Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2007 Proceedings

European Conference on Information Systems (ECIS)

2007

XBRL Taxonomy Engineering. Definition of XBRL Taxonomy Development Process Model

Maciej Piechock
TU Bergakademie Freiberg, maciej.piechocki@bwl.tu-freiberg.de

Carsten Felden
TU Bergakademie Freiberg, carsten.felden@bwl.tu-freiberg.de

Follow this and additional works at: http://aisel.aisnet.org/ecis2007

Recommended Citation
Piechock, Maciej and Felden, Carsten, "XBRL Taxonomy Engineering. Definition of XBRL Taxonomy Development Process Model" (2007). *ECIS 2007 Proceedings*. 1.
http://aisel.aisnet.org/ecis2007/1

This material is brought to you by the European Conference on Information Systems (ECIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

XBRL TAXONOMY ENGINEERING. DEFINITION OF XBRL TAXONOMY DEVELOPMENT PROCESS MODEL.

Piechocki, Maciej, Felden, Carsten, Technische Universität Bergakademie Freiberg, Lessingstraße 45, 09599 Freiberg, Germany,
{maciej.piechocki|carsten.felden}@bwl.tu-freiberg.de

Abstract

The growing number of eXtensible Business Reporting Language (XBRL) projects around the world and strong interest from bodies such as Security Exchange Commission in the United States (SEC), Central European Banking Supervisors in the European Union (CEBS) or International Accounting Standards Board (IASB) in building XBRL taxonomies demonstrate the need for formalisation and methodical approach to the process of the XBRL taxonomy development. Although many approaches exist in favour of software engineering and knowledge engineering, building an XBRL taxonomy is not about creating a software product or a knowledge-based system. It is creating a standardised taxonomy for a particular domain in order to enable standardised exchange of business reports. Nevertheless experiences learned from software and knowledge engineering areas are very useful for what can be called XBRL taxonomy engineering. On the other hand a clear parallel with the ontology engineering appears treating XBRL taxonomies as ontologies. The ontology development process could resemble in many aspects the way that XBRL taxonomies are created. The development process models presented in the literature are either project or software driven. Hence it is difficult to apply them as a generic and formal taxonomy development process model. This paper presents an approach to define a taxonomy development process model. The definition of the model is preceded with the status quo analysis of the existing development models known from software engineering and ontology engineering domains. The model definition itself is also based on empirical analysis of taxonomy development projects.

Keywords: XBRL, taxonomy development, software engineering, ontology engineering.

1 INTRODUCTION

The financial reporting world has faced a number of changes in recent years. The Internet with XML standards and especially eXtensible Business Reporting Language (XBRL) has impacted what is recognised as the financial reporting supply chain (FRSC) (Romney et al. 2006, p. 532). Some claims in the market report XBRL to reduce inefficiencies, automate and optimise the FRSC (Hannon 2004, p. 55, Romney et al. 2006, p. 530, Nutz et al. 2002, p. 457). But the real nature of the impact remains unclear (Sutton 2006, p. 3). Important aspect for XBRL market acceptance is the development of XBRL taxonomies in order to standardise various business reporting domains (Hoffman 2006, p. 66). The analysis of the development processes of many significant taxonomies as well as project based experience[1] provides a common knowledge - the taxonomy development requires a lot of time and resources and is often finished beyond the defined deadlines. The International Financial Reporting Standards (IFRS) taxonomy of the International Accounting Standards Committee Foundation for the reporting year starting on 01 January 2006 was released on 15 August 2006. The German accounting principles taxonomy of the XBRL Germany based on the XBRL 2.1 specification released on 31 December 2003 is still under development. The FINREP taxonomy of the Central European Banking Supervisors based on the regulations being in place at the beginning of 2006 was released in September 2006. This leads to the obvious consideration to support the project management by the development process by using a taxonomy development process model. However neither XBRL International[2] feels any responsibility for the development of such an approach nor provides the academic community formal and generic process models for the taxonomy development. The existing best practices released by XBRL International included in the Financial Reporting Taxonomies Architecture (FRTA) document merely indicate the modelling rules to be followed. But it has to be stated that the development process as such is neither shown nor discussed. It seems to be a first solution to solve this problem by using already available development models. The most significantly applying to the business reporting standardisation domain are the approaches known from software engineering as well as ontology engineering with the latter placed in the knowledge engineering context. But it needs to be clearly stated that XBRL taxonomy development is not equal to the development of software systems or knowledge-based systems. The reason is the nature of a taxonomy being a standardisation of a reporting domain in form of metadata and not in form of a software product. Moreover a taxonomy is usually later implemented in software products as a way to describe the metadata according to which a report must be constructed. Important here is the fact that the development process of the XBRL taxonomy is usually strictly separated from an organisational point of view from the development process of the software in which the taxonomy is implemented. But the existing knowledge in the area of phase-oriented software development as well as agile software development models can be transferred to the XBRL taxonomy development. Even closer correlation can be found between ontology engineering and XBRL taxonomy development. Nevertheless the XBRL framework is based on taxonomies as metadata and instance documents as data of the reports. This distinction cannot be found in the ontologies domain. Moreover in this case the analysis of the most common approaches to ontology development delivers mainly project, ontology or ontology editing software oriented development process models. It is questionable how to apply one of the ontology development models as an abstract and generic model for the XBRL taxonomy development. But the project experiences derived from the ontology engineering domain supports the definition of a

[1] The authors of the paper are or were actively involved in the development of the IFRS taxonomy, German accounting principles taxonomy, Polish accounting principles taxonomy, as well as in review processes of the United States Generally Accepted Accounting Principles taxonomy (US GAAP), Common Reporting (COREP) and Financial Reporting (FINREP) taxonomies as well as Global Common Data (GCD) taxonomy.

[2] XBRL International is an umbrella organisation of over 450 institutions and companies worldwide supporting the XBRL standard development.

XBRL taxonomy development process model. Due to the reason that the analysis and presentations of all existing models in software and ontology engineering is not possible, we discuss the most known approaches of both engineering fields in the second chapter. Combined with project based experience the induction process is being conducted. The result in a form of a generic and formal taxonomy development model is proposed in the third chapter. The model is critically verified using the IEEE as well as universal usage criteria and compared to the software engineering and ontology engineering approaches. The paper is summarised with conclusions about the possible implementation areas of the designed model as well as further research opportunities in the described field.

2 SOFTWARE AND ONTOLOGY ENGINEERING

This chapter conducts an analysis of existing software engineering and ontology engineering approaches. The chapter is summarised with the comparison of both domains in a graphical form using relevant criteria. Although ontology engineering, being knowledge representation is often recognised as knowledge engineering approach (Studer 1998) they are both characterised by a number of differences. Ontology engineering approaches address issues of creating an ontology without addressing specific issues of development of knowledge-based systems. Therefore in the context of this paper only ontology development approaches are analysed as being more relevant from the domain modelling point of view.

2.1 Software Engineering

Due to the complexity of realising software, researchers like Sommerville and Balzert pointed out that the development process should be divided into clear phases in order to define a process model. Process model is the term generally used for the development plan of a software product (Balzert 1998, p. 71). Such a model defines the workflow of the processes and corresponds with the overall strategy defining the order of the development steps. More precisely the model defines the activities that need to be performed at each step as well as their order together with the organisational framework e.g. used standards, guidelines, procedures, methods, and tools (Balzert 1998, p. 98). Using the process models, the development of complex software systems can be conducted in an effective and efficient way so the software development process is controllable and manageable (Fink et al. 2001, p. 166). In the area of software development the literature provides a number of process models according to various criteria and presented in different variations (Fairley 1985, Sommerville 2001 and Pomberger et al. 1996). The development process can be divided into six phases (Pomberger et al. 1996, p. 17) which are called planning and analysis, specification, system and components design, implementation and components test, system test and operation and maintenance. These phases are recognised as software life cycle.

The software engineering literature provides a distinction between two categories of process models (Dumke 2001, p. 103). The first group are sequential models with relatively strong phase sequence. The strong sequence can be recognised when a following phase can be only started after the previous phase is completely finished. The classical waterfall model is an example of this category. All the software life cycle phases are combined in a natural way. Additionally there is a validation step after each phase in the development cycle. Therefore practical experiences lead to the conclusion that a feedback between different phases is necessary because of technical and economical reasons. This leads to the modified waterfall model (Ludewig et al. 2007, p.176). Firstly the sequence is not as strong and feedback between different phases is allowed. The advantage of the waterfall model is its simplicity and minor coordination effort. According to the strong phases' order and the feedback between them the development steps must be conducted in a complete and sequential manner (Balzert 1998, p. 101). The above statement indicates the need for the risk evaluation for the whole project before project development (Zuser et al. 2001, p. 46).

The second category of process models is non-sequential models (cyclic models) which, per definition, enable feedback between phases (Sommerville 2001, p. 63). A significant example of this category is called prototyping. It was developed in order to solve the issues appearing during the realisation of the requirements defined in the specification phase as well as enabling the realisation of various solution possibilities during software development (Balzert 1998, p. 114, Fink et al. 2001, p. 169). A prototype demonstrates specific characteristics of the end product according to the practical application and is only a rudimental executable version. The major requirements and functionalities can be recognised discussing the prototype and can be implemented in forthcoming versions. Prototypes are also useful for gathering the first practical experiences and feedback. There are a number of prototypes used with various aims. The most common examples are the demonstration prototype, prototype in the narrow sense, laboratory prototype, and pilot system (Fink et al. 2001, p. 170). Prototyping is often not recognised as a complete process model. It leads to the assumption that prototypes can be constructed in a reasonable timeframe and this assumption cannot be validated, because it strongly depends on the kind of the realized prototype.

The above presented models are just examples from the rich domain of software engineering nevertheless they represent most important features of this area and allow relevant conclusions to be drawn for the taxonomy engineering described in the third chapter.

2.2 Ontology Engineering

Although ontology development is comparable with software development life cycles, special requirements of ontologies have to be kept in mind. In the recent years, numerous suggestions were made about how to develop an ontology. Existing ontology development process models are either not generic or refer to a specific domain or application (Staab et al. 2001). The modeller has to keep in mind that the ontology development assumes different prerequisites and objectives compared to software development.

The Methontology approach which constitutes a universal knowledge level ontology development procedure model, was published by Fernandez-Lopez, Gomez-Perez and Juristo in 1997 (Fernández-López 1999). Methontology is a comprehensive ontology development methodology according to the IEEE-norm which describes the activities of a software development process and of knowledge management. The life cycle of ontology development is based on iterative enhancements of a developed prototype, so that something can be added in each new version. This can be element modification or element elimination. The activities of the ontology development process are divided into three categories: project management activities, development activities and supporting activities. Besides the point that development activities describe the procedure of ontology construction in detail and technical activities concern the project management including planning, control, and quality assurance, they have to be distinguished from the accompanying supporting activities. These activities are divided into knowledge acquisition, integration, evaluation, documentation, and configuration management. This methodology was applied to develop ontologies and applications in different domains (Corcho et al. 2003).

The On-To-Knowledge project is concentrated on a procedure model which aids the design of an ontology based knowledge management system. Ontologies and corresponding tools respectively enable access to semi-structured or textual information in such systems. The On-To-Knowledge procedure model consists of the following phases: feasibility study, kickoff-phase, refinement, evaluation, and maintenance (Sure 2002).

The feasibility study should be prior to the beginning of an actual development process to identify chances and risks and to analyse the primary application areas using the CommonKADS methodology. This methodology covers specific aspects of knowledge based systems using manifold models (Schreiber et al. 2000). The results of this phase are the basis for the kickoff-phase. The created application specification contains the specified domain, the objective, design directives, available

resources, and potential users. Subsequent competence questions are formulated in order to collect domain specific terms in an informal manner.

The main focus of the TOVE (Toronto Virtual Enterprise) methodology, created by Grüninger and Fox (1995), is to provide a series of competence questions. Questions on the problems that have to be solved are formulated and should be answered afterwards by the ontology. They are used in order to build the concept hierarchy and to evaluate the ontology. Prerequisites for the usage of competence questions are the availability of domain experts and the support of the chosen tool.

The SENSUS approach was introduced by Swartout and covers just domain ontologies (Swartout 1996). The initial point is the SENSUS ontology itself. This ontology represents an extensive ontology including 70,000 domain independent concepts. Representative concepts of this domain are selected and manually linked with the SENSUS ontology in order to create a domain specific ontology. Afterwards, all concepts are inserted which are located directly at the path from the specific terms to the root. Further and so far not incorporated but potentially useful concepts are included manually. The remaining SENSUS concepts are discarded as irrelevant.

The KACTUS (1995) approach was developed with the scope of the Esprit-project. It postulates already existing ontologies which are reused or customized in order to create a new one. First of all the applications, thus the relevant concepts and objectives, are specified. A new ontology is developed by adjusting and refining the already existing top-level or reusable ontologies. It has to be stated that the existence of reusable ontologies is a necessary assumption.

2.3 Comparison of Software and Ontology Engineering

From the above overview it can be stated that the existing approaches to the ontology engineering domain are project or software driven. There is a lack of generic models which are more often in the area of software development. A classification of development approaches is presented in figure 1. We differentiate ontology and software engineering approaches according generic or specific usage. The approaches are classified according to their brief description stated above.

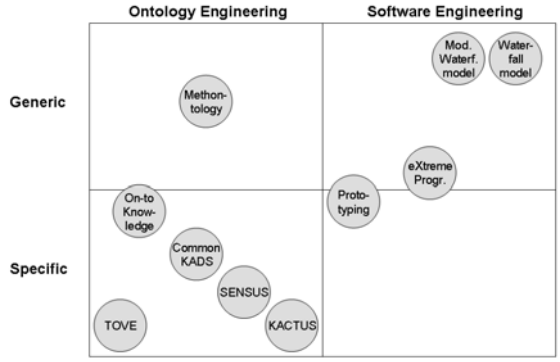

Figure 1. Classification of development approaches

For example, the waterfall model and the modified waterfall model are absolute generic software engineering approaches, because they are the basis for most of the software engineering projects; TOVE is a very specific approach for the ontology development. According to the idea of prototyping and the related eXtreme Programming, these approaches are classified between generic and specific.

The observation of software engineering and ontology engineering development models is presented in figure 2. First conclusion coming from the above analysis is that the software engineering models are mostly generic whereas ontology engineering models are based on a specific ontology or software. Software engineering focuses mostly on the phases approach while ontology development models are not stressing the phases and their order so strong. Due to the weak emphasis of the phases order, the

feedback between different phases is not part of the models either. Also the test phases is stressed much stronger in software development process models and actually omitted in ontology engineering. The result of the software engineering is a software product and in ontology engineering an ontology or ontology together with a knowledge-based system. The comparative analysis provides the conclusion that software engineering and ontology engineering approaches are not appropriate in context of XBRL taxonomy engineering. Main reasons for this are strong phase orientation observed during taxonomy development projects not strongly supported in the area of ontology engineering. On the other hand the knowledge transfer from the subject matter experts to the technical experts having significant role while constructing XBRL taxonomies is not discussed in the software development models. The taxonomy engineering needs be analysed separately and build up on various criteria derived from both of the analysed engineering approaches.

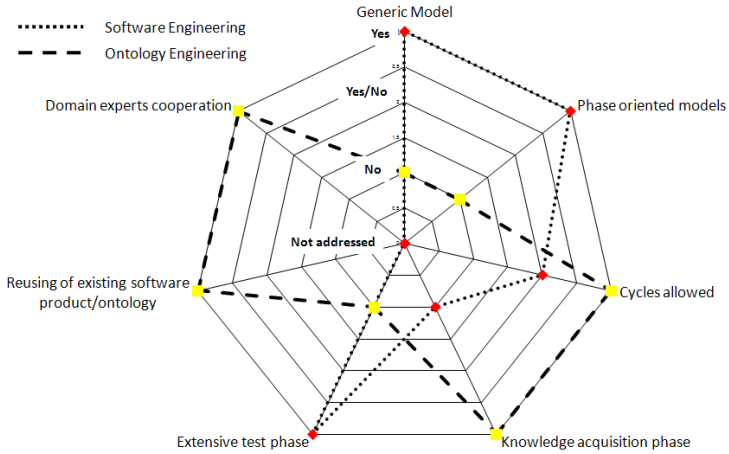

Figure 2. Comparison of software and ontology engineering

3 XBRL TAXONOMY ENGINEERING

XBRL taxonomy development can be regarded as a transfer of the domain knowledge from a domain expert into an implemented knowledge base which is encoded within an XBRL taxonomy. It leads to the conclusion of applying one of many ontology development models for the XBRL taxonomy development. As already indicated, ontology engineering approaches result from certain projects and are absolutely software related. The analysis of the existing XBRL taxonomy development projects indicates a clear taxonomy life cycle as well as the possibility to clearly recognise defined phases during the process. The proposed approach analyses the taxonomy life cycle in order to recognise and classify the different phases of the taxonomy development. The phases are ordered and integrated in a process model. The process model as such is based on the incremental software development model combined with prototyping. However the internal activities within each phase are based on experiences derived from ontology engineering approaches. The introduced process model is critically reviewed by comparing it to software engineering and ontology engineering approaches. Finally on the basis of these critical considerations the term the taxonomy engineering is addressed.

3.1 XBRL Taxonomy Process Model

XBRL taxonomy development is strongly related to ontology engineering because of the knowledge conceptualisation and specification in case of both. However the analysed approaches to the ontology development are very project and software specific. Therefore it is not possible to apply them as a generic XBRL taxonomy development process model. The term generic means applicable to all kinds of financial reporting taxonomies worldwide and not being specific to accounting standards, software

used or conducted project. The approach presented in this part of the paper is organising the taxonomy life cycle phases in a process model. In many respects it resembles the rigorous and formal software development process than knowledge or ontology development process. The main reason for that are the existing generic approaches to the software engineering and established process models which can be adapted in the XBRL taxonomy development environment.

3.2 XBRL Taxonomy Lifecycle

A correspondence between software development processes and taxonomy development is observed mainly in the phase structure as well as document-oriented approach. Neither the literature nor the documentation of existing XBRL taxonomies provides explicit phases for the taxonomy development3. But looking at XBRL taxonomies developments all over the world, clear phases can be identified. Taxonomy development starts with planning and analysing, followed by design. Later, the taxonomy building stage finishes the development with testing, publication, and recognition. The final phase is taxonomy usage and maintenance.

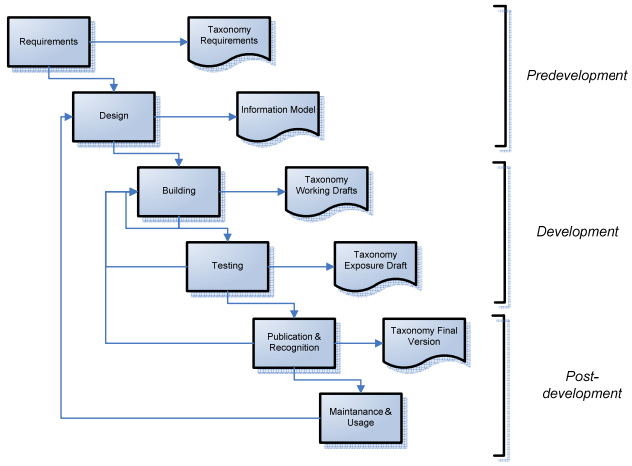

Figure 3. Taxonomy development process model

The conducted research and the results from the chapter 2 together with the observation and cooperation in a number of XBRL taxonomy development projects allow constructing the process model for the taxonomy development. The process model presented in figure 3 reflects in its principal construction the phase model with the possibility of feedback between the phases. Similarly to software and ontology engineering approaches there are clear predevelopment, development and post-development parts to be recognised within the model. The single phases are the requirements, design, building, testing, publication and recognition as well as maintenance and usage. At the end of each phase there is a formal document or set of files produced. The main feedback cycles are allowed within the building phase while creating a number of taxonomy working drafts, between the testing and building phase as well as between recognition and building phase. Also at the very end in the maintenance and usage phase the feedback provided leads the changes in the taxonomy information model and implies development of a new taxonomy version.

The XBRL taxonomy development starts with the *planning and analysis* phase where the taxonomy development project requirements are defined and considered. The first aspect of the analysis is to state which kind of taxonomy is developed. This is normally referred to the taxonomy's scope. XBRL

[3] Although the XBRL International released a document called Taxonomy Life Cycle the issue that this document concerns is merely the XBRL taxonomy versioning and taxonomy development process as such is not discussed (Hernández-Ros et al. 2006).

taxonomies can represent core accounting standards such as German GAAP or International Financial Reporting Standards. It can be a national or industry extension to these standards as well as company specific taxonomies. The core taxonomy compromises knowledge included in a single and stand-alone accounting or reporting standard or regulation. The jurisdictional taxonomy extension is a taxonomy using the knowledge base included in the core taxonomy and usually extends it for a certain geographical region. The industry taxonomy extension is enhancing a taxonomy with industry specific aspects, analogue. Finally, the reporting company itself can build an own extension taxonomy including individual reporting aspects. The requirement, which taxonomy is to build and, in case of an extension taxonomy, which taxonomy is to use as a basis, should be defined in an early stage of the planning and analysis phase. The resulting activity is the design of the taxonomy framework. A taxonomy framework can be defined as a way of combining different taxonomies for a jurisdiction or industry. A lot of European taxonomies are part of a sophisticated framework which demonstrates the importance of this activity[4]. How a taxonomy framework is constructed influences the taxonomy architecture which is the way of modelling and building single taxonomies. During the development of XBRL taxonomies the knowledge from different sources such as accounting standards or other legal documents, model financial statements, real financial statements, have to be gathered. These sources need to be clearly stated in the requirements for the taxonomy before the design phase starts. The requirements documentation as a result of the first phase should be part of the taxonomy documentation clarifying the actions taken which is learned from software engineering.

Similar to the knowledge transfer process known from ontology engineering, the accounting knowledge from domain experts has to be implemented in the knowledge base during the *taxonomy design* phase. Especially the crucial role of the task division between domain and technical experts must be stated very clearly at this stage. Observing XBRL taxonomy development projects there always exists a mixture of domain and technical experts working together to establish the taxonomy. There exist similar considerations since the introduction of the Structured Systems Analysis and Design Method (SSADM) where the following stages drive from the business view on the information system, through the logical design to the physical design and close cooperation of the business and technical experts is required (DeMarco 1978, p. 78). But we have learned that the approaches of each group are different. The domain experts prefer expressing their knowledge in a structured way. The technical experts prefer to formalise the structured knowledge directly in XBRL code while modelling the taxonomy. The danger of this approach is the transfer process itself, where the structured knowledge may not be appropriate to the restrictions of the XBRL standard. Studer et al. (1998) indicates another problem within the modelling process concerning the subjectivity of the domain experts while modelling the reality. Both issues can lead to a disruption in the modelling process. Due to the reason that there is no formal XBRL modelling notation for the taxonomies development, yet, the common approach is using spreadsheets for the representation of the knowledge base. A so called information model is created with possible structures that need to be encoded in an XBRL taxonomy, later. The information model can vary from just representation tasks of the reports in form of tables (FINREP taxonomy) to a very detailed description of each attribute that needs to be included in the taxonomy (German GAAP). The lack of a common modelling approach in XBRL and a formal notation is a major drawback of the standard. With the introduction of the multidimensional XBRL taxonomies (Hernández-Ros and Wallis 2005) the modelling aspects are getting more advanced and the modelling aspects are more important. The result of the design phase is information model.

The *taxonomy building* phase begins after the development of the information model by domain experts. The technical knowledge which is necessary to build an XBRL taxonomy and therefore the building phase requires XBRL technical experts. There are XBRL taxonomy editing tools already available, but the taxonomy building is still a sophisticated task. Similarly to the previous phase the

[4] Examples for the European taxonomies where the frameworks are available are the Dutch taxonomy, the COREP taxonomy and the IFRS taxonomy.

potential risk is the transfer of the knowledge base from the information model into the coded structures of the XBRL taxonomy. This transfer process can result in changes or even losses in semantic of the constructed taxonomy. XBRL International has released a document called FRTA with established rules and conventions that assist in a comprehensive manner the usage and performance among different financial reporting taxonomies. Moreover Hoffman (2006) describes 25 so called taxonomy patterns which define the possible structures appearing in financial reports and their representation in taxonomy code. Taxonomy developers are free to encode the information model according to individual rules due to the reason that patterns are not a formal notation. There is a number of so called taxonomy working drafts to support the visibility of the taxonomy building phase. After each working draft the feedback is being included in the information model. Another option is that the next working draft fulfils the requirements. Initial taxonomy working drafts are usually not using the full functionality of XBRL, but represent certain aspects of the information model (IASCF 2006). The usage of available taxonomy editors supports the testing of the taxonomy working drafts at the end of each development cycle. Although extensive test are rarely conducted in the building phase, the tools ensure XML and XBRL specifications compatibility.

Extensive tests are conducted after the last iteration of the building phase in the *taxonomy testing* phase. There are various test levels. The basic prerequisite is the compliance of the taxonomy with XML, XML Schema, XLink specifications, and XBRL specification (Hoffman 2006, p. 378). These prerequisites are ensured by using a proper XBRL taxonomy editing tools. Already mentioned FRTA rules should be obeyed for the constructed taxonomies in order to follow the best practices of XBRL taxonomy development (Hamscher 2005). Another important step is the domain experts review of the final working draft (Hoffman 2006, p. 379). The domain experts ensure that the transformation of the information model into XBRL taxonomy code is conducted in a proper way as well as during the building phase. The reviewed taxonomy is usually released as an exposure draft to gain user feedback. The feedback is discussed during the building phase with the following revised working draft, tests, and exposure draft (IASCF 2006). Crucial for the test are samples and real instance documents. Instance documents are financial reports contain reported facts and are created according to the taxonomy. Sample instance documents are usually filled with artificial numbers when real instance documents represent real financial statements. Both should be constructed in order to validate the proper taxonomy design. Potential taxonomy users take often part during the testing phase. Their special task is often the preparing of real instance documents based on the taxonomy exposure draft.

The testing phase is followed by *taxonomy publication and recognition phase*. Usually XBRL taxonomies are published on the websites of the corresponding organisation as well as they are announced on the website of XBRL International (Calvert and Macdonald 2004, p. 6). There is a recognition process at the XBRL International level assigning to a taxonomy either the status acknowledged or approved. An acknowledged taxonomy is recognised as being in compliance with the XBRL Specification. An approved taxonomy is recognised as complying with the official XBRL guidelines for that taxonomy type as well as with the XBRL Specification. XBRL International can approve, ask for changes or reject the taxonomy during the recognition process (Calvert and Macdonald 2004, p. 4). The result of the publication and recognition phase is the final taxonomy version.

The *taxonomy usage and maintenance* starts directly after the publication. The taxonomy users are reporting entities and also software vendors building in the XBRL reporting capabilities into their software products. The experience derived from developed projects indicates that taxonomy maintenance requires taxonomy developers to monitor the taxonomy usage and track the issues and bugs the users experience with the taxonomy. The feedback from the taxonomy usage and maintenance phase has to be incorporated in the next version of the information model and has to be considered in the next taxonomy release. The problematic of taxonomy versioning is also a part of the maintenance phase. Approaches known from software versioning as well as ontology versioning systems offer an initial point for taxonomy versioning (XBRLLab 2006). But it has to be stated that there is no solution, yet, from either XBRL International or software vendors.

3.3 Critical review

Two different kinds of criteria are used to evaluate the presented approach. The first criteria type follows the IEEE standard (1998). XBRL taxonomies can be evaluated using the same criteria as in the field of software development, because they are at least, similar to ontologies, just a part of software products (Fernández-López et al. 1999). According to the IEEE-norm 1074-1997, criteria can be classified into three categories: taxonomy management activities, development oriented activities, and accompanying activities (IEEE 1998). The taxonomy management activities constitute the tasks and functions of the technical project management within the development process. This can be subdivided into planning, control, and quality assurance. The second category of IEEE criteria, development oriented activities, assume that these activities are divided into three groups. They are executed during the actual development, predevelopment, and post-development. The accompanying activities support the development process and are executed parallel to the specific phases of the development process. This development process can be subdivided into knowledge acquisition, evaluation, integration, configuration management, and documentation. It has to be considered that the executed activities are just identified. This is not precisely mentioned, when the activities have to be executed. The presented taxonomy development process model is described in details above. There are clearly taxonomy pre-development, development and post development activities indicated in the model. The model defined the accompanying activities for the building phase which are cyclic taxonomy working draft creations.

Criteria	***Software Engineering***	***Ontology Engineering***	***Taxonomy Engineering***
Generic model	Yes	No	Yes
Phase oriented models	Yes	No	Yes
Cycles allowed	Yes/ No	Yes	Yes
Knowledge acquisition phase	No	Yes	Yes
Extensive test phase	Yes	No	Yes
Reusing of existing software product/ ontology	Not addressed	Yes	Yes
Domain experts cooperation	Not addressed	Yes	Yes
Result	Software product	Ontology or knowledge-based system	XBRL taxonomy

Table 1. Comparison of software engineering and ontology engineering with taxonomy engineering

The second kind of criteria refers to different properties of the examined approach which can be helpful in developing a universal procedure model. Category parts are the criteria life cycle, application dependence, and the usage of core taxonomies. The life cycle describes the activities of each phase with its sequence and the interconnection to other phases. One criterion is to validate, if a certain sequence is suggested or not. There are two options. An incremental life cycle describes improvements which lead to a new version of the taxonomy. The result is that taxonomies can be developed step by step (McCracken et al. 1982). In contrast to this, evolutionary prototype modifications are allowed at any time without waiting for another version (Kendall 1995). The suggested approach identifies the taxonomy life cycle as well as suggests applying a waterfall model with feedback enabled combined with the prototyping approach as a process model. Nevertheless different process models can be adapted for the taxonomy life cycle without difficulties. The criteria application dependence examines, if the approach develops taxonomies which just can be adopted by single applications or if it is usable for different purposes. There are three different specifications: application dependent, application independent and semi-application dependent. They denote that the possible application scenarios are established during the specification. The discussed approach is application independent and is not developed in relation to any specific project or software.

Reusability is, according to an economic point of view, an important demand. It enables an efficient handling of available knowledge. The entire process of taxonomy development can be accelerated, so that it has not to be started completely from scratch. Taking this point into account, the criteria reusability is, of course, concerned in the evaluation.

Table 1 enhances figure 2 and presents a comparison between the two analysed approaches as well as taxonomy engineering described in the last section. The presented taxonomy development model builds upon different aspects of software engineering as well as ontology engineering. The model is generic and not software or project specific as well as indicates strong phases orientation. Cycles are allowed which is related to the idea of knowledge acquisition which has to be performed in order to build a taxonomy and which is a cyclic process. The taxonomy engineering approach indicates the great importance of the taxonomy testing phase where numerous best practices and patterns are used. Similarly to ontology engineering the idea of reusing of already existing taxonomies is realised in the model in form of developing a taxonomy extension instead of a core taxonomy. Not strongly addressed in the software engineering is the cooperation with domain experts.

4 CONCLUSIONS

The discussion during several recent XBRL International conferences raised a need for formalising the taxonomy development approach in a form of an abstract and generic process model. Although many such approaches exist in the area of software and ontology engineering they cannot be directly applied in the XBRL domain. Therefore based on the models known from two mentioned engineering approaches as well as on empirical experience of the authors, especially during the development of the IFRS, Polish GAAP and German GAAP taxonomies, an XBRL taxonomy development model is presented. The proposed XBRL taxonomy development process model fulfils the IEEE group of criteria as well as the universal procedure criteria analysed in the last chapter. The approach suggested in the fourth chapter should be helpful for organising the taxonomy development projects. The phases defined in the taxonomy life cycle as well as the proposed ordering and feedback allows a manageable and controllable taxonomy development and quantifiable XBRL projects. The model can be the base for defining the term taxonomy engineering. The definition can be derived and based upon the definition of software engineering suggested by IEEE Standard 610.12 (1998). Accordingly taxonomy engineering is the application of systematic, formal, quantifiable approach to the design, building, usage and maintenance of XBRL taxonomies, that is application of engineering to taxonomies. The paper represents an analysis of the subject and should be regarded as an introduction into the discussed matter. Although phases of taxonomy development are clearly defined various other possible ordering should be discussed as further research. Also the impact of agile software development should be analysed as further research in context of XBRL taxonomy development.

References

Balzert, H. (1998). Lehrbuch der Software-Technik. Software Mangement, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum Akademischer Verlag. Heidelberg.

Calvert, P., Macdonald, J. (2004). XBRL Taxonomy Recognition Process. http://www.xbrl.org/Taxono my Recognition /XBRL-Taxonomy-Recognition-Process-2004-11-19.doc. Last call 2007-03-14.

Corcho, Ó., Fernández-López, M. and Gómez-Pérez, A. (2003). Methodologies, Tools and Languages for Building Ontologies: Where is their meeting point? IEEE Transactions on Data and Knowledge Engineering, 46, 41-64. http://portal.acm.org/citation.cfm?id=864179.2003, Last call 2006-11-24.

DeMarco, T. (1978). Structured Analysis and System Specification. Yourdon Press. Englewood Cliffs.

Dumke, R. (2001). Software-Engineering: eine Einführung für Informatiker und Ingenieure: System, Erfahrungen, Methoden, Tools. 3rd Edition. Braunschweig.

Fairley, R. (1985). Software Engineering Concepts. McGraw-Hill.

Fernández-López, M., Gómez-Pérez, A., Pazos-Sierra, A. and Pazos-Sierra, J. (1999). Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent System, 14 (1), 37-46.
Fink, A., Schneidereit, G., Voß, S. (2001). Grundlagen der Wirtschaftsinformatik. Heidelberg.
Grüninger, M., Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies. Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada.
Hamscher, W. (2005). Financial Reporting Taxonomies Architecture 1.0. http://www.xbrl.org/technical/guidance/FRTA-RECOMMENDATION-2005-04-25+corrected-errata-2006-03-20.rtf. Last call 2007-03-14.
Hernández-Ros, I., Wallis, H. (2005). XBRL Dimensions 1.0. http://www.xbrl.org/Specification/XDT-REC-2006-09-18.rtf. Last call 2007-03-14.
Hernández-Ros, I. (2006). Taxonomy Life Cycle. http://www.xbrl.org/technical/requirements/TVER-REQ-PWD-2006-02-21.rtf. Last call 2007-03-14.
Hannon, N. (2004). Why Should Management Accountants Care about XBRL? Strategic Finance, July, 55-56.
Hoffmann, C. (2006). Financial Reporting Using XBRL: IFRS and US GAAP Edition, Lulu.
IASCF (2006). Status Descriptions. http://www.iasb.org/xbrl/taxonomies/taxonomy_status_description.html. Last call 2007-03-14.
IEEE Standard for Developing Software Life Cycle Processes (1998). IEEE Std 1074-1997. http://ieeexplore.ieee.org/xpl/standardstoc.jsp?isnumber=16018. Last call 2006-11-24.
The KACTUS Booklet version 1.0. (1995). Esprit Project 8145 KACTUS, http://www.swi.psy.uva.nl/projects/NewKACTUS/ Reports.html, Last call 2005-08-20.
Kendall, K. E. and Kendall, J. E. (1995). Systems Analysis and Design. 3rd Edition. New Jersey.
Ludewig, J. and Lichter, H. (2007). Software Engineering: Grundlagen, Menschen, Prozesse, Techniken. 1st Edition. dpunkt, Heidelberg.
McCracken, D. D. and Jackson, M. A. (1982). Life Cycle Concept Considered Harmful. ACM Software Engineering Notes 7 (2), 29-32.
Nutz, A. and Strauß, M. (2002). eXtensible Business Reporting Language (XBRL) - Konzept und praktischer Einsatz, Wirtschaftsinformatik, 44, 447-457.
Pomberger, G., Blaschek, G. (1996). Software Engineering: Prototyping und objektorientierte Software-Entwicklung. München.
Romney, B. M. and Steinbart, P. J. (2006). Accounting Information Systems, 10th Edition, Pearson Prentice Hall.
Schreiber, G., Akkermans, H., Anjewierden, A. A., de Hoog, R., Shadbolt, N. R., van de Velde, W. and Wielinga, B. J. (2000). Knowledge Engineering and Management: The CommonKADS Methodology. Cambridge / London.
Sommerville, I. (2001). Software Engineering. 6th Edition. Pearson Studium. München.
Staab, S., Schnurr H. P., Studer, R. and Sure, Y. (2001). Knowledge Processes and Ontologies. IEEE Intelligent Systems, 16 (1), 26-34.
Studer, R., Benjamins, R. V. and Fensel, D. (1998). Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering 25 (1-2), 161-197.
Sure, Y. (2002). On-To-Knowledge: Ontology-based Knowledge Management Tools and their Application. KI 14 (1), 35-37.
Sutton, S. G. (2006). Enterprise systems and the re-shaping of accounting systems: A call for research, in International Journal of Accounting Information Systems, 7, 1-6.
Swartout, B., Patil, R., Knight, K. and Russ, T. (1996-9). Towards Distributed Use of Large-Scale Ontologies. Proceedings of the 10th KAW'96, Banff, Canada and AAAI'97 Spring Symposium on Ontological Engineering, 138-148.
XBRLLab. (2006). Taxonomy Versioning. http://www.xbrl-ifrs.org/versioning/background.html. Last call 2007-03-14.
Zuser, W., Biffl, S., Grechenig, T., Köhle, M. (2001). Software Engineering. München.