seven process models using the seven algorithms. As described earlier, this sample PDM is an abstract of a real financial filing product model with one reuse of data "Revenue" and one alternative choice for the same data. The table below gives an overview of the evaluation result of these seven algorithms.

	Alpha	Bravo	Charlie	Delta	Echo	Foxtrot	Golf
Is able to reuse data	-	-	+	+	+	+	+
Is able to reach the end place	+	+	-	+	+	+	+
No multiple tokens in end place	+	+	+/-	+	+/-	+	+
No remaining tokens	+	+	+/-	+	-	+	+
Ability to handle choices	+	+	-	+	-	+	+
Space complexity	+	+	+	+	+	-	-
Total +	5	5	2	6	3	5	5
Total +/-	0	0	2	0	1	0	0
Total -	1	1	2	0	2	1	1

Total +: Match **Total -: Does not match** **Total +/-: Match under condition**

Table 2 Evaluation result of the algorithms against the selected sets of requirements

From the table above, we can see that algorithm Delta seems to be the best match for this project. The reason why we chose to convert PDM is because we would like to have an executable model to prove all the concepts introduced in this thesis. As explained in the previous section, except for Delta produces a YAWL model, the other six algorithms produce Petri nets model. This, in addition, makes algorithm Delta even more appealing to us, since YAWL model is executable already, while Petri nets model is not.

However, in order to execute the YAWL model, we still need to include extra information, such as data information and organizational structure. So the YAWL model converted from algorithm Delta cannot be used directly. In this case, no matter YAWL model or Petri nets model, we need to change the converted process model in order to execute the process to meet our goal in this project. For this reason, we decided to investigate closely on all the seven algorithms. The conclusion of the sub section will be a suggestion of a group of useful algorithms that can be used for this project. Furthermore, Chapter 5 will describe how the suggested algorithm can help to make the prototype.

In extension to the table shown above, now we are going to analyze the seven process models in more details in terms of its advantages of disadvantages.

- **Evaluation on algorithm Alpha and Bravo**

First of all, process model converted by Bravo has a nice interpretation of alternative choice in the PDM (See Figure 5). In the sample PDM (See Figure 3), there are two ways of determining the value of Revenue. It can be either informed by the client or determined by the value of Advertising. In the process model of Bravo, it interprets this as a deferred choice which enables an easy skip of the unnecessary step in the process without causing any remaining tokens in the process.

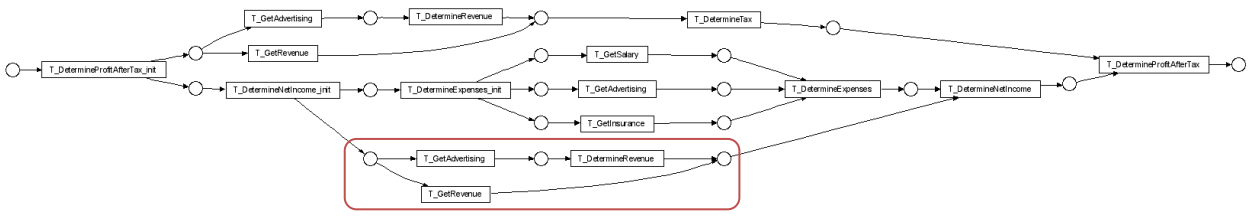


Figure 5 Process Model by Bravo - alternative choice

However, process model converted by Bravo is not able to reuse data. In the sample PDM, value of Revenue is used twice to determine the value of Tax and the value of Net Income. In the process model converted by Bravo, the value of Revenue is produced twice rather than consumed twice (See Figure 6). This will cause a data consistency problem is the two groups of input data for Revenue are not the same. Similarly, this is the same with process model converted by Alpha (See Figure 7).

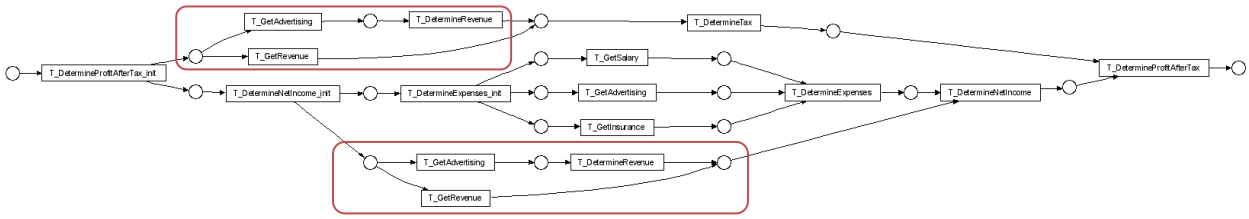


Figure 6 Process Model by Bravo - no reuse of data

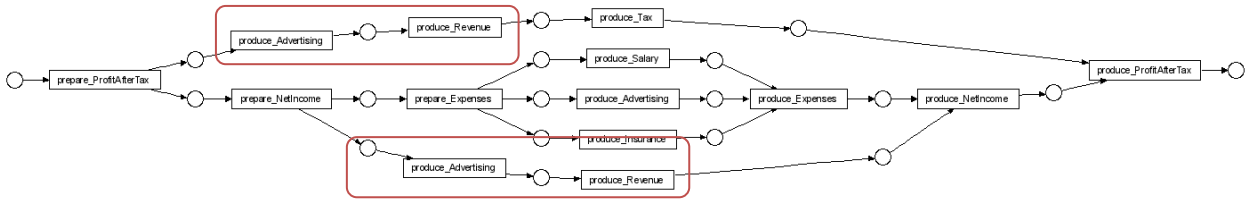


Figure 7 Process Model by Alpha - no reuse of data

Being one of the two problems in the XBRL based financial filing process, reuse of data is also one of the main requirements that the converted process model should meet. Thus, converting algorithm Alpha and Bravo are excluded from the suggested algorithms because they cannot efficiently reuse the available data.

- **Evaluation on algorithm Charlie and Echo**

Secondly, algorithm Charlie contains two deadlocks, which makes it impossible to reach the end place (See Figure 8). The first deadlock occurs after task “Advertising”. Since only one token is produced by the task “Advertising”, task “Determine Expenses” will never be enabled, if task “Determine Revenue” first fires. In another case, if task “Determine Expenses” fire first, there will be only one token for task “Revenue”, which will cause the second deadlock after task “Revenue” is fired, since task “Determine Tax” and task “Determine Net Income” will be never to enabled at the same time. This allows implies that algorithm Charlie cannot produce a process model that can reuse data. Furthermore, even considering the relax soundness definition given previously (See section 4.2.2), which defines a less strictly soundness property, process model converted by Charlie is not qualified to match it.

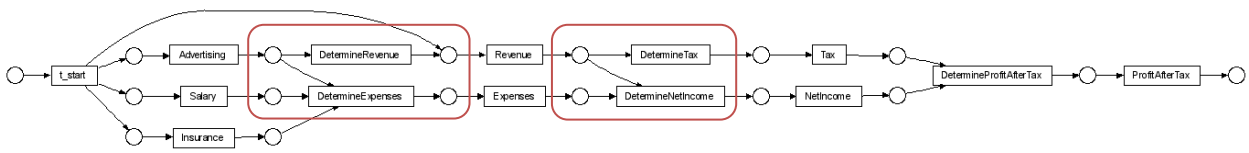


Figure 8 Process Model by Charlie - deadlocks

Nevertheless, the process model converted by Charlie can still be used for this project after making a small change to it (See Figure 9). In the revised process model, task "Advertising" and "Revenue" produces two tokens instead of one token after they are fired, since value of Advertising and value of Revenue may be needed twice during the process. This change can give a deadlock free process model. However, the revised process model will cause remaining tokens in the process because it lacks the ability to handle alternative choice. In the sample PDM file, it gives an alternative choice to decide the value of "Revenue". In the revised process model converted by Charlie, no matter choosing which way, the value of "Revenue" has to be produced twice, thus value of "Tax" will also be produced twice. Fortunately, this will not cause the process to produce multiple products in the end place for this project. However, this will be problematic if "Tax" is one of the final products.

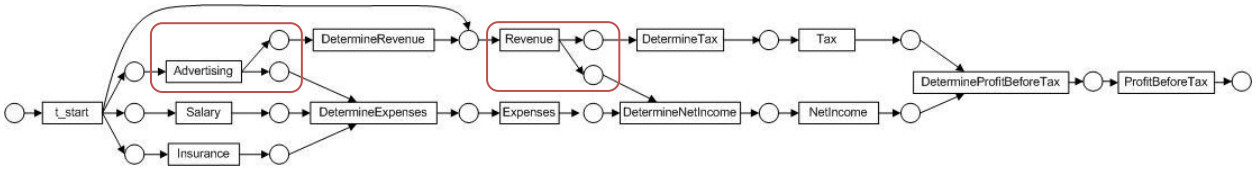


Figure 9 Revised process model by Charlie - deadlock free

Very similar with the revised process model converted by algorithm Charlie, the process model converted by algorithm Echo has the same problem of containing remaining tokens and bad performance of choice handling. But one main advantage of algorithm Echo is that is can reuse data. Moreover, the process model converted by algorithm Echo looks very alike to the revised model converted by algorithm Charlie (See Figure 9 and Figure 10).

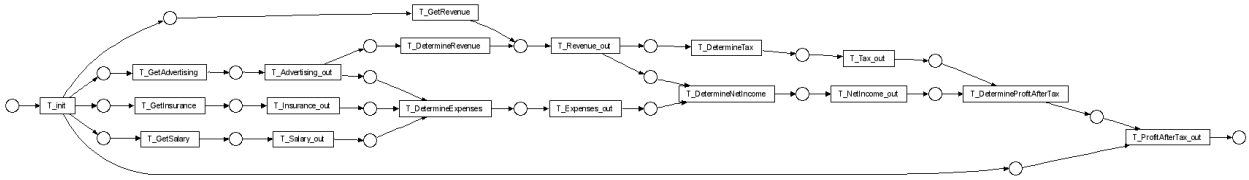


Figure 10 Process model by Echo

Additionally, though process models converted by algorithm Echo and revised algorithm Charlie does not meet the strict soundness definition, they match with the lazy soundness definition. Both process models have remaining tokens in the process, but no unwanted tasks can be fired when the process model finishes execution. This is also one of the requirements that algorithm Echo intended to meet, i.e. lazy soundness property. Considering that the process models can reuse data and match with lazy soundness property, we suggest including algorithm Charlie and Echo in the recommended group of conversion algorithms.

- **Evaluation on algorithm Delta**

Thirdly, the YAWL model converted by algorithm Delta (See Figure 11) is not very readable for control flow analysis. So we improved the layout before we evaluate the process model converted by algorithm Delta (See Figure 12).

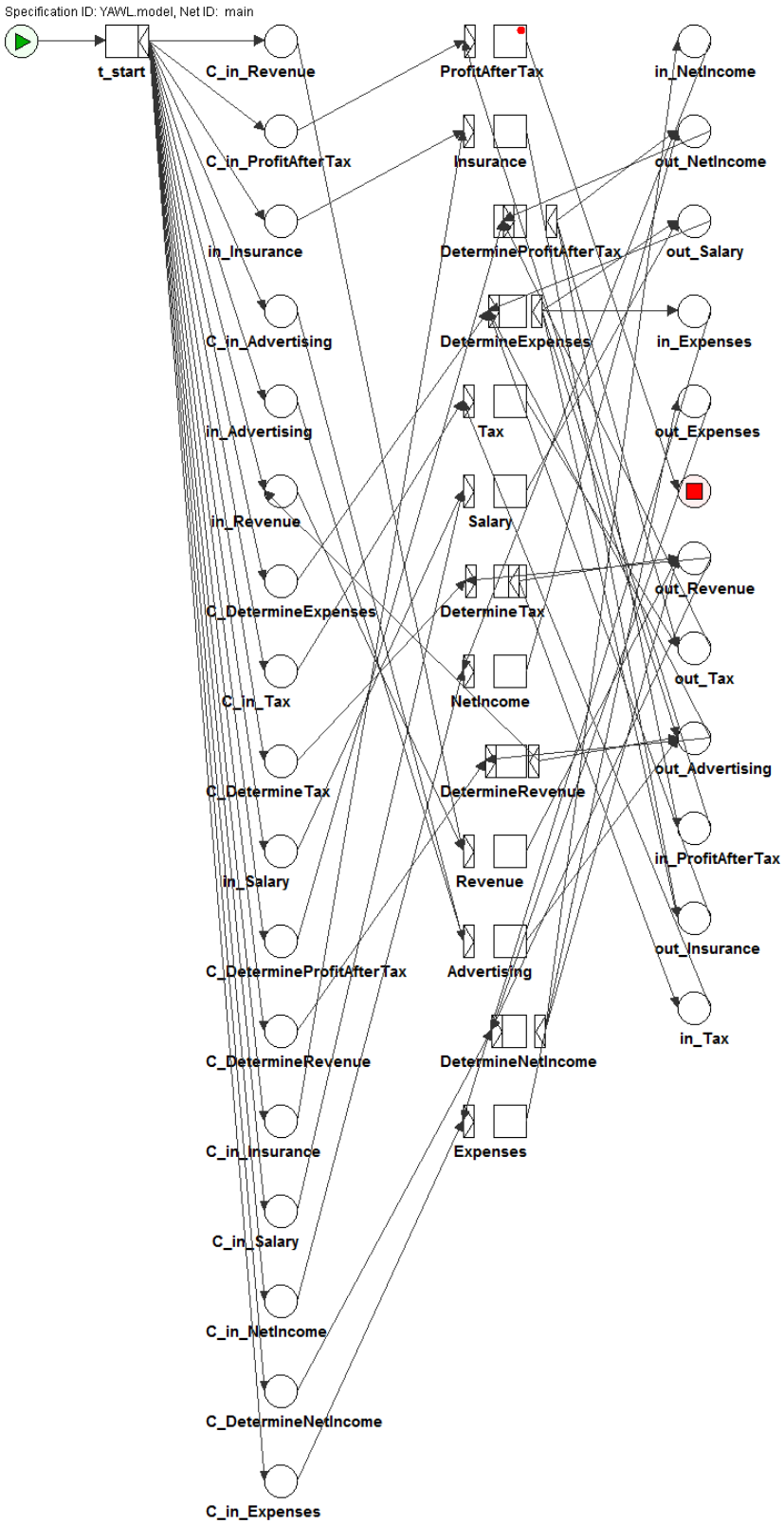


Figure 11 Process model by Delta - original layout

From the process model with an improved layout (See Figure 12), we can see that the YAWL model converted by algorithm Delta is able to reuse data because it always sends the used token back to the place where it was consumed. And because of the use of cancellation region, the YAWL model does not have any problems with remaining tokens in the process. It is also able to handle choice since the process only allows every data value to be produced only once (See Figure 12). In the model, the place "C_in_Revenue" is used to ensure that the task "Revenue" will be able to fire only once in the process.

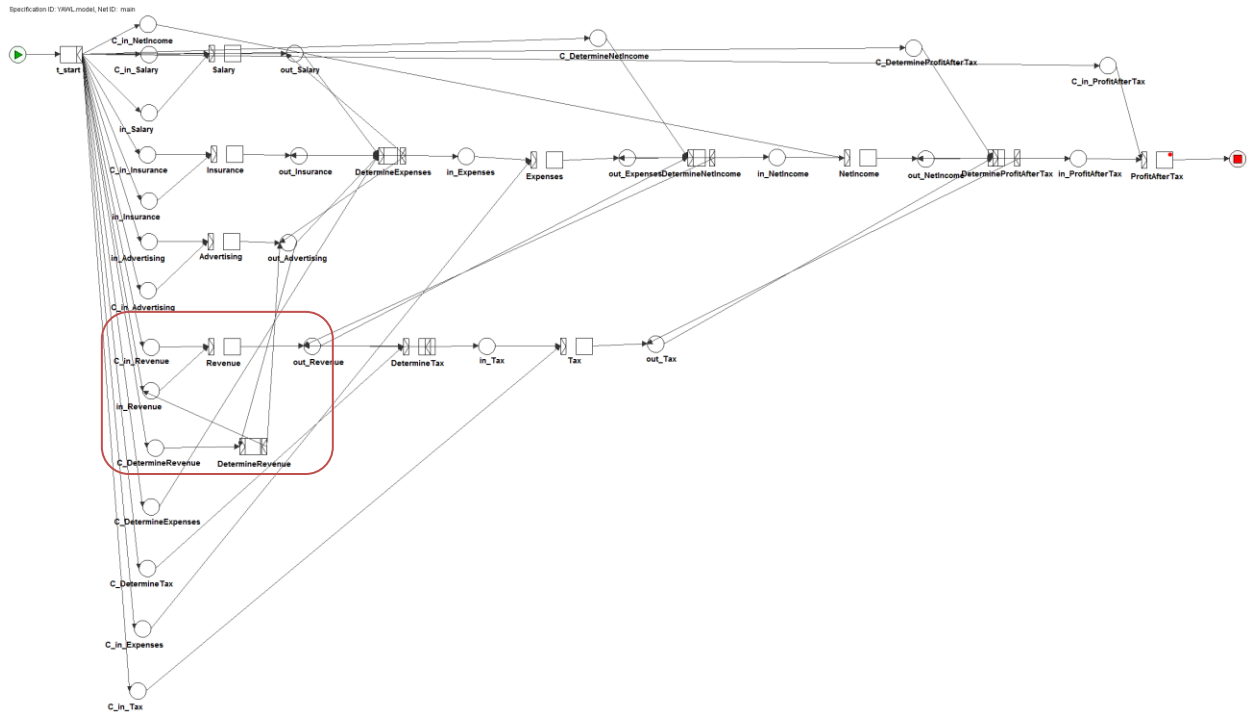


Figure 12 Process model by Delta with improved layout - alternative choice

As mentioned before, there are three main advantages of the process model converted by algorithm Delta. First, it is an executable model. Second, it has cancellation region that can deal with remaining tokens and alternative choices. Third, it can reuse data. However, the main drawback of the process model converted by YAWL is its unclear format. The original YAWL model converted by algorithm Delta is very difficult to read since places and tasks do not follow the PDM structure (See Figure 11). Considering the format of a process model can always be improved, we include algorithm Delta as one of the recommended algorithms to use for this project.

- **Evaluation on algorithm Foxtrot and Golf**

Finally, the last two algorithms, i.e. algorithm Foxtrot and algorithm Golf, consider every possible path that can take place in a process. Thus one advantage of the process model converted by these algorithms is the extreme extension of its flexibility. However, this is also one of the main disadvantages of these two algorithms, since it takes up too much computing space to build up the complete process model. The sample PDM has four leaf nodes and a four level tree structure, and the process model converted by algorithm Foxtrot (See Figure 13) and algorithm Golf (See Figure

14) contains 67 and 96 tasks respectively. Considering that a real taxonomy can contain as many as more than 200 data elements, it is impossible for these algorithms to produce a process model within the computing capacity of a normal computer. Thus these two algorithms are excluded from suggestion for this project.

Figure 13 Process model by Foxtrot

Figure 14 Process model by Golf

To conclude, there are three algorithms that can be used for this project, i.e. Algorithm Charlie, Algorithm Delta, and Algorithm Echo. The evaluation result shows that the outcome from algorithm Delta matches best with our expectations. However, one main disadvantage is that the process model converted by algorithm Delta is hardly readable for us. The format of the process model is not structured. Moreover, financial filing rules and organizational model are not included in the converted YAWL model by algorithm Delta, thus changes need to be made to the converted YAWL model.

As for algorithm Charlie and algorithm Echo, even though the two process models converted by these two algorithms seem very similar, the original process model converted by algorithm Charlie contains deadlocks while process model converted by algorithm Echo does not. And the process model converted by algorithm Echo even meets lazy soundness property if we loosen the soundness requirement to use lazy soundness rather than strict soundness.

Overall, all three algorithms cannot produce a process model that can be directly used for the prototype of this project. Small adjustments, such as changes to the format of the YAWL model by algorithm Delta, application of financial filing rules, conversion from Petri nets to YAWL model by algorithm Charlie and Echo, have to be made to obtain desired properties of the process model. As a proof of concept, Chapter 5 will show a prototype using algorithm Echo as an example. It will discuss how the process model converted by algorithm Echo can be modified to support XBRL based financial filing process. To point it out, algorithm Echo is selected as a proof of concept. In fact, the other two algorithms are also possible to use for this prototype.

5 Prototype

As a proof of concept, we made a prototype for this project using the concepts described in the previous chapters. This chapter describes this prototype in more details and explains how to achieve each step shown in the framework of this project (See Figure 15).

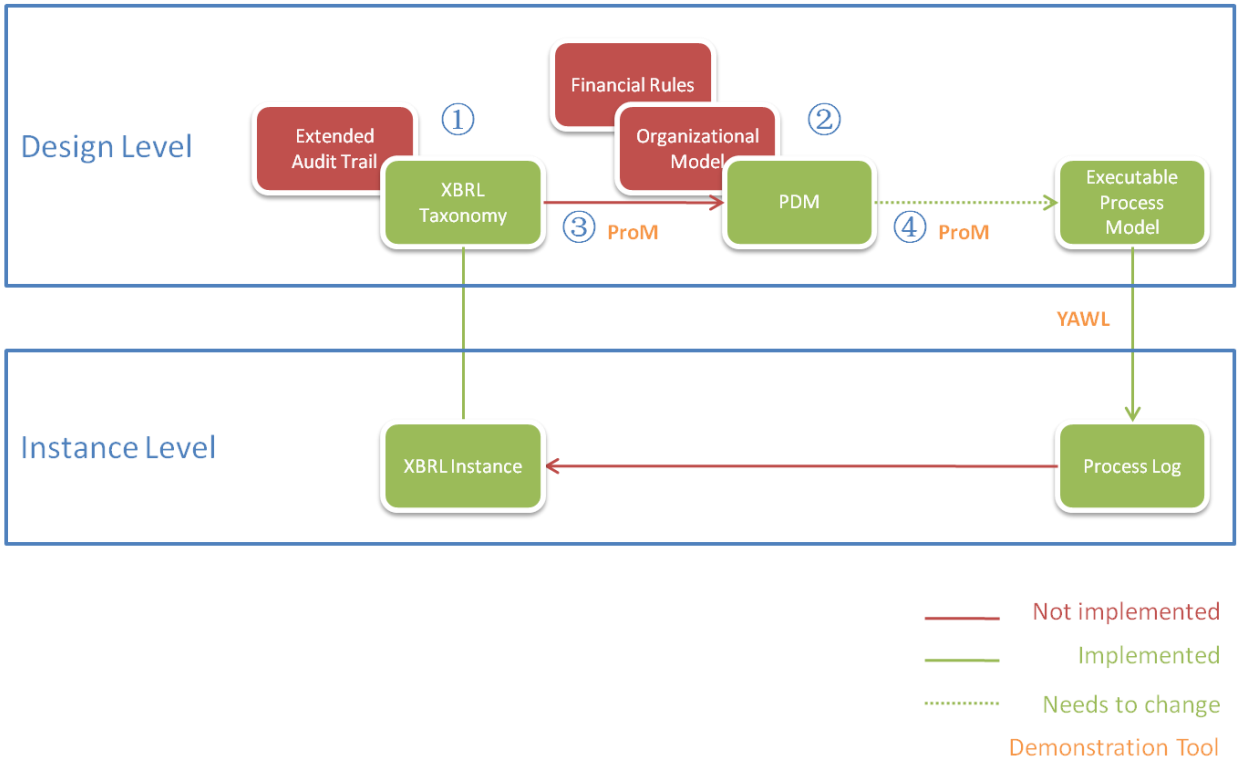


Figure 15 Framework of the project

Section 5.1 lists the assumptions used for this prototype. Section 5.2 explains the demonstration tools used in this project, i.e. ProM, YAWL, and UBmatrix. Then, section 5.3 gives a sample XBRL taxonomy that will be used for this project (Step 1), followed by Section 5.4, which shows the sample PDM converted from the sample XBRL taxonomy (Step 3). Section 5.5 shows the executable process model converted from the PDM in three aspects (Step 4), i.e. control flow, data information, and organizational structure. Section 5.6 gives the log file obtained from the YAWL Engine. Section 5.7 summarizes the result of prototype in terms of the problems that the prototype has solved.

5.1 Assumptions

The following assumptions are made in order to set a scope for this prototype.

Assumption 1 The sub process for finalizing financial documents is “Create”, “Review”, and “Sign off”.

In reality, more detailed steps will be taken during the financial filing process, and these steps also differ depending on what financial report is going to be created. However, the three steps, i.e. “Create”, “Review”, and ”Sign off”, are milestones that are needed for almost all financial reports.

This assumption is made in order to provide a conceptual solution to the problem by abstracting high level information from reality.

Assumption 2 In the XBRL taxonomy, a concept with a "false" value for "abstract" attribute is considered as a data element; while a concept with a "true" value for "abstract" attribute is considered as a document.

Assumption 3 Data elements only need one step of "Create", while documents need all three steps of "Create", "Review", and "Sign off".

In real financial filing processes, normally all data elements are reviewed as a whole document. Also the purpose of the project is to show a conceptual solution to the problem, Together with Assumption 2, these two assumptions are made to simplify the question while still reserve its level of difficulty.

Assumption 4 Values in the initial XBRL instance file are correct.

Since rejecting initial XBRL instance file is not taken into account in this project, this assumption is made to cross out the possibility of having conflicting values handled in the process.

Assumption 5 An XBRL instance file has two values for an instant concept and one value for a durational concept.

An XBRL instance file can have infinite number of values for one concept, since the purpose is to provide a conceptual solution, in this project, the instance file will include two values, i.e. at the beginning of the fiscal year and at the end of the fiscal year, for every instant type of concept and include one value, i.e. for the duration for the whole fiscal year, for every durational type of concept. This assumption is made to simplify the question while make use of Rule 2 defined previous.

Assumption 6 The activity will be immediately completed once a notification is sent to the corresponding resource.

Some process-related rules set deadlines for producing certain data elements or reports. Since the scenario of missing the deadline is not considered in this project, this assumption is made to make sure that every deadline will be met during the process.

Assumption 7 The process-related rules are a representative set of rules retrieving information on the design level of the process.

This assumption limits the project to consider financial filing rules only on the design level, which means no instance level information is used in these rules.

5.2 Demonstration tools

There are two types of demonstration tools used for this prototype, i.e. XBRL tools and workflow management tools. Many XBRL tools, e.g. UBmatrix, Altova MissionKit, and Fujitsu, are available in the market to support XBRL taxonomy design, XBRL instance file creation, and XBRL validation. For this prototype, we used UBmatrix Taxonomy Designer, because the starting point of this prototype

is to create XBRL taxonomy; in addition, UBmatrix XBRL products are commonly used within Deloitte for XBRL projects. Workflow management tools used in this prototype are ProM and YAWL. ProM is used for converting PDM to executable process models, while YAWL is used as a workflow engine. These two tools are completely or partly implemented within Eindhoven University of Technology. The following three sub sections will briefly introduce these three tools.

5.2.1 UBmatrix

UBmatrix (http://www.ubmatrix.com) is one of the founding developers of the XBRL standard and a key contributor to major XBRL taxonomies and implementations worldwide (11). UBmatrix products cover the full lifecycle of XBRL development, such as taxonomy design, report builder, database adapter, processing engine, and deployment of XBRL applications. With Charles Hoffman, main creator of XBRL, being one of the technical leaders, UBmatrix provides all kind of XBRL tools for companies that have different demands. The product that is used in this project, i.e. UBmatrix XBRL Taxonomy Designer, is a desktop application designed for building, extending, and maintaining XBRL taxonomies (11). In the framework of this project, UBmatrix can be used for XBRL taxonomy design, XRBL instance file creation, and financial report generating.

5.2.2 ProM

ProM (http://prom.win.tue.nl/tools/prom/) 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 (12). 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. Petri Nets, EPC, BPMN, YAWL, and etc. In the framework of this project, ProM can be used to convert XBRL to PDM, and convert PDM to YAWL model. In this prototype, conversion from XBRL to PDM is done manually. ProM is used to load the PDM file and convert it to a YAWL model using the chosen algorithm, i.e. Algorithm Echo, explained in Chapter 3.

5.2.3 YAWL

YAWL (http://www.yawlfoundation.org/) stands for Yet Another Workflow Language. It is a workflow system based on a concise and powerful modeling language. It handles complex data, transformations, integration with organizational resources and Web Service integration (13). YAWL contains two parts, i.e. YAWL Editor and YAWL Workflow Engine. The YAWL Editor works as a process modeling tool, while the YAWL Workflow Engine provides full functionality of a workflow engine, e.g. design of organizational structure, execution of process model, processing data information, evoking service application and etc. In the framework of this project, YAWL can be used to execute the process model directly converted from PDM. However, since the algorithm of converting PDM to YAWL does not include conversion of information such as financial filing rules and organizational structure. The process model used for execution in this project is manually constructed in the YAWL Editor including these kinds of extra information. The YAWL Engine is then used to execute this process model and provides process logs that can be later used to generate XBRL instance file.

YAWL was selected as the executing workflow modeling language for three reasons. First of all, the algorithms that convert a PDM to an executable process model produce either a Petri Nets model or a YAWL model. And YAWL model uses a very similar structure and workflow patterns as Petri Nets model. Secondly, YAWL is a free tool that has a complete set of functions for the purpose of our

project. YAWL covers almost all workflow patterns so it is easier to handle possible problems that may occur during the project. Thirdly, as many other conceptual solutions described in previous chapters, YAWL is not the only solution to work for this prototype. YAWL is only selected as a proof of concept.

5.3 Sample XBRL

The following figure shows the structure of the sample XBRL used in this project (See Appendix for the XBRL taxonomy file).

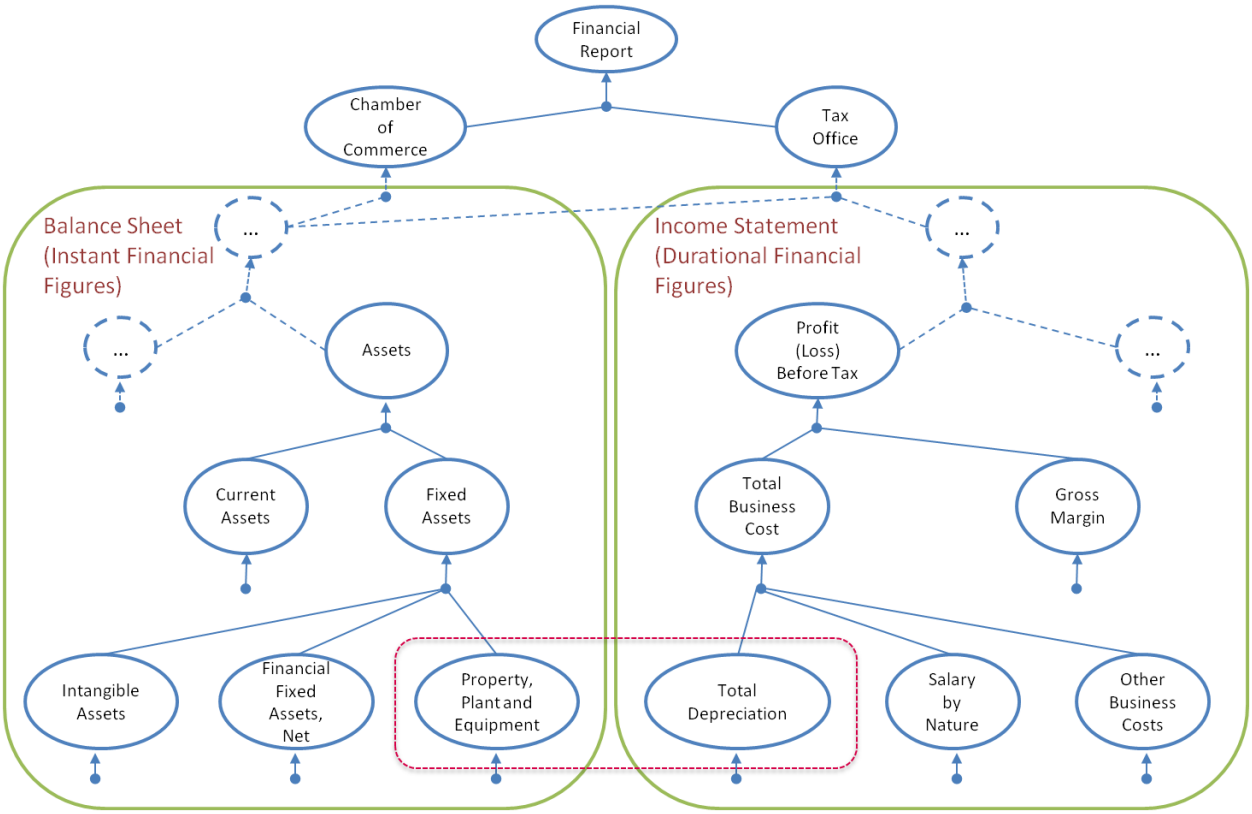


Figure 16 Structure of the Sample XBRL

This sample taxonomy is an abstract of a real XBRL taxonomy used for Dutch government. It is made by referring to the complete set of Dutch Taxonomy files. The following requirements are used when creating this sample taxonomy file.

1. It has to include instant financial figures and durational financial figures. In order to demonstrate the content-related rule between instant values and durational values, as shown in Financial Filing Rule 2, two types of reports are needed. As explained in Section 2.3, financial reports can be categorized into static reports and dynamic reports. The instant values and durational values can be found in these two types of reports. In this prototype, we use Balance Sheet as the example of a static report, and Income Statement as the example of a dynamic report.

2. It has to show the basic XBRL taxonomy structure. By basic XBRL taxonomy structure, we mean that it shows a clear and correct hierarchical relationship between data values in the XBRL taxonomy. This requirement is made in order to show the similarities between XBRL and PDM. It additionally proves one of the many advantages of using PDM for this project, i.e. PDM provides insight into the financial filing process.

3. It has to reuse of some concepts. In the taxonomy structure, we can see that Balance Sheet is used twice for both Tax Office and Chamber of Commerce.

5.4 Sample PDM

Once the XBRL taxonomy is created, a PDM is needed as product model to guide the financial filing process. In section 3.4, we explained the steps of how to convert XBRL taxonomy to a PDM. In this section, we are going to demonstrate those steps using the sample XBRL taxonomy given in the previous section. Technically, by creating algorithms, it is possible to automate the conversion steps from XBRL taxonomy to PDM. However, because of time limit in this project, conversion from XBRL to PDM is done manually for this prototype. In addition, since the extended schema of organizational structure and financial rules for PDM is still at a proposal stage, we only converted the data structure from XBRL taxonomy to PDM.

Figure 17 shows a PDM that is converted from the sample XBRL taxonomy file. (See Appendix for the PDM xml file.) From the picture, we can see how the three converting steps were used in this PDM.

1. One concept in XBRL taxonomy is interpreted as one data element in the PDM. For example, Report for Tax Office in the XBRL taxonomy is interpreted as TO_ForYY in the PDM.

2. XBRL taxonomy concepts that allow more than one entries (two entries in this project) of instance fact figures were duplicated as two data elements in PDM. For example, Assets in the XBRL taxonomy is interpreted as Assets_PreYY and Assets_YY in the PDM.

3. The PDM tree is built up based on the content related rules that are defined in the XBRL taxonomy. For example, according the XBRL taxonomy, Depreciation = PPE previous Year - PPE this Year. In the PDM, we can see that one input for data element Depreciation is the combined result of PPE_PreYY and PPE_YY. This indicates that with both the value of PPE_PreYY and PPE_YY known in the system, the XBRL processing engine can trigger the task to determine the value for data element Depreciation.

Besides the three converting steps from XBRL taxonomy to PDM that deserve special attention, the following points also need to be noticed in the PDM.

1. Basic data structure in PDM is the same as in the XBRL taxonomy structure. This is because PDM is built up based on the content related rules that are defined in the XBRL taxonomy. This similarity between the data structure gives extra points to the PDM being the product model for XBRL based financial filing process, because it is easier for the auditors and financial filers to understand the process and their tasks.

2. PDM has a duplicate tree for Balance Sheet with two different time labels. This is a demonstration of how the second step for converting XBRL taxonomy to PDM. According to Assumption 5, the static report, Balance Sheet, needs two sets of data values, e.g. for the current fiscal year and for the previous fiscal year, to form a complete report.

3. One of the main concerns in this project is to reuse available data. In this PDM, we can see that three data elements, i.e. Property Plant and Equipment Previous Year (PPE_PreYY), Property Plant and Equipment Current Year (PPE_YY), and Balance Sheet (BS_ForYY), are used more than once in the PDM. This shows the ability of PDM when handling data reusing problem.

4. Value of Depreciation (DepreciationDYY) can be determined in two ways, i.e. by the direct input from client, and by the combined inputs of Property Plant and Equipment Previous Year (PPEPreYY) and Property Plant and Equipment Current Year (PPEYY).

5. This PDM does not include financial filing rules and organizational structure.

As mentioned previously, since the PDM schemas for financial filing rules and organizational structure are proposals that need to be agreed (See section 3.4.3 and 3.4.4), thus these information is not included in this PDM file. However, it is possible to change the PDM schema as we proposed and include this information.

Another reason for not including the financial filing rules and organizational structure in the PDM is because PDM is not the process model that we are going to use for the execution purpose. Our goal is to show how all this information and restrictions can be applied when the process is under progress. Actually, these two kinds of information are possible to store and applied in the YAWL process model and engine. We are going to show how this can be done in the next section.

FR: Financial Report

CoC: (Report of) the Chamber of Commerce

TO: (Report of) the Tax Office

BS: (Report of) Balance Sheet

IS: (Report of) Income Statement

EBIT: Earnings Before Interest and Tax

TotBisCosts: Total Business Costs

GM: Gross Margin

Dep: Depreciation

OC: Other Costs

Sa: Salary

PPE: Property, Plant, and Equipment

FFAN: Fixed Financial Assets Net

IA: Intangible Assets

CA: Current Assets

FA: Fixed Assets

ForYY: for the year of YY

DurYY: during the year of YY

YY: the year of YY

PreYY: previous year of YY

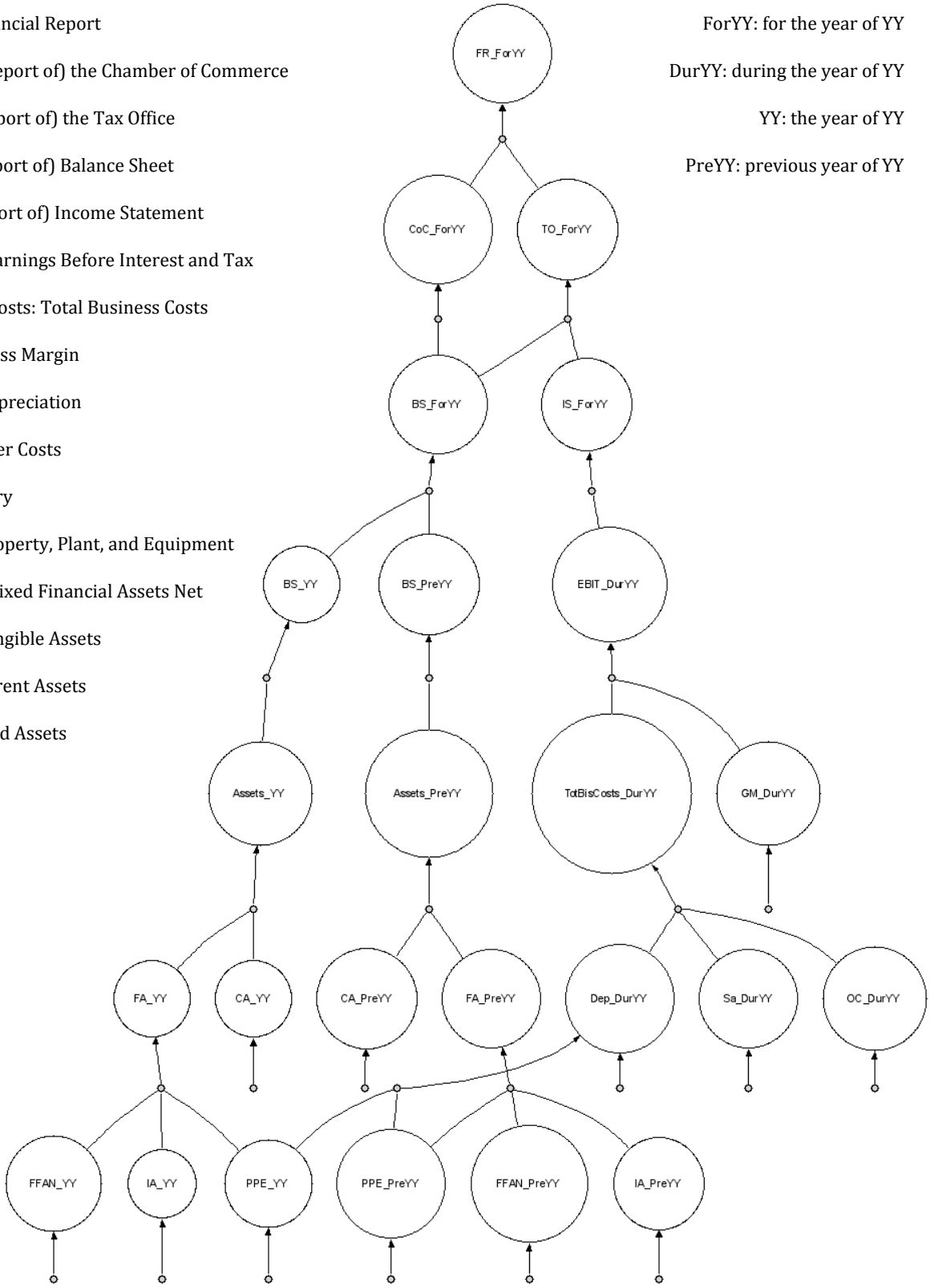


Figure 17 PDM converted from the sample XBRL taxonomy

5.5 YAWL model

The previous section shows the structure of the PDM. However, since the execution tool for PDM in not mature enough to support all functions that we need for this prototype, e.g. logging function, adding complex organizational structure, applying financial filing rules and so on (See section 2.4), we decided to further convert PDM into an executable process model, i.e. YAWL model. In Chapter 4, we introduced and evaluated seven algorithms that can convert a PDM into either a Petri Nets model or a YAWL model. For our project, we chose to use YAWL as the workflow engine because YAWL model looks very similar to a Petri Nets model, and the YAWL engine is an open source tool that has enough but not too many functions to prove our conceptual solutions proposed in the previous chapters in this thesis.

As concluded in Chapter 4, we chose to use algorithm Echo as the conversion algorithm from PDM to the executable YAWL model. The first sub section will explain the data information used in the YAWL model. The second sub section will explain how the YAWL control flow model was obtained, what problems it originally had and how we solved those problems. The third sub section will explain how organizational model looks like in this YAWL model. The last section shows some clips of the logging file that is generated during the execution of the YAWL model in YAWL engine.

5.5.1 Data Aspect

The following table shows a list of data that are used in this YAWL model.

As you can see that, we used Boolean type rather than Double type or Integer type of data because we do not want to calculate data values in the workflow engine. Since the financial filing process is an XBRL based process, we have the XBRL process engine to calculate and determine data values. Thus, the meaning of tokens in this process model as well as in each task in the process stands for the availability of that data element.

According to Assumption 2 (see section 5.1), we distinguish data elements and reports in the process. This assumption is made because reports need to undertake a three-step sub process while data elements do not. In the list below, you can see that for each data element, only one variable is used; but for report, three variables are used. For example, for Balance Sheet of the year of YY, three variables, i.e. BS_YY_Created, BS_YY_Reviewed, and BS_YY_SignedOff, are used. This was done to indicate the status of the Balance Sheet for the year of YY.

Actually, there are many ways to indicate the status of the reports and data element. We chose to define the variables this way because Boolean is the easiest way to show the availability of a data.

Net Decomposition Variables

Name	Type	Usage
IA_YY	boolean	Input & Output
IA_PreYY	boolean	Input & Output
CA_YY	boolean	Input & Output
CA_PreYY	boolean	Input & Output
FFAN_YY	boolean	Input & Output
FFAN_PreYY	boolean	Input & Output
PPE_YY	boolean	Input & Output
PPE_PreYY	boolean	Input & Output
FA_YY	boolean	Input & Output
FA_PreYY	boolean	Input & Output
Assets_YY	boolean	Input & Output
Assets_PreYY	boolean	Input & Output
BS_YY_Created	boolean	Input & Output
BS_YY_Reviewed	boolean	Input & Output
BS_YY_SignedOff	boolean	Input & Output
BS_PreYY_Created	boolean	Input & Output
BS_PreYY_Reviewed	boolean	Input & Output
BS_PreYY_SignedOff	boolean	Input & Output
BS_ForYY_Created	boolean	Input & Output
BS_ForYY_Reviewed	boolean	Input & Output
BS_ForYY_SignedOff	boolean	Input & Output
Dep_DurYY	boolean	Input & Output
OC_DurYY	boolean	Input & Output
Salary_DurYY	boolean	Input & Output
GM_DurYY	boolean	Input & Output
TotBisCosts_DurYY	boolean	Input & Output
EBIT_DurYY	boolean	Input & Output
IS_DurYY_Created	boolean	Input & Output
IS_DurYY_Reviewed	boolean	Input & Output
IS_DurYY_SignedOff	boolean	Input & Output
ReportIO_ForYY_Created	boolean	Input & Output
ReportIO_ForYY_Reviewed	boolean	Input & Output
ReportIO_ForYY_SignedOff	boolean	Input & Output
ReportCoC_ForYY_Created	boolean	Input & Output
ReportCoC_ForYY_Reviewed	boolean	Input & Output
ReportCoC_ForYY_SignedOff	boolean	Input & Output
FinancialReport_ForYY_Created	boolean	Input & Output
FinancialReport_ForYY_Reviewed	boolean	Input & Output
FinancialReport_ForYY_SignedOff	boolean	Input & Output

Table 3 Net variables in YAWL model

5.5.2 Control Flow

The control flow model is the main part in this prototype. The first step of getting the YAWL control flow model from algorithm Echo is to convert the PDM to a process model by algorithm Echo. This step was done in ProM (File -> Open PDM file -> Without Log file -> Find PDM file and Open -> Conversion -> PDM Model -> Product Data Model to Process Model Algorithm Echo). Followed by this step, the Petri Nets model will be converted into a YAWL model (See Figure 18) also using ProM (Conversation -> Selected Petri Net -> Labeled WF net to YAWL model). However, there are several problems with the converted YAWL model.

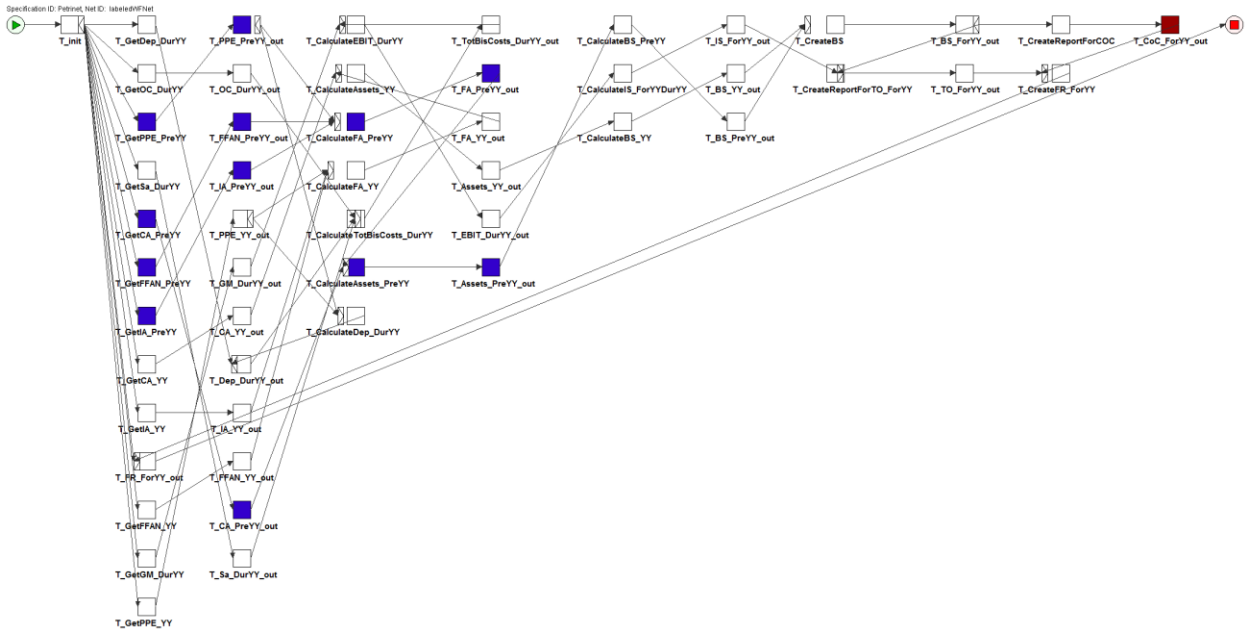


Figure 18 YAWL model converted by ProM

The biggest problem is that the financial filing rules and organizational model are not included in the YAWL model. This is not only because that PDM does not include those kinds of information but also because the conversion algorithms only consider the control flow aspect. Furthermore, it is very difficult see the hierarchical data structure in the YAWL control flow model. Thus it makes it even more difficult to change the YAWL model as to include the financial filing rules and organizational model.

The second problem with the converted YAWL model is that sub processes are needed for some data elements or tasks (See section 5.1 Assumption 3). For example, the task to create Balance Sheet for the year of YY (T_BS_YY) needs three steps, i.e. “Create”, “Review”, and “SignOff”. The same with the previous problem, the reason for this problem is that PDM does not include sub processes information and the conversion algorithm does not consider sub processes. In addition, Financial Filing Rule 5 (See section 3.2) requires that higher level data element cannot be signed off before the lower level data element is signed off. However, this implies that higher level data element can be created after the lower level data element is created. Bearing this in mind, the YAWL model should be adjusted accordingly.

The third problem with the converted YAWL model is that the process model does not meet the soundness requirement, i.e. it has remaining tokens in the process. Moreover, there are also unnecessary tasks in the process model, e.g. every task is split into two tasks, i.e. a start and an end task.

Lastly, the automatically converted YAWL model breaks the original structure in the PDM. For instance, the blue tasks are tasks to gathering inputs for Balance Sheet for the previous year. They are not grouped together but spread in the model, mixed with tasks for other financial reports. Also,

the outgoing and incoming arcs of all tasks are not clear to see in this YAWL model with its original layout.

Considering the problems mentioned above, we decided to manually construct a YAWL model (See Figure 19) based on the YAWL model converted from the Petri net model which is generated by algorithm Echo. In this YAWL model, one color stands for one type of data element or report. For example, the blue tasks are tasks that are needed to gather input data to create Balance Sheet for the previous year of YY; the pink tasks are the sub process to generate Balance Sheet for the year of YY. The following section will explain how those problems were solved in the manually constructed YAWL model.

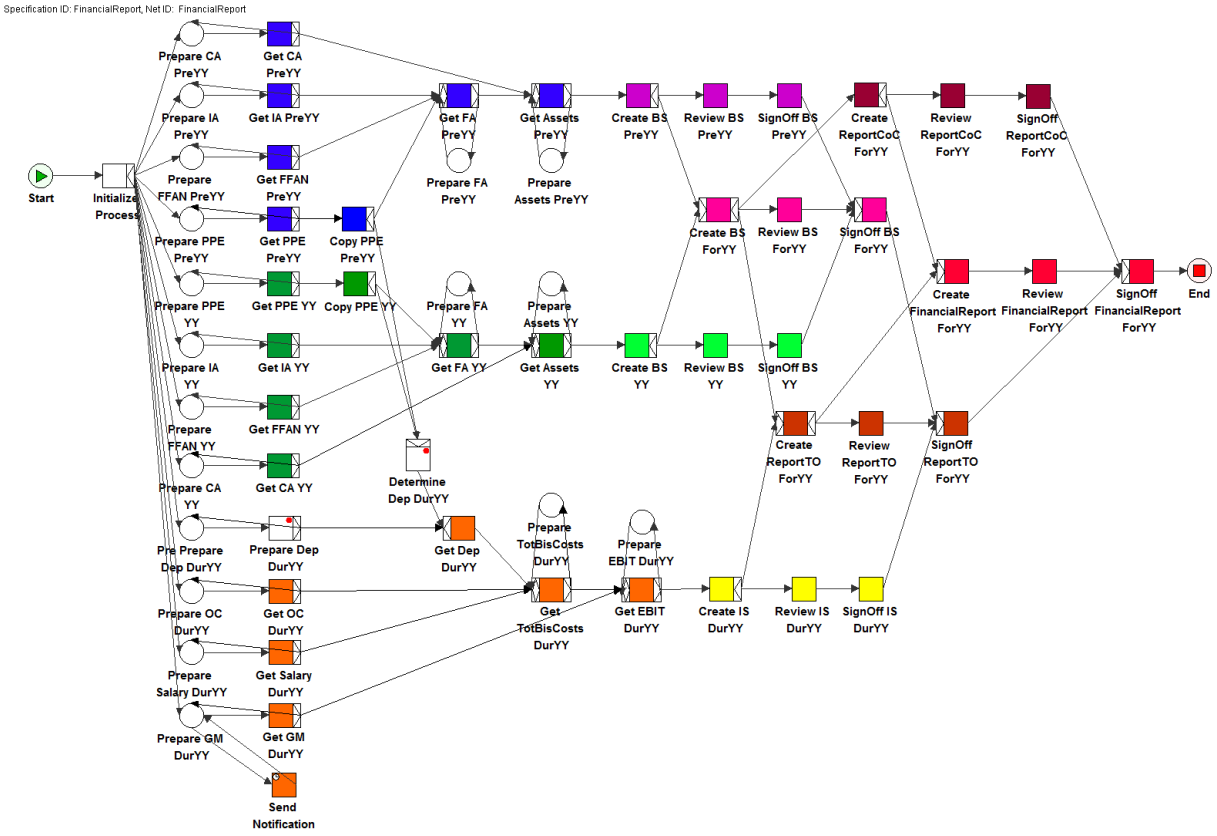


Figure 19 YAWL Model in this prototype

1. Ensure the process to continue only when the needed data is available in the process.

   In the YAWL model with its original layout, every task has one starting task and one ending task. In order to simplify the process model as well as to ensure the process can only continue when the needed data is available, we created a self loop instead of a couple of start and end tasks, for every task to get data element (See Figure 20 and Figure 21). By giving a self loop to each task, conditions can be specified to each task that only when the corresponding variable is "true", can the process continue with next step. Actually, this self loop should be the same with tasks for generating reports, but we did not use self loops for those tasks for the reason of simplicity.

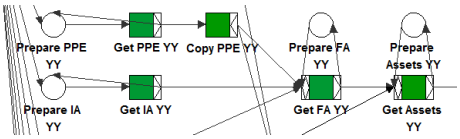


Figure 20 Self-loop in the YAWL model

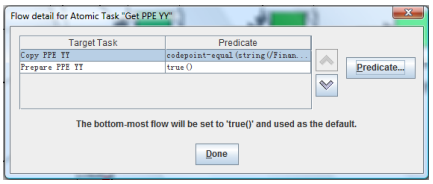


Figure 21 Flow detail for Task "Get PPE_YY"

2. Reuse of data

   How to reuse data is one of the main problems we need to solve for this project. By using YAWL model as the execution model, this problem can be simply solved by using an AND-split task (See Figure 22). In the process model, tokens can be copied as many times as they are needed. For our case, since we need the value of PPE twice, i.e. once for generating the value of Fixed Assets (FA), and once for determine the value of Depreciation (Dep), accordingly, the task “Copy PPE” produces two tokens for the next two tasks.

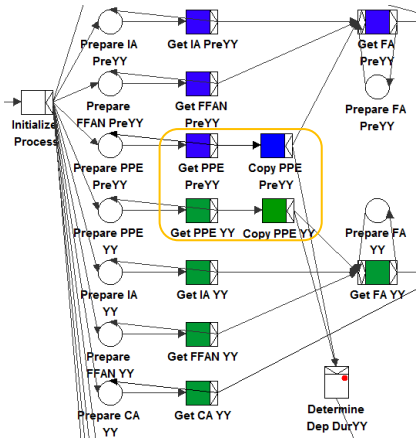


Figure 22 Copy of data in YAWL Model

3. Sub process

   Because it is not possible to specify sub processes in PDM, the converted Petri Nets model and thus the YAWL model do not contain sub process information. However, by manually constructing the YAWL model, it is possible to include sub processes in the YAWL model as shown below (See Figure 23).

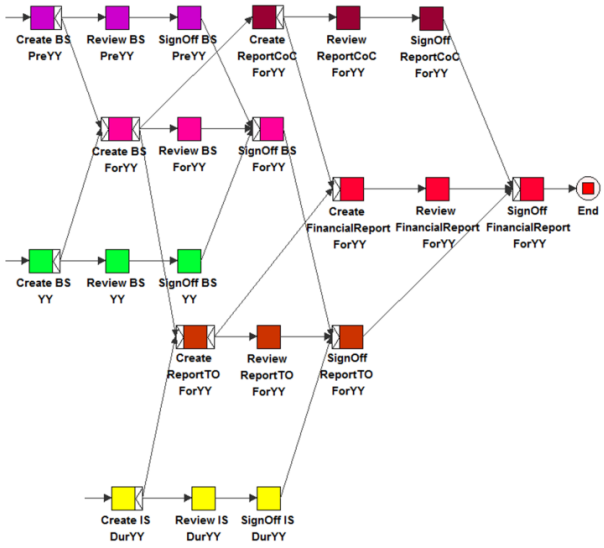


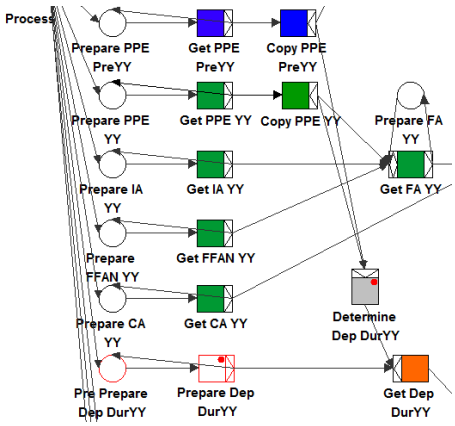
Figure 23 Sub processes for financial reports in YAWL model

The structure shown in the figure above solved Financial Filing Rule 5 (See section 3.2). As the structure shown, we can see that

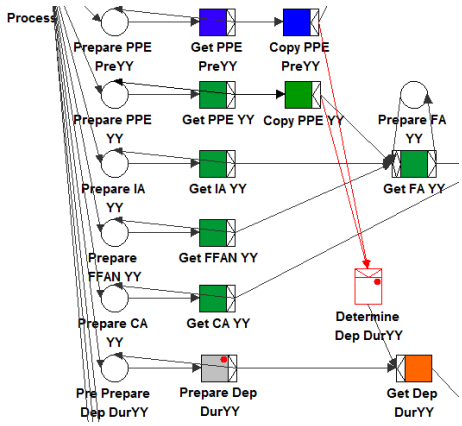
- Higher level reports will be created immediately after all necessary lower level reports are created.
- But higher level reports will only be signed off when all necessary lower level reports are signed off.

4. Remaining tokens

The process model converted from PDM by algorithm Echo has remaining tokens in the process after the process execution is finished. Though this meets the lazy soundness property, it can be easily improved to have strict soundness property by using cancellation regions in YAWL (See figures below).



Cancellation Region 1



Cancellation Region 2

Cancellation regions often appear as pairs. In our case, either task “Prepare Dep DurYY” or task “Determine Dep DurYY” can be done in order to avoid remaining tokens. Thus, these two tasks can include each other in its cancellation regions. After the cancelling task is fired, all tokens in its cancellation regions will be removed. This makes sure that only one in of the two tasks can fire. The table below summarizes cancellation regions used in this YAWL model.

Task	Cancellation Region
Prepare Dep YY	Determine Dep DurYY
	Arc between task “Copy PPE YY” and “Determine Dep DurYY”
	Arc between task “Copy PPE PreYY” and “Determine Dep DurYY”
Determine Dep DurYY	Pre Prepare Dep DurYY
	Prepare Dep DurYY

Table 4 Cancellation Regions in YAWL model

However, the cancellation regions in YAWL work under conditions for our case. Whether the cancellation regions work or not depends on the order of the tasks that are fired. If task “Determine Dep DurYY” is enabled first, the tokens in its cancellation regions can be removed without any problems; if both task “Prepare Dep DurYY” and task “Determine Dep DurYY” are enabled, tokens in their cancellation regions can also be removed without any problems as well; however, if task “Prepare Dep DurYY” is enabled before task “Determin Dep DurYY” is enabled, it can be fired without cancelling any tokens in its cancellation region. In this case, task “Copy PPE YY” and “Copy PPE PreYY” will produce two tokens that can enable task “Determine Dep DurYY” again and one unneeded token will be produced in task “Determin Dep DurYY”.

5. Deadline rule

As listed in the financial filing rules, two of them can be classified as Deadline Rules (Rule 6 and Rule 9). Rule 6 restricts that the value for “Gross Margin” should be available two weeks before the end of the reporting period. Since the workflow engine only works as a management tool of work items, it cannot force the resource to perform a specific task. Thus, we solved this problem by adding a task “Send Notification” to the client as a reminder for them to update the missing data in the process (See Figure 24).

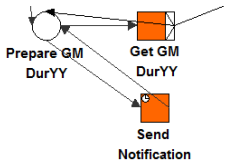


Figure 24 Deadline Rule in YAWL model

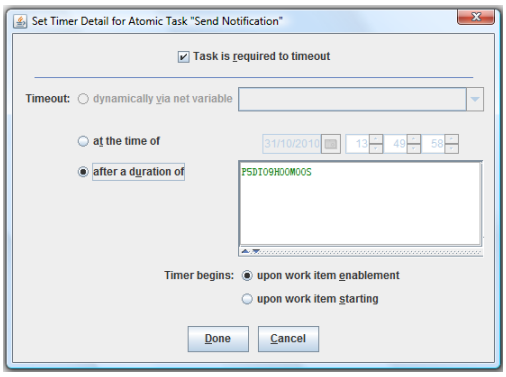


Figure 25 Deadline Rule in YAWL Model - Set Task Timer

We set the task "Send Notification" with a timeout after duration of "P5DT09H00M00S" upon work item enablement (See Figure 25). This means that the "Send Notification" task will be fired automatically at the time of 09:00:00 am, 5 days after it is enabled, i.e. after the process starts in our case.

However, by sending reminders to the client cannot guarantee that the missing data can arrive on time. Because this is a task that needs to be done in the client's workflow management system, thus this task is beyond the control of the workflow system for our project.

5.5.3 Organizational Model

The previous two sections explained the YAWL model from its data perspective and control flow perspective. This section is going to explain the third perspective of the YAWL model, i.e. organizational model. The figure below shows the organizational model in YAWL.

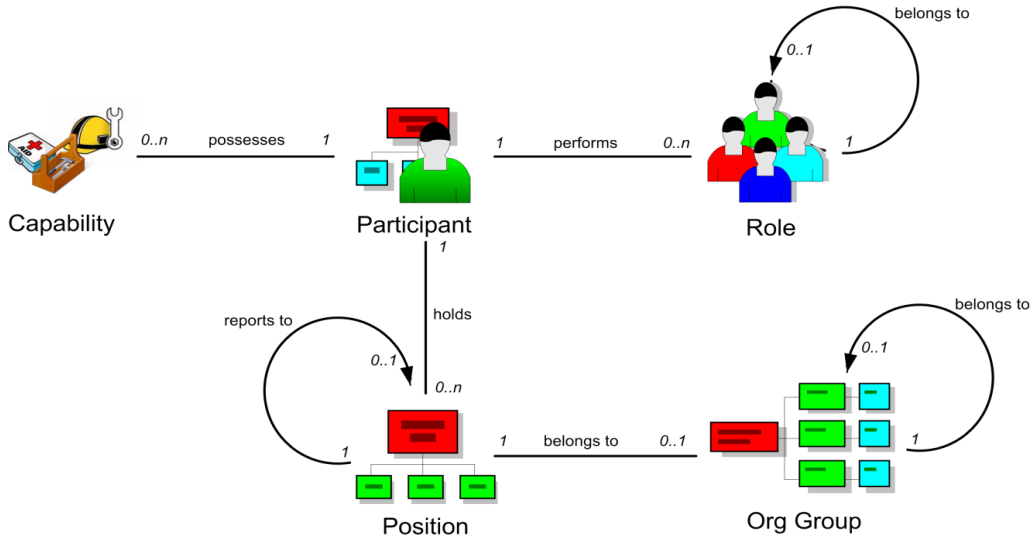


Figure 26 Organizational Model of YAWL

From the figure above, we can see the main part in the organizational model is the participant. One participant can have more than one role and more than one position. And one position can belong

to one or more organizational group. A participant can also have capabilities, but capability is not considered in our project. The table below shows the organizational structure used in the YAWL model (See Appendix for the xml file of the organizational model).

Org Group	Participants	Role	Position
Audit Team	Chris Chang	Creator	Audit Junior Consultant
	Yaqing Sun	Creator, Reviewer	Audit Junior Consultant
	Paul Hulst	Reviewer	Audit Senior Consultant
	Dave van den Ende	SignOff	Audit Director
Tax Team	Stefania Rusu	Creator, Reviewer	Tax Junior Consultant
	Irene Vanderfeesten	Reviewer	Tax Senior Consultant
	Vanessa Restrepo Uribe	SignOff	Tax Director
All	Chris Chang	Creator	Junior Consultant
	Yaqing Sun	Creator, Reviewer	
	Stefania Rusu	Creator, Reviewer	
	Paul Hulst	Reviewer	Senior Consultant
	Irene Vanderfeesten	Reviewer	
	Dave van den Ende	SignOff	Director
	Vanessa Restrepo Uribe	SignOff	Partner

Table 5 Organizational Structure for this prototype

We have seven participants with three different roles, i.e. creator, reviewer, and signoff, in this organization. Seven participants are grouped into two organizational groups, i.e. audit team, tax team. Positions of the participants are named with their team name plus the position.

There are two financial filing rules related with resource information, i.e. Rule 3 and Rule 4 (See section 3.2). For Rule 3, it is a rule that can be specified using Four-Eye Principle in YAWL (See figure below).

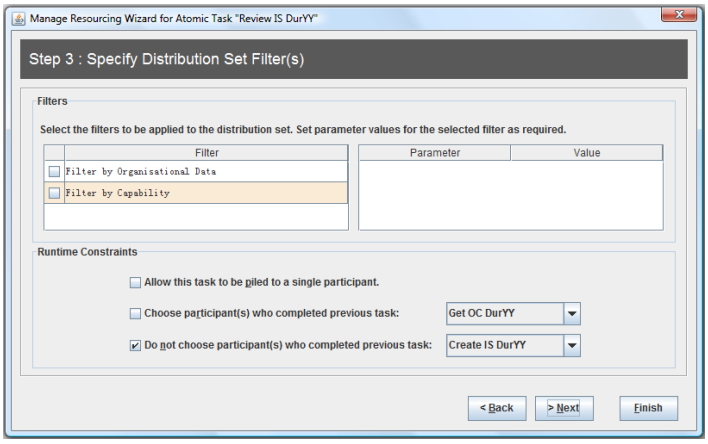


Figure 27 Four-Eye Principle in YAWL model

For Rule 4 it requires that task "SignOff" can either be done by directors or partners. Unfortunately, YAWL can only use one position for one task. However, this rule can be restricted by making a reasonably organizational model. For instance, in our case, both "Directors" and "Partners" have the

role of “SignOff”, thus by assigning the task to the role of “SignOff” can make the task “SignOff” done by the desired resource (See figure below).

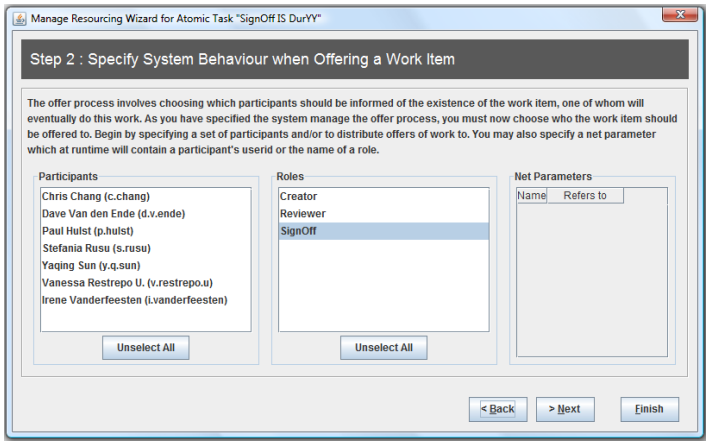


Figure 28 Resource assign rule

Though Rule 4 is not completely met within the YAWL model, at least it proves that it is possible to assign resource based on their role or position. The following figure shows how a task is assigned to a resource by its position in YAWL (See figure below).

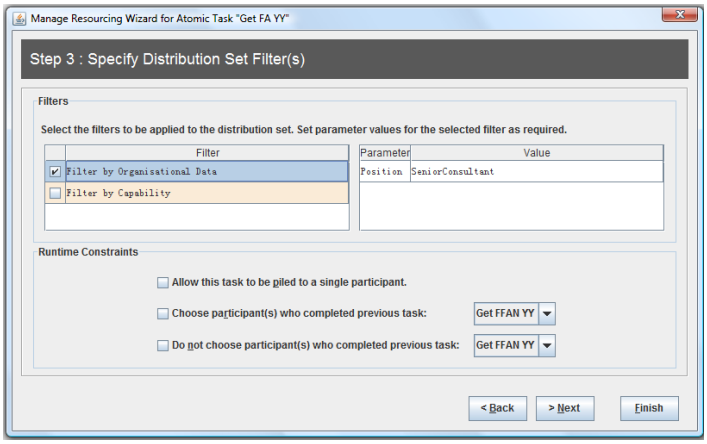


Figure 29 Assign resource by its position

5.6 Log file

There are many different ways to get the process log. The first and most direct way is to check the log from the YAWL engine. A clip of the sample log can be found below.

```
2010-10-31 13:29:50,243 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Prepare_Dep_DurYY_785
2010-10-31 13:29:51,471 [INFO ] ResourceManager    :-  checkout successful: 12:Prepare_Dep_DurYY_785
2010-10-31 13:29:51,637 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Get_CA_PreYY_81
2010-10-31 13:29:51,878 [INFO ] ResourceManager    :-  checkout successful: 12:Get_CA_PreYY_81
2010-10-31 13:29:51,912 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Get_PPE_YY_11
2010-10-31 13:29:52,072 [INFO ] ResourceManager    :-  checkout successful: 12:Get_PPE_YY_11
2010-10-31 13:29:52,161 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Send_Notification_1112
2010-10-31 13:29:52,265 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Get_GM_DurYY_788
2010-10-31 13:29:52,559 [INFO ] ResourceManager    :-  checkout successful: 12:Get_GM_DurYY_788
2010-10-31 13:29:52,653 [INFO ] ResourceMap        :- Resourcing specification parse completed for task: Get_IA_YY_8
```

```
2010-10-31 13:29:52,989 [INFO ] ResourceManager     :-  checkout successful: 12:Get_IA_YY_8
2010-10-31 13:29:53,046 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_FFAN_PreYY_83
2010-10-31 13:29:53,281 [INFO ] ResourceManager     :-  checkout successful: 12:Get_FFAN_PreYY_83
2010-10-31 13:29:53,344 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_CA_YY_9
2010-10-31 13:29:53,601 [INFO ] ResourceManager     :-  checkout successful: 12:Get_CA_YY_9
2010-10-31 13:29:53,653 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_OC_DurYY_786
2010-10-31 13:29:53,864 [INFO ] ResourceManager     :-  checkout successful: 12:Get_OC_DurYY_786
2010-10-31 13:29:53,933 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_PPE_PreYY_85
2010-10-31 13:29:54,268 [INFO ] ResourceManager     :-  checkout successful: 12:Get_PPE_PreYY_85
2010-10-31 13:29:54,318 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_IA_PreYY_79
2010-10-31 13:29:54,518 [INFO ] ResourceManager     :-  checkout successful: 12:Get_IA_PreYY_79
2010-10-31 13:29:54,571 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_Salary_DurYY_787
2010-10-31 13:29:54,793 [INFO ] ResourceManager     :-  checkout successful: 12:Get_Salary_DurYY_787
2010-10-31 13:29:54,878 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_FFAN_YY_10
2010-10-31 13:29:55,188 [INFO ] ResourceManager     :-  checkout successful: 12:Get_FFAN_YY_10
2010-10-31 13:31:27,774 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Get_TotBisCosts_DurYY_793
2010-10-31 13:31:28,297 [INFO ] ResourceManager     :-  checkout successful: 12:Get_TotBisCosts_DurYY_793
2010-10-31 13:32:12,371 [INFO ] ResourceMap         :- Resourcing specification parse completed for task: Determine_Dep_DurYY_791
```

From the log clip above we can see that, the log is not complete, it only shows the time for completing each task without the name of the originator or the change of values.

Another way to get the log file is to use a Java application (14) that can retrieve necessary information from the YAWL Engine. In this way, a more complete log file, e.g. including originator information, can be obtained from the engine. The following list shows a sample clip from the retrieved log file after executing the YAWL model.

```
<event>
        <date key="time:timestamp" value="2010-10-26T19:52:57.190+0200"/ >
        <string key="concept:name" value="Create BS PreYY"/ >
        <string key="lifecycle:transition" value="start"/ >
        <string key="org:resource" value="PA-4e1b40c7-f97c-4301-85f4-dc5d247462ca"/ >
</ event>
<event>
        <date key="time:timestamp" value="2010-10-26T19:53:06.378+0200"/ >
        <string key="concept:name" value="Create BS PreYY"/ >
        <string key="lifecycle:transition" value="suspend"/ >
        <string key="org:resource" value="PA-4e1b40c7-f97c-4301-85f4-dc5d247462ca"/ >
</ event>
<event>
        <date key="time:timestamp" value="2010-10-26T19:53:19.042+0200"/ >
        <string key="concept:name" value="Create BS PreYY"/ >
        <string key="lifecycle:transition" value="resume"/ >
        <string key="org:resource" value="PA-4e1b40c7-f97c-4301-85f4-dc5d247462ca"/ >
</ event>
<event>
        <date key="time:timestamp" value="2010-10-26T19:53:37.142+0200"/ >
        <string key="concept:name" value="Create BS PreYY"/ >
```

<string key="lifecycle:transition" value="complete"/ >

<string key="org:resource" value="PA-4e1b40c7-f97c-4301-85f4-dc5d247462ca"/ >

</ event>

The log above shows how task "BS PreYY" was started, suspended, and finally completed. The log also contains resource information of the task originator. However, the current name of the originator is an unreadable resource ID that needs to be retrieved again in the organizational model. Except for this, all needed information is included in this log. This makes it possible to use this log file for further conversions to the XBRL instance file.

5.7 Demonstration result

The demonstration result can be considered from three points of view. First, we need to check whether we successfully applied all financial filing rules or not. Second, we need to sort out the remaining problems that still need a closer investigation. Third, we need to see to what extend can this demonstration be generalized and reproduced.

First of all, all financial filing rules can be applied in YAWL model. In section 3.2, we classified financial filing rules into two groups, i.e. content-related rules, and process-related rules. Now we are going to look these financial filing rules in a different way. We re-categorized the financial filing rules into three groups, based on the way it was solved. The following list gives a better illustration of the new categorization.

Green: Solved by the process structure
Yellow: Solved by the resource allocation rule
Red: Solved by the deadline rule

Rule 1 **(Between Instant Values)**

Assets = Current Assets + Fixed Assets

Rule 2 **(Between Instant Value and Durational Value)**

Depreciation from 2008/1/1 to 2008/12/31 = PP&E 2008/1/1 – PP&E 2008/12/31

Rule 3 "Create" and "Review" should be done by different resources

Rule 4 "Sign Off" should be done by resource with the position of either "Director" or "Partner"

Rule 5 A higher level concept in the XBRL taxonomy should not be "Signed off" if a lower level concept in the XBRL taxonomy is not "Signed off"

Rule 6 "Current Assets" should be available two weeks before the end of the reporting period for the corresponding fiscal year

Rule 7 "PP&E" should have two instance values, i.e. at the beginning of the fiscal year and at the end of the fiscal year

Rule 8 The duration for data element "Depreciation" should be from the beginning of the fiscal year to the end of the fiscal year

Rule 9 "Depreciation" for a certain period should not be available before the end of that period

(Same with **Rule 9**) "PP&E" at a certain time should not be available before that time

As indicated, rules with green color (Rule 1, 2, 5, 7, 8) are solved by the process structure itself. To be more specific, content-related rules (Rule 1, 2, 7, 8) are used when converting XBRL taxonomy to PDM (see section 5.4). Rule 5 are used when adding sub processes to the manually constructed YAWL model. Rules with yellow color (Rule 3, 4) are solved by applying resource allocation rules in the manually constructed YAWL model (see section 5.5.3). Rules with red color (Rule 6 and 9) are solved by using Task Timer in the YAWL model (see section 5.5.2).

However, even though all financial filling rules were applied in the prototype, there are still some remaining problems that need to be considered. For example, we used cancellation region to remove remaining tokens in the process model. But whether the remaining token is successfully cancelled or not depends on the enabling order of related tasks (see section 5.5.2). Moreover, we used task "Send Notification" to remind the client of the missing data in the process. But this reminder does not guarantee the punctuality of the availability of the missing data (see section 5.5.2). Finally, it is not possible to assign resources with two positions to one task. The only way to use Rule 4 was to create an organizational model that meets the requirements specified in Rule 4 (see section 5.5.3).

Finally, referring to the basic workflow routings that we showed in Chapter 2 (see section 2.1), all of the four basic routings were used in the YAWL model. Thus it should possible to apply our proposed solutions in other workflow languages and workflow management systems vendors. By checking the evaluation result of different workflow management systems vendors on workflow patterns (See Appendix A), you can find what workflow language and vendor can be used to reproduce this prototype.

6 Conclusion and recommendation

In this project, we investigated possibilities to extend XBRL with workflow management system that facilitates financial filing processes. The project is focused on the two problems that are often concerned by the XBRL adaptors, i.e. data assurance and data reusing. Furthermore, we introduced financial filing rules in order to formalize the two problems. The research question proposed for this project is:

> How to enable **data assurance** and **data reuse** to control data gathering in **XBRL based** financial filing process, while respecting **financial filing rules**?

This question was divided into eight more specific questions.

1. How can data validity be ensured to enable reuse? (Chapter 2, section 2.1)

   Audit trail information is needed in order to ensure data validity for financial reports generated using XBRL. To provide this audit trail information, it requires the workflow management systems to record process logs. And to store this information, it requires either the schema of XBRL taxonomy to be extended with Workflow Linkbase, or the XBRL instance file to include workflow information.

2. What workflow technology is most suitable for XBRL based financial filing process? (Chapter 2, section 2.4)

   A proper workflow technique is needed to reuse data and improve the process efficiency. This requires the workflow technique to be flexible when handling choices. And an orientation on data rather than task would be a better match since financial filing process is more like an information flow rather than a sequence of tasks. For this reason, we chose to use Product Data Model (PDM) as the design method in this project.

3. What financial filing rules should be specified in a XBRL based financial filing process? (Chapter 3, section 3.2)

   For most companies, financial filing process does not have a hard copy of pre-defined financial filing rules that can be used as guidebook for every financial filer. In most cases, financial filing work depends highly on the financial filer's working experience and varies because of specific financial filing case. As a result, we consulted financial filing experts within Deloitte and made a sample set of financial filing rules.

4. How to extend XBRL with workflow information? (Chapter 3, section 3.3)

   There are many ways to extend XBRL with workflow information. The conceptual solution proposed for this project is to create Workflow Linkbase using Generic Links. There are several advantages of creating workflow linkbase. First, it enables financial filers to specify their own ideas of creating financial filing rules based on their working experience. Second, it is easier for financial filers to manage the reports when fact figures are separate from audit trail

information. Third, it enables a way to standardize financial filing rules that can be applied within the whole organization.

5. How should PDM be extended to include all necessary information? (Chapter 3, section 3.4)

   In order to store all necessary information as to further execute PDM (if possible), information such as financial filing rules, sub processes, and organizational model should be included in PDM. Similar with question 4, a conceptual solution was proposed for PDM to include these three kinds of information. For sub processes, the operation element in the PDM schema needs to be extended with a task list. For financial filing rules and organizational model, separate files are suggested to store this information in PDM.

6. How to convert PDM from XBRL taxonomy? (Chapter 3, section 3.4)

   The conversion from XBRL taxonomy to PDM in this project mainly focused on the data structure part. Three steps are needed to convert an XBRL taxonomy to a PDM.

7. How to execute the process based on PDM? (Chapter 4)

   Since the execution tool for PDM does not have a complete function for our project, we have to convert a PDM into an executable model first. Based on a group of requirements, we suggest PDM conversion algorithm Charlie, Echo and Delta to get the executable model.

8. How should financial filing rules be applied during execution of financial filing process, using the process model generated from PDM? (Chapter 5)

   A powerful workflow engine is needed to apply financial filing rules while executing the financial filing process model. Since the tool of executing PDM is at prototype level, we changed PDM to an executable process model, i.e. YAWL model, and specified those rules in it. In the YAWL model, content-related rules are obeyed by the nature of the way PDM reserves the data structure; while process-related rules are obeyed by specifying them in the YAWL Engine.

As a proof of concept, this project demonstrated the ideas into a prototype. In the prototype, it proved that is it possible to complete the circle in the project framework (See Figure 1). However, this prototype also has the following limitations due to the assumptions discussed in sub section 5.1.

1. The sample XBRL taxonomy is a really small abstract of a running XBRL taxonomy in practice. The sample XBRL taxonomy contains 15 concepts, while in a running XBRL taxonomy, e.g. the Dutch taxonomy 2010, it contains approximate 2000 concepts. It requires expertise knowledge to understand the complete structure of the taxonomy. Nevertheless, the techniques presented in this thesis are possible to be extended to such large taxonomy.

2. Data values are not perfectly correct in real financial filing process. In this prototype, we assumed that all input values received from clients are correct. This crosses out the satiation when some data values do not match and the financial filing staffs stop the process until correct data values arrive.

3. There is much larger set of financial filing rules than the one used in this prototype. In many companies, e.g. Deloitte, there is no officially documented financial filing rules served as guidelines for auditors to follow during a financial filing process. The financial filing rules used in this prototype can only be seen as a part of the whole rules that is known by the auditors from their filing experience and knowledge.

4. The sub process to generate a financial report is far more complex than three steps as "Create", "Review", and "Sign Off". The reason to consider only these three steps is because they are the milestones in a complete financial filing process. As mentioned above, there are many more detailed steps such as the one mentioned in Limitation 2. Also, in most cases, "Review" step is done more than once in order to ensure the absolute correctness of data values in a financial report. However, this can be solved by introducing a hierarchy in the process model.

5. The requirements for algorithm selecting only applies for the PDM used for this prototype. Even though the selected algorithm works very well for the sample PDM, it is not guaranteed that this set of requirements will also work fine for other PDM, for example PDM with more complex data structure and larger number of data elements.

Based on the limitations, we propose the following recommendations for further work.

1. Study on a larger and more complex, preferably a real taxonomy, and use it as the input for the prototype.

2. Implement all the converting steps that are currently done by hand, i.e. conversion from XBRL taxonomy to PDM, conversion from PDM to an executable process model, conversion from the process log to XBRL instance file.

3. Implement a tool to directly execute PDM.

4. Create a Workflow Linkbase for XBRL taxonomy.

7 Bibliography

1. **Singapore, Deloitte.** What is XBRL? [Online] http://www.deloitte.com/view/en_SG/sg/services/assurance-advisory/8ef70653462fb110VgnVCM100000ba42f00aRCRD.htm.

2. **Organization, XBRL.** About the Organisation. [Online] http://www.xbrl.org/AboutTheOrganisation/.

3. **Wil van der Aalst and Kees van Hee.** *Workflow Management - Models, Methods, and Systems.* Massachusetts London : The MIT Press Cambridge, 2002. 0-262-01189-1.

4. **Bas, Groenveld.** *Assurance on XBRL.* 2010.

5. *Business Process Management: A Survey.* **W.M.P. van der Aalst, A.H.M. ter Hofstede, M. Weske.** Berlin : Springer-Verlag, 2003. International Conference on Business Process Management. Vol. 2678.

6. Welcome to the Workflow Patterns home page. *Workflow Patterns.* [Online] http://www.workflowpatterns.com/.

7. **Vanderfeesten, Irene.** *Product-Based Design and Support of Workflow Processes.* s.l. : Eindhoven University of Technology, 2008. PhD Thesis. 978-90-386-1506-6.

8. XBRL 2.1 Specification. [Online] http://www.xbrl.org/Specification/XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2008-07-02.htm.

9. Generic Links. *XBRL International Organization.* [Online] http://www.xbrl.org/Specification/gnl/REC-2009-06-22/gnl-REC-2009-06-22.html.

10. *Relaxed Soundness of Business Processes.* **K.R. Dittrich, A. Geppert, M.C. Norrie.** Berlin Heidelberg : © Springer-Verlag, 2001. CAiSE 2001. Vol. LNCS 2068, pp. 157–170.

11. UBmatrix XBRL Solutions. *UBmatrix.* [Online] http://www.ubmatrix.com/downloads/UBmatrix_overview.pdf.

12. ProM - the leading process mining toolkit. *ProM.* [Online] http://prom.win.tue.nl/tools/prom/.

13. YAWL: Yet Another Workflow Language. *YAWL.* [Online] http://www.yawlfoundation.org/.

14. **Wynn, Dr Moe Thandar K.** *LoggingJars of YAWL.* Brisbane, Australia : Queensland University of Technology.

Appendix A Evaluations of WFMS vendors on workflow patterns

A.1 Control-flow perspective

1 – BPMN 2 – UML AD 3 – BPEL

		1	2	3			1	2	3
Branching					**Multiple Instances**				
1	Sequence	+	+	+	12	MI without Synchronization	+	+	+
2	Parallel Split	+	+	+	13	MI with a priori Design Time Knlg	+	+	+
6	Multiple Choice	+	+	+	14	MI with a priori Runtime Knlg	+	+	-
4	Exclusive Choice	+	+	+	15	MI without a priori Runtime Knlg	-	-	-
16	Deferred Choice	+	+	+	34	Static Partial Join for MI	+/-	-	-
42	Thread Split	+	+	+/-	35	Cancelling Partial Join for MI	+/-	-	-
Synchronisation					36	Dynamic Partial Join for MI	-	-	-
3	Synchronization	+	+	+	**Concurrency**				
33	Generalised AND-Join	+	-	-	40	Interleaved Routing	+/-	-	+
30	Structured Partial Join	+/-	+/-	-	17	Interleaved Parallel Routing	+/-	-	+/-
31	Blocking Partial Join	+/-	+/-	-	39	Critical Section	-	-	+
32	Cancelling Partial Join	+/-	+	-	18	Milestone	-	-	-
9	Structured Discriminator	+/-	+	-	**Trigger**				
28	Blocking Discriminator	+/-	+/-	-	23	Transient Trigger	-	+	-
29	Cancelling Discriminator	+	+	-	24	Persistent Trigger	+	+	+
7	Str. Synchronizing Merge	+/-	-	+	**Cancellation & Completion**				
37	Local Synchronizing Merge	-	+/-	+	19	Cancel Activity	+	+	+
38	General Synchronizing Merge	-	-	-	20	Cancel Case	+	+	+
5	Simple Merge	+	+	+	25	Cancel Region	+/-	+	-
8	Multiple Merge	+	+	-	26	Cancel MI Activity	+	+	-
41	Thread Merge	+	+	+/-	27	Complete MI Activity	-	-	-
Repetition					**Termination**				
10	Arbitrary Cycles	+	+	-	11	Implicit Termination	+	+	+
21	Structured Loop	+	+	+	43	Explicit Termination	+	+	-
22	Recursion	-	-	-					

A.2 Data Perspective

1 – YAWL 2 – BPMN 3 – BPEL

		1	2	3			1	2	3
Data Visibility					**Data Interaction (External), cont.**				
1	Task Data	+	+	+/-	21	Env. to Case - Push-Oriented	-	-	-
2	Block Data	+	+	-	22	Case to Env. - Pull-Oriented	-	-	-
3	Scope Data	-	-	+	23	Workflow to Env. - Push-Oriented	-	-	-
4	MI Data	+	+/-	-	24	Env. to Workflow - Pull-Oriented	-	-	-
5	Case Data		+	+	25	Env. to Workflow - Push-Oriented	-	-	-
6	Folder Data	-	-	-	26	Workflow to Env. - Pull-Oriented	-	-	-
7	Workflow Data	-	-	-	**Data Transfer**				
8	Environment Data	-	-	+	27	by Value - Incoming	+	+	+
Data Interaction (Internal)					28	by Value - Outgoing	+	+	+
9	between Tasks	+	+	+	29	Copy In/Copy Out	-	+/-	-
10	Task to Sub-workflow Decomp.	+	+	-	30	by Reference - Unlocked	-	-	+
11	Sub-workflow Decomp. to Task	+	+	-	31	by Reference - Locked	-	+	+/-
12	to MI Task	+	-	-	32	Data Transformation - Input	+	+/-	-
13	from MI Task	+	-	-	33	Data Transformation - Output	+	+/-	-
14	Case to Case	-	-	+/-	**Data-based Routing**				
Data Interaction (External)					34	Task Precondition Data Exist.	+/-	+	+/-
15	Task to Env - Push-Oriented	+/-	+	+	35	Task Precondition Data Value	+/-	-	+
16	Env. to Task - Pull-Oriented	-	+	+	36	Task Postcondition Data Exist.	+/-	+	-
17	Env. to Task - Push-Oriented	-	+	+/-	37	Task Postcondition Data Value	+/-	-	-
18	Task to Env - Pull-Oriented	-	+	+/-	38	Event-based Task Trigger	-	+	+
19	Case to Env. - Push-Oriented	-	-	-	39	Data-based Task Trigger	-	+	+/-
20	Env. to Case - Pull-Oriented	-	-	-	40	Data-based Routing	+	+	+

A.3 Resource Patterns

1 – YAWL 2 – BPMN 3 – Oracle BPEL PM

		1	2	3			1	2	3
Creation Patterns					**Pull Patterns (cont.)**				
1	Direct Allocation	+	+	+	24	System-Determ. Work Queue Cont.	+	-	-
2	Role-Based Allocation	+	+	+	25	Resource-Determ. Work Queue Cont.	+	-	+
3	Deferred Allocation	+	-	+	26	Selection Autonomy	+	-	+
4	Authorization	+	-	-	**Detour Patterns**				
5	Separation of Duties	+	-	-	27	Delegation	+	-	+
6	Case Handling	-	-	+	28	Escalation	+	-	+
7	Retain Familiar	+	-	+	29	Deallocation	+	-	+
8	Capacity-based Allocation	+	-	+	30	Stateful Reallocation	+	-	+
9	History-based Allocation	+	-	+/-	31	Stateless Reallocation	+	-	-
10	Organizational Allocation	+	-	+/-	32	Suspension/Resumption	+	-	+
11	Automatic Execution	+	+	+	33	Skip	+	-	+
Push Patterns					34	Redo	-	-	-
12	Distribution by Offer-Single Resource	+	-	+	35	Pre-do	-	-	-
13	Distribution by Offer-Multiple Resources	+	-	+	**Auto-start Patterns**				
14	Distribution by Allocation-Single Resource	+	+	+	36	Commencement on Creation	+	+	-
15	Random Allocation	+	-	+/-	37	Commencement on Allocation	+	-	-
16	Round Robin Allocation	+	-	+/-	38	Piled Execution	+	-	-
17	Shortest Queue	+	-	+/-	39	Chained Execution	+	+	-
18	Early Distribution	-	-	-	**Visibility Patterns**				
19	Distribution on Enablement	+	+	+	40	Config. Unallocated WI Visibility	-	-	-
20	Lata Distribution	-	-	-	41	Config. Allocated WI Visibility	-	-	-
Pull Patterns					**Multiple Resource Patterns**				
21	Resource-Init. Allocation	+	-	-	42	Simultaneous Execution	+	+	+
22	Resource-Init. Exec. - Allocated WI	+	-	+	43	Additional Resources	-	-	+
23	Resource-Init. Exec. - Offered WI	+	-	+					

Appendix B Example PDM

See CD

Appendix C Sample XBRL Taxonomy

C.1 Financial Report.xsd

See CD

C.2 Financial Report-presentation.xml

See CD

C.3 Financial Report-label.xml

See CD

D.4 Financial Report-calculation.xml

See CD

Appendix D PDM converted from sample taxonomy

D.1 FinancialReport - PDM.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with Notepad++ by Yaqing Sun (Technische Universiteit Eindhoven) -->
<PDM xmlns:xsi="http:/ / www.w3.org/ 2001/ XMLSchema-instance" xsi:noNamespaceSchemaLocation="">
	<Name>Financial Filing</ Name>
	<DataElement DataElementID="FR_ForYY"/ >
	<DataElement DataElementID="TO_ForYY"/ >
	<DataElement DataElementID="CoC_ForYY"/ >
	<DataElement DataElementID="BS_ForYY"/ >
	<DataElement DataElementID="IS_ForYY"/ >
	<DataElement DataElementID="BS_PreYY"/ >
	<DataElement DataElementID="BS_YY"/ >
	<DataElement DataElementID="Assets_PreYY"/ >
	<DataElement DataElementID="Assets_YY"/ >
	<DataElement DataElementID="FA_PreYY"/ >
	<DataElement DataElementID="FA_YY"/ >
	<DataElement DataElementID="CA_PreYY"/ >
	<DataElement DataElementID="CA_YY"/ >
	<DataElement DataElementID="IA_PreYY"/ >
	<DataElement DataElementID="IA_YY"/ >
	<DataElement DataElementID="FFAN_PreYY"/ >
	<DataElement DataElementID="FFAN_YY"/ >
	<DataElement DataElementID="PPE_PreYY"/ >
	<DataElement DataElementID="PPE_YY"/ >
	<DataElement DataElementID="EBIT_DurYY"/ >
	<DataElement DataElementID="TotBisCosts_DurYY"/ >
	<DataElement DataElementID="GM_DurYY"/ >
	<DataElement DataElementID="Dep_DurYY"/ >
	<DataElement DataElementID="Sa_DurYY"/ >
	<DataElement DataElementID="OC_DurYY"/ >
	<Resource ResourceID="w1"/ >
	<Operation OperationID="CreateFR_ForYY">
		<Input>
			<DataElementRef>CoC_ForYY</ DataElementRef>
			<DataElementRef>TO_ForYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>FR_ForYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="CreateReportForCOC">
		<Input>
			<DataElementRef>BS_ForYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>CoC_ForYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="CreateReportForTO_ForYY">
		<Input>
			<DataElementRef>BS_ForYY</ DataElementRef>
			<DataElementRef>IS_ForYY</ DataElementRef>
```

```
		</ Input>
		<Output>
			<DataElementRef>TO_ForYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CreateBS">
		<Input>
			<DataElementRef>BS_PreYY</ DataElementRef>
			<DataElementRef>BS_YY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>BS_ForYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateBS_PreYY">
		<Input>
			<DataElementRef>Assets_PreYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>BS_PreYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateAssets_PreYY">
		<Input>
			<DataElementRef>FA_PreYY</ DataElementRef>
			<DataElementRef>CA_PreYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>Assets_PreYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>

<Operation OperationID="GetCA_PreYY">
		<Input / >
		<Output>
			<DataElementRef>CA_PreYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateFA_PreYY">
		<Input>
			<DataElementRef>IA_PreYY</ DataElementRef>
			<DataElementRef>FFAN_PreYY</ DataElementRef>
			<DataElementRef>PPE_PreYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>FA_PreYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
```

```
</ Operation>
<Operation OperationID="GetIA_PreYY">
        <Input / >
        <Output>
                <DataElementRef>IA_PreYY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetFFAN_PreYY">
        <Input / >
        <Output>
                <DataElementRef>FFAN_PreYY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetPPE_PreYY">
        <Input / >
        <Output>
                <DataElementRef>PPE_PreYY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateBS_YY">
        <Input>
                <DataElementRef>Assets_YY</ DataElementRef>
        </ Input>
        <Output>
                <DataElementRef>BS_YY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateAssets_YY">
        <Input>
                <DataElementRef>FA_YY</ DataElementRef>
                <DataElementRef>CA_YY</ DataElementRef>
        </ Input>
        <Output>
                <DataElementRef>Assets_YY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetCA_YY">
        <Input / >
        <Output>
                <DataElementRef>CA_YY</ DataElementRef>
        </ Output>
        <ResourceRef>w1</ ResourceRef>
        <Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateFA_YY">
        <Input>
                <DataElementRef>IA_YY</ DataElementRef>
                <DataElementRef>FFAN_YY</ DataElementRef>
                <DataElementRef>PPE_YY</ DataElementRef>
```

```
		</ Input>
		<Output>
			<DataElementRef>FA_YY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetIA_YY">
		<Input / >
		<Output>
			<DataElementRef>IA_YY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetFFAN_YY">
		<Input / >
		<Output>
			<DataElementRef>FFAN_YY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetPPE_YY">
		<Input / >
		<Output>
			<DataElementRef>PPE_YY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateIS_ForYYDurYY">
		<Input>
			<DataElementRef>EBIT_DurYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>IS_ForYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="CalculateEBIT_DurYY">
		<Input>
			<DataElementRef>TotBisCosts_DurYY</ DataElementRef>
			<DataElementRef>GM_DurYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>EBIT_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
</ Operation>
<Operation OperationID="GetGM_DurYY">
		<Input / >
		<Output>
			<DataElementRef>GM_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
```

```
	</ Operation>
	<Operation OperationID="CalculateTotBisCosts_DurYY">
		<Input>
			<DataElementRef>Dep_DurYY</ DataElementRef>
			<DataElementRef>Sa_DurYY</ DataElementRef>
			<DataElementRef>OC_DurYY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>TotBisCosts_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="GetDep_DurYY">
		<Input / >
		<Output>
			<DataElementRef>Dep_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="CalculateDep_DurYY">
		<Input>
			<DataElementRef>PPE_PreYY</ DataElementRef>
			<DataElementRef>PPE_YY</ DataElementRef>
		</ Input>
		<Output>
			<DataElementRef>Dep_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="GetSa_DurYY">
		<Input / >
		<Output>
			<DataElementRef>Sa_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<Operation OperationID="GetOC_DurYY">
		<Input / >
		<Output>
			<DataElementRef>OC_DurYY</ DataElementRef>
		</ Output>
		<ResourceRef>w1</ ResourceRef>
		<Condition>true</ Condition>
	</ Operation>
	<RootElementRef>FR_ForYY</ RootElementRef>
</ PDM>
```

Appendix E YAWL file

E.1 FinancialReport .yawl

See CD

E.2 OrganizationalModel.ybkp

See CD