Enterprise Knowledge

Reference Architecture for Visualization

and Prototype Implementation using Wiki

Thesis for the degree of Master of Science of the University of Twente

S.U. Wicaksono

Enschede, August 2012

UNIVERSITEIT TWENTE.

BiZZdesign

Enterprise Knowledge

Reference Architecture for Visualization and Prototype Implementation using Wiki

Author	Sulistyo Unggul Wicaksono
Study program	Master of Science in Business Information Technology
Student number	S1027085
e-mail	unggul.suw@gmail.com

Graduation committee

First supervisor	Dr. M.E. Iacob School of Management and Governance Information Systems and Change Management Group University of Twente, Enschede, The Netherlands
Second supervisor	Dr. N. Sikkel Faculty of Electrical Engineering, Mathematics, and Computer Science Information Systems Group University of Twente, Enschede, The Netherlands
External supervisors	Dr. Ir. H. Bakker CTO BiZZdesign, The Netherlands W. Engelsman, Msc. BiZZdesign, The Netherlands

UNIVERSITEIT TWENTE.

BiZZdesign

ABSTRACT

In this information-rich era, knowledge plays an important role for organizations. The challenge is not only of having the right knowledge, but also of making relevant knowledge actionable. In organizations having knowledge generated from problem-solving interactions, management and accommodation of that knowledge play an important role. Repeated process of regulating and advancing IT support that is available or prepared to be made available for an organization is called management of enterprise architecture. It is a process that involves IT, business process, strategies, and business goals.

To accommodate those generated knowledge, organizations will need to focus on how to sort and filter knowledge. The kind of knowledge that should be sorted and codified are knowledge related to problem solving interactions. We refer to that kind of knowledge as enterprise knowledge.

This research aims at studying how to present enterprise knowledge using a visualization tool. From literature research, we find that enterprise knowledge can be embedded in diagrams as a form of visualization formats. These diagrams will be referred as models in the remainder of this research.

A study of presenting enterprise knowledge is important because it is highly practical. This is due to knowledge visualization and visual representations exist in many fields across different domains of business. There are many ways to present enterprise knowledge contained in models. To do so, choices can be made from numerous visualization tools.

Growth of wiki software since the advent of Wikipedia has made wiki software a relevant choice as tool to manage and visualize knowledge, including enterprise knowledge. Some of the softwares have high extensibility, which enables extra functions added by developers, including that related to knowledge extraction and presentation. This thesis provides a reference architecture and its implementation in form of a prototype built with a wiki software. Both are provided to facilitate presentation of enterprise knowledge. The reference architecture is designed to facilitate enterprise knowledge contained in any models. Utilization of knowledge repository and various possible visualization tools is prominent in the reference architecture.

For prototype implementation, the enterprise knowledge embedding format is restricted to business process models and architecture models. The prototype is then validated by means of a survey. This survey's result shows that although our prototype still has limitations, it fulfills the requirements to present enterprise knowledge embedded in models.

Keywords: enterprise knowledge, models, knowledge visualization, knowledge repository, reference architecture, wiki.

BiZZdesign

Table of Contents

ABSTRACT 3

1. INTRODUCTION 9

1.1 CONTEXT 9
1.2 RESEARCH GOAL 10
1.3 RESEARCH QUESTIONS 10
1.4 RESEARCH APPROACH AND REPORT STRUCTURE 10

2. STATE OF THE ART 13

2.1 KNOWLEDGE AND ITS CONVERSION 13
2.2 ENTERPRISE KNOWLEDGE 14
2.3 KNOWLEDGE VISUALIZATION 15
2.4 KNOWLEDGE VISUALIZATION USING WIKI 16
2.4.1 USING WIKI TO MANAGE KNOWLEDGE 16
2.4.2 WIKI TECHNOLOGIES FOR KNOWLEDGE VISUALIZATION 17
2.4.3 OVERVIEW OF WIKI SOFTWARE 19
2.5 KNOWLEDGE REPOSITORY 19
2.6 ENTERPRISE KNOWLEDGE EMBEDDING 21
2.7 CURRENT SITUATION AT BIZZDESIGN 21
2.7.1 BIZZDESIGN 21
2.7.2 BIZZDESIGN REPOSITORY 22
2.8 SUMMARY 24

3. REFERENCE ARCHITECTURE FOR ENTERPRISE KNOWLEDGE VISUALIZATION 25

3.1 PRELIMINARIES 25
3.1.1 MANAGING ENTERPRISE KNOWLEDGE TO BE VISUALIZED 25
3.1.2 VISUALIZING MODELS CONTAINING ENTERPRISE KNOWLEDGE 26
3.2 ENTERPRISE KNOWLEDGE VISUALIZATION REFERENCE ARCHITECTURE 26
3.2.1 REQUIREMENTS FOR REPOSITORY 29
3.2.2 BUSINESS LAYER OF REFERENCE ARCHITECTURE 36
3.2.3 APPLICATION LAYER OF REFERENCE ARCHITECTURE 38
3.2.4 TECHNOLOGY LAYER OF REFERENCE ARCHITECTURE 39
3.3 DISCUSSION AND SUMMARY 39

4. ENTERPRISE KNOWLEDGE VISUALIZATION AT BIZZDESIGN 41
4.1 WIKI AND MEDIAWIKI 41
4.2 BUSINESS PROCESS MODELS AND ARCHITECTURE MODELS 42
4.3 IMPLEMENTATION OF ENTERPRISE KNOWLEDGE VISUALIZATION USING WIKI SOFTWARE 44
4.3.1 REQUIREMENTS ANALYSIS AND DEFINITION 44
4.3.2 SYSTEM AND SOFTWARE DESIGN 45
4.3.3 IMPLEMENTATION AND UNIT TESTING 49
4.3.4 INTEGRATION AND SYSTEM TESTING 53
4.3.5 OPERATION AND MAINTENANCE 57
4.4 IMPACT TO BIZZDESIGN'S SITUATION 57
4.4.1 FEATURES OF THE PROTOTYPE 58
4.4.2 ADDITIONAL INFORMATION 59
4.4.3 ADDITIONAL FUNCTIONS 60
4.5 SUMMARY 61

5. VALIDATION 62
5.1 VALIDATION APPROACH 62
5.2 VALIDATION RESULT 65
5.2.1 FUNCTIONALITY 65
5.2.2 UTILITY/USEFULNESS 67
5.2.3 QUALITY 69
5.3 SUMMARY 70

6. FINAL REMARKS 72
6.1 CONCLUSION 72
6.2 LIMITATIONS 74
6.3 FUTURE WORK 75

ACKNOWLEDGEMENTS 76

REFERENCES 77

APPENDIX A. PROTOTYPE SOURCE CODE 82

APPENDIX B. SURVEY QUESTIONS 101

Table of Figures

Figure 1-1 Methodology of this research 11

Figure 2-1 Potential explicability of knowledge (Zack, 1999) 15

Figure 2-2 Main page of ArchiXL's wiki (ArchiXL, 2012) 16

Figure 2-3 "Fiscaal" as an application service in ArchiXL's wiki (ArchiXL, 2012) 17

Figure 2-4 Categorization in Wikipedia 18

Figure 2-5 BiZZdesign tool architecture (BiZZdesign, 2011) 22

Figure 2-6 Sample repository structure (BiZZdesign, 2011) 23

Figure 2-7 Metamodel hierarchy 24

Figure 3-1 Enterprise knowledge visualization reference architecture 37

Figure 4-1 Example of a business process model (Dijkman & Joosten, 2002) 42

Figure 4-2 Example of an architecture model (Lankhorst at al., 2009) 43

Figure 4-3 Waterfall software development life cycle (Huo et al., 2004) 44

Figure 4-4 Architecture of the prototype 46

Figure 4-5 Data model of BiZZdesign repository 48

Figure 4-6 Interface design 49

Figure 4-7 Testing of initial page of enterprise knowledge visualization using repository (Initializer component) 54

Figure 4-8 Testing visualization of 'profit' model package 55

Figure 4-9 Testing visualization of 'claim settlement', inside 'profit' model package 55

Figure 4-10 Testing visualization of 'Archisurance' model package 56

Figure 4-11 Testing of search function of the prototype with search query that includes wildcard character 56

Figure 4-12 Testing of search function of the prototype with search query that includes both wildcard and operator 57

Figure 4-13 An XML page generated by repository system 60

Figure 5-1 Map of EK visualization prototype's validation question 63

Figure 5-2 Score of validation aspects 65

Figure 5-3 Answers to Q1 66
Figure 5-4 Answers to Q2 66
Figure 5-5 Answers to Q3 66
Figure 5-6 Answers to Q4 66
Figure 5-7 Answers to Q5 67
Figure 5-8 Answers to Q6 67
Figure 5-9 Answers to Q7 67
Figure 5-10 Answers to Q8 68
Figure 5-11 Answers to Q9 68
Figure 5-12 Answers to Q10 68
Figure 5-13 Answers to Q11 68
Figure 5-14 Answers to Q12 69
Figure 5-15 Answers to Q13 69
Figure 5-16 Answers to Q14 69
Figure 5-17 Answers to Q15 70
Figure 5-18 Answers to survey questions 70

List of Tables

Table 1-1 Research questions and corresponding methodology 11
Table 3-1 Mapping of quality goals of reference architecture and requirements of repository 35
Table 3-2 Requirements for repository 36
Table 4-1 Mapping of prototype architecture to reference architecture 48
Table 4-2 Features of the prototype 59
Table 5-1 Design-science research guidelines (Hevner et al., 2004) 62
Table 5-2 Survey questions 64

Abbreviations

BPS	Business Process Support
DBMS	Database Management System
EA	Enterprise Architecture
GUI	Graphical User Interface
IS	Information System
IT	Information Technology
KM	Knowledge Management
KMS	Knowledge Management System
PHP	PHP Hypertext Preprocessor
SQL	Structured Query Language
WAMP	Windows Apache MySQL PHP
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

1. Introduction

1.1 Context

Growth of information technology including how it enables organizational growth has resulted in business and technology initiatives within organizations. Challenges and dynamics in organizational activities require those initiatives to be aligned with a common organizing logic.

In aligning those initiatives within a process of change involves integration, a holistic approach is needed (Iacob et al., 2007a). Enterprise architecture management can be defined as a repeated and perpetual process of regulating and advancing IT support that is available for an organization. It is a process that involves IT, business process, strategies, and business goals (Ernst et al., 2006).

A model can include knowledge (Purvis et al., 2001) and can also be represented in a graphical representation (Goul & Corral, 2007). To communicate graphical representations, a relevant presentation enabler shall be of high importance in consideration.

As a provider of enterprise architecture management tools, BiZZdesign-made tools have been widely used by information system architects and business analysts. Results from the tools easily connect to users and stakeholders with related background. Nonetheless, there is further need to have knowledge spread and made understandable to different stakeholders, including organization personnel without education background in the area. In regard to that, knowledge contained inside architecture models and business processes models has to be represented.

Thus, a possible solution of visualization is sought to represent knowledge contained in the repository. Knowledge that is going to be represented is gained from usage of tools related to business process and architecture. This particular repository has been a mean for business process-related design tools to retrieve and store knowledge.

Moreover, we argue that a level of understanding can be drawn from how a system presents enterprise knowledge that is embedded in models. We believe that an instance of visualization system with pertinent software can facilitate it. Therefore, we argue that software like wiki that is capable of presenting knowledge will be needed to handle visualization of models.

1.2 Research goal

With context from previous section, the need for a conceivable knowledge presentation is addressed. We can thus state the goal of this research:

"To study how to design a system to present embedded enterprise knowledge."

1.3 Research questions

With the research goal in mind, the followings research questions are formed:

1. What is the knowledge stored in a repository of an organization?
   a. What type of knowledge is stored?
   b. What are the formats of the knowledge stored?
2. How is wiki supporting presentation of enterprise knowledge?
   a. What are the advantages of a wiki?
   b. How can wiki support presentation of enterprise knowledge?
3. What is the design of a system that transfers enterprise knowledge from a repository to a wiki?
   a. How to design reference architecture for visualizing enterprise knowledge?
   b. How to implement a prototype based on the reference architecture?

1.4 Research approach and report structure

The research will be conducted with steps shown in Figure 1-1. Each research question in previous section will be solved using relevant methodology. The corresponding methodologies can be seen in Table 1-1.

Research question	Methodology
• What is the knowledge stored in a repository of an organization? ▪ What type of knowledge is stored?	BiZZdesign documentation and literature research.
• How is wiki supporting presentation of enterprise knowledge? ▪ What are the advantages of a wiki? ▪ How can wiki support presentation of enterprise knowledge?	Literature and case studies research.

• What is the design of a system that transfers enterprise knowledge from a repository to a wiki? ▪ How to design reference architecture for visualizing enterprise knowledge? ▪ How to implement a prototype based on the reference architecture?	Implementation of prototype and literature research, survey.

Table 1-1 Research questions and corresponding methodology

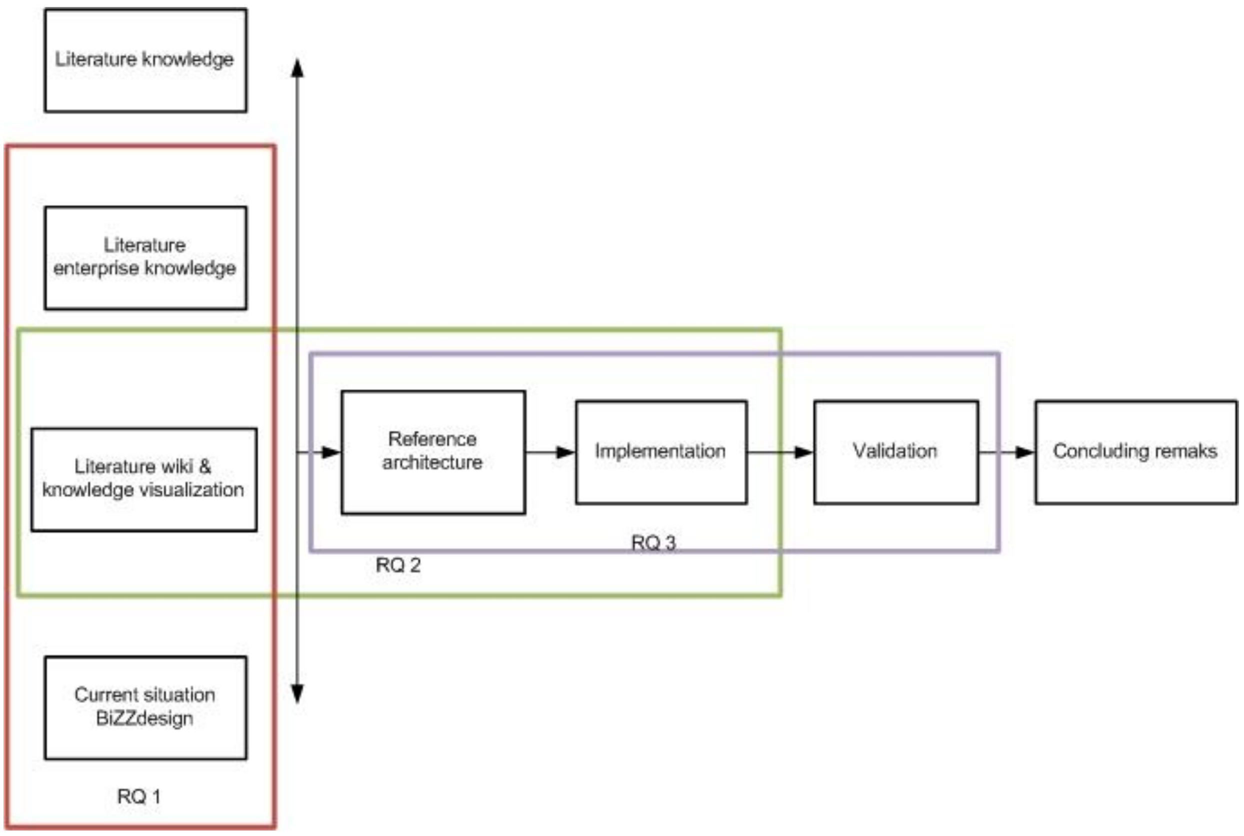

Literature knowledge
Literature enterprise knowledge
Literature wiki & knowledge visualization
Current situation BiZZdesign
RQ 1
Reference architecture
Implementation
Validation
Concluding remaks
RQ 2
RQ 3

Figure 1-1 Methodology of this research

Research questions 1 will be answered with literature studies and current situation assessment. Research question 2 and 3 will be responded with combination of steps, which is depicted in our methodology in Figure 1-1.

This report will be structured with each box/boxes in Figure 1-1 that has the same vertical position be organized in a chapter. Literature of knowledge, enterprise knowledge, knowledge visualization, and current situation at BiZZdesign will constitute chapter 2. Next chapter will specify the aspects for reference architecture, including a reference architecture for knowledge visualization itself. The case of BiZZdesign, in which this research is also aiming to solve, will be addressed in the implementation part, which will be covered in chapter 4. In the next chapter, validation will be carried out and chapter 6 will cover relevant concluding remarks.

2. State of The Art

Existing theories and research that serves as bases and reference for this research covers these domains: knowledge-enterprise knowledge, knowledge visualization, knowledge repositories, and embedding of enterprise knowledge. Current situation of the company that this research is conducted at is added in next section. We describe each domain starting from knowledge in this chapter.

2.1 Knowledge and its conversion

Some knowledge can be expressed with written formats or spoken words. This kind of knowledge is called explicit knowledge. On the other side, knowledge that is rooted in people's mind and cannot be articulated easily is called tacit knowledge (Hasan & Pfaff, 2006).

On knowledge importance, Nonaka (1991) comes with the argument below:

One sure source of lasting competitive advantage is knowledge.

Knowledge can be seen by how it is attached to individual or by its background (i.e. situation) (Stenmark, 2001). Knowledge ranges from specific to general. Specific knowledge is subject-oriented or topic-oriented. Meanwhile, general knowledge is extensive and independent to a certain phenomenon. It is also commonly available in public and tends to be able to be structured easier. Consequently it has more significance in communities of practice or communities of knowledge (Zack, 1999).

Nonaka (1994) proposes four types of knowledge conversion: tacit to tacit (socialization), tacit to explicit (externalization), explicit to explicit (combination), and explicit to tacit (internalization). Socialization deals with relocation of tacit knowledge in an interpersonal direction. Direct interactions with stakeholders inside or outside the organization are important in socialization. Experience and attained skill play an important role in this process.

The second type is externalization. Tacit knowledge is made explicit by making it understood by writing out or putting it to another medium, which can happen with different means of format involving a number of individuals. Explicit form of knowledge can thus be relocated with a process called combination. Support of IT is prominent in this process, because it serves as enabler for format of this explicit-to-explicit knowledge conversion.

Communication between groups in an organization is an example of this. Internalization, the last conversion type, is a process of understanding and gripping explicit knowledge into tacit knowledge possessed by a person. Internalization is largely experiential.

In this research, we consider organization and embedding to be prominent and we relate it with externalization and combination from Nonaka (1994). The relation motivates us to study how knowledge and its representation can use the help of IT in its conveyance.

2.2 Enterprise knowledge

Organizations have different ways to discover knowledge. This can include different tools and approaches to examine data in text and numeric format. Knowledge generation from numeric databases is an example of how organizations discover knowledge. Application of knowledge discovery has been used in fields such as security, customer analysis, fraud analysis, and product analysis (O'Leary, 1998).

To gain effectiveness in growth and performance, organizations with knowledge-intensive activities need to share and integrate highly distributed knowledge (Zack, 1999). In distributing, tacit knowledge needs to be turned into explicit to be shared, exchanged, and combined (Nonaka, 1991). To be able to have a competitive performance, organizations need to seek a balance between portions of knowledge made explicit and knowledge left tacit. Outside initiating community, parts of tacit knowledge need to be made explicit. Determining which knowledge to be made explicit relevantly is a challenge towards that balance (Zack, 1999).

Zack (1999) argues that the process of turning private knowledge to become accessible can cause refusal to accept in certain organizational cultures. Lack of formal model or language to articulate can also cause tacit knowledge to stay unarticulated, which is an example of intellectual restriction. There are four conditions in regard to relation of tacit knowledge and explicit knowledge. In Figure 2-1, explicability of knowledge and connection to opportunity is shown. A lost opportunity means a failure to explicate potentially explicable knowledge. An organization needs to pay attention on ability to articulate certain knowledge to decide whether to articulate certain knowledge or not. This will in turn lead to competitive advantage as a result of a right balance between tacit and explicit knowledge (Zack, 1999).

From the perspective of IS, knowledge is on top of data-information-knowledge hierarchy. In this hierarchy, Hasan & Pfaff (2006) consider information to be significant, prepared data. Meanwhile, knowledge is information that is actionable.

Enterprise knowledge is knowledge within an enterprise. Firestone (2000) defines it as a continuous, dynamic circulation of knowledge-related problem-solving interactions. These interactions are source of knowledge to be generated. This generated knowledge will in turn be accommodated with business processes of the enterprise.

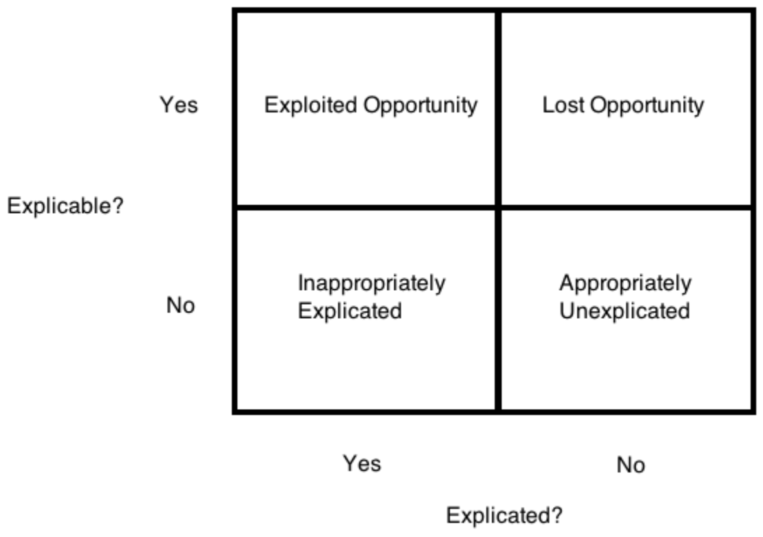


Figure 2-1 Potential explicability of knowledge (Zack, 1999)

Adopting the definition from Firestone (2000), enterprise knowledge management can be defined as management function responsible for regular implementation, selection and evaluation of knowledge strategies that are directed towards creating an environment to support knowledge work within continuous, dynamic circulation of knowledge-related problem-solving interactions.

2.3 Knowledge visualization

Visualization can be defined as cognitive process using human vision system, which has analytical functions and potent information processing capabilities. It has always been a dominant factor in scientific progress. With aid of computers, visualization has become even more capable (Hoogeboom, 2005).

According to Kurfess (2008), visualization is one way of presenting knowledge and it is possibly the most important. Still in the same context, Kurfess also argues that explicit knowledge can be presented almost directly, while tacit knowledge must be regulated. Moving on, Eppler and Burkhard (2004) suggest knowledge visualization as being concerned with:

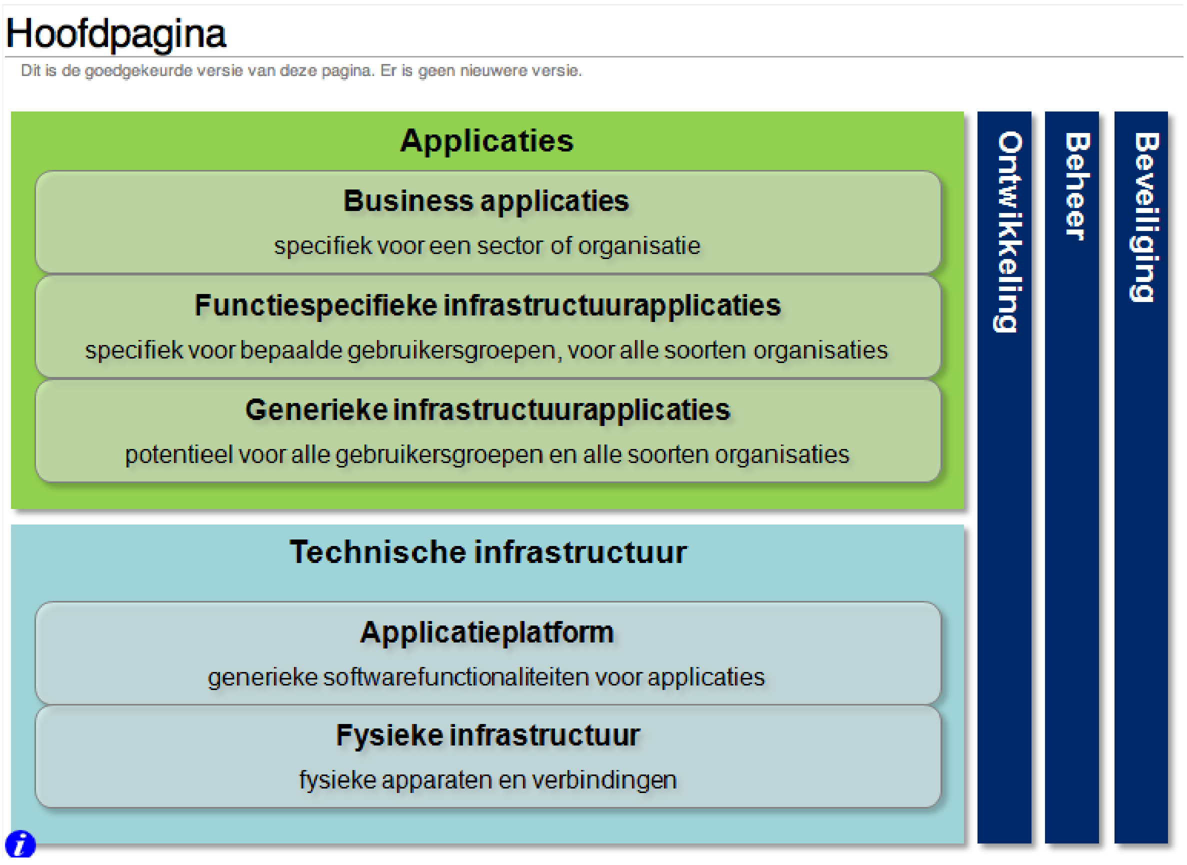
The use of visual representations to improve the creation and transfer of knowledge between at least two people.

Keller and Tergan (2005) argue that knowledge visualization began to emerge in social sciences (learning and instructional science) and that information visualization is more related to computer science. Although information visualization and knowledge visualization are closely related, we argue that knowledge visualization is the relevant subject to be studied in this research. We also consider presentation of knowledge to be closely related to knowledge visualization.

2.4 Knowledge visualization using Wiki

2.4.1 Using Wiki to manage knowledge

Gonzalez-Reinhart (2005) divides early wiki use for knowledge management (KM) into three generations. The first generation provides a KM that can be characterized as conventional. This generation starts from Wiki-Wiki in 1995 and until the use of wiki before 2004. The second generation is more intended for users in corporate level that has not switched to wiki, mostly in 2004.


Hoofdpagina

Dit is de goedgekeurde versie van deze pagina. Er is geen nieuwere versie.

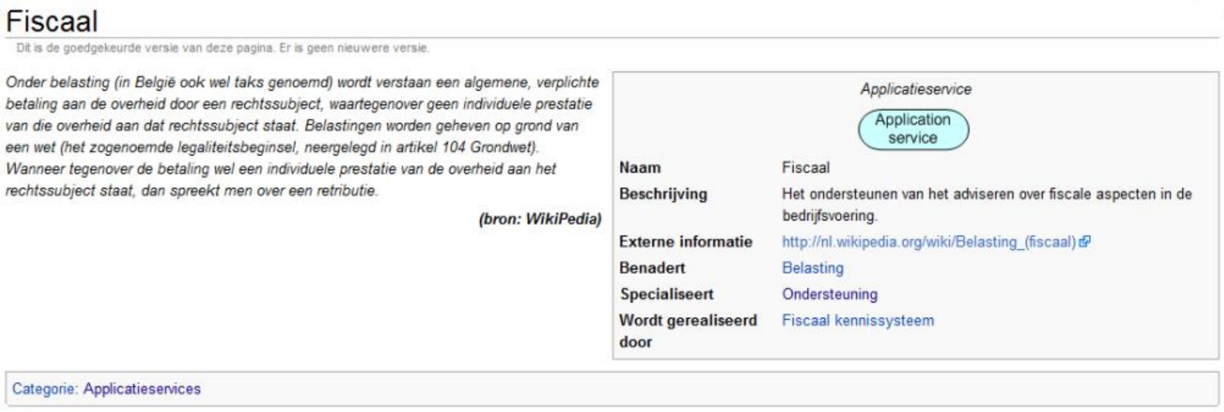
Applicaties
Business applicaties
specifiek voor een sector of organisatie
Functiespecifieke infrastructuurapplicaties
specifiek voor bepaalde gebruikersgroepen, voor alle soorten organisaties
Generieke infrastructuurapplicaties
potentieel voor alle gebruikersgroepen en alle soorten organisaties
Technische infrastructuur
Applicatieplatform
generieke softwarefunctionaliteiten voor applicaties
Fysieke infrastructuur
fysieke apparaten en verbindingen
Ontwikkeling
Beheer
Beveiliging

Figure 2-2 Main page of ArchiXL's wiki (ArchiXL, 2012)

Meanwhile, the third generation emerged around October 2005. A release called Application Wiki was showing functionality exceeding attributes found in previous two generation of wikis. In this third generation, an addition to basic wikis is WYSIWYG editing, allowing users or contributors to edit pages (Gonzalez-Reinhart, 2005).

Figure 2-2 and 2-3 shows the use of wiki for knowledge management by ArchiXL. The wiki's content is composed of parts of IT reference architecture defined by the company. Figure 2-2 shows how a wiki software can handle links and facilitate them using image representation.

Relation between concepts in ArchiXL's wiki has been handled so that users can browse through related objects. Relations and related objects are clearly shown in the information box (see Figure 2-3).



Fiscaal

Dit is de goedgekeurde versie van deze pagina. Er is geen nieuwere versie.

Onder belasting (in België ook wel taks genoemd) wordt verstaan een algemene, verplichte betaling aan de overheid door een rechtssubject, waartegenover geen individuele prestatie van die overheid aan dat rechtssubject staat. Belastingen worden geheven op grond van een wet (het zogenoemde legaliteitsbeginsel, neergelegd in artikel 104 Grondwet). Wanneer tegenover de betaling wel een individuele prestatie van de overheid aan het rechtssubject staat, dan spreekt men over een retributie.

(bron: WikiPedia)

Applicatieservice

Naam	Fiscaal
Beschrijving	Het ondersteunen van het adviseren over fiscale aspecten in de bedrijfsvoering.
Externe informatie	http://nl.wikipedia.org/wiki/Belasting_(fiscaal)
Benadert	Belasting
Specialiseert	Ondersteuning
Wordt gerealiseerd door	Fiscaal kennissysteem

Categorie: Applicatieservices

Figure 2-3 "Fiscaal" as an application service in ArchiXL's wiki (ArchiXL, 2012)

Moreover, wiki is also a common tool for knowledge management. A research done using two different wiki softwares had shown that wiki facilitates knowledge creation and knowledge sharing, two important aspects of knowledge management (Liang et al., 2009).

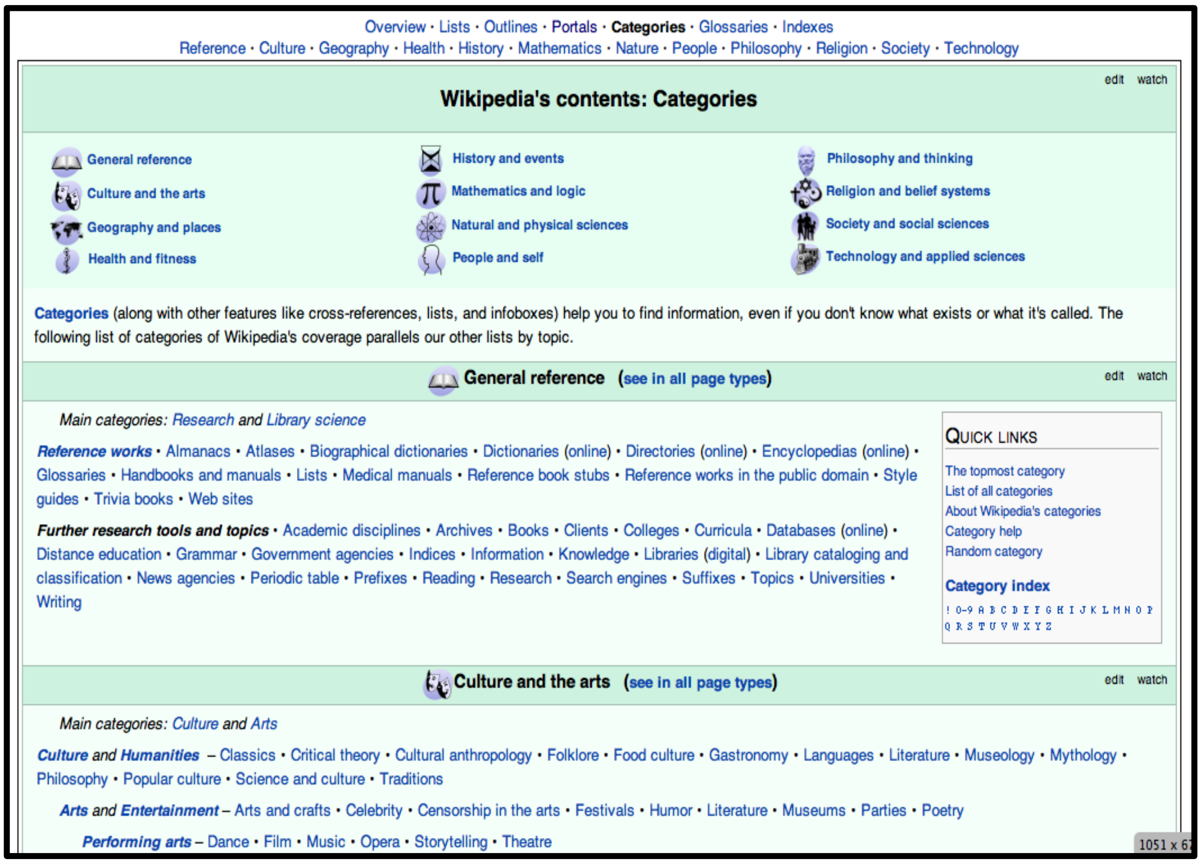
2.4.2 Wiki technologies for knowledge visualization

A page under a certain title in Wikipedia can be written on any subject. McMahon (2008) adds that while there are regulations in order to prevent offensive or profit-oriented contents, wide range of knowledge that can be introduced represents the width inside Wikipedia. Di Iorio and Zacchiroli (2006) examine that authors pay attention to domains in a wiki with previously-assigned structure, even if they do have the liberty to create new contents. Contents placed in wikis indirectly go through a process of checking. This is due to that wiki make it possible for people to collect and present information of importance or interest. Additionally, organizational peers are actually editors that can modify other person's post. The role of editors will enrich a posting by adding additional information, links, tables, and photos (McMahon, 2008).

In a wiki, taxonomy can have two meanings to users. On one side, it eases navigation. On the other side, users can feel that the wiki is unable to be modified in terms of content structure. Thus, it is a wiki manager's job to make sure the navigation is easy enough while sending right signal of flexibility (McMahon, 2008).

In a study carried out on wiki workspaces, an intranet for a certain computer science department is examined. The organization of content inside is based on type of content, covering projects, courses, department's issues, and students' clubs (Buffa, 2006). Its structure and taxonomy facilitate 4000 pages that are classified into 25 workspaces, showing that taxonomy can be an enabler of knowledge visualization.

In relation to taxonomy, one of other important features of a wiki for knowledge visualization is categorization. In Figure 2-4, an example of wiki technology used for categorizing contents of Wikipedia at the end of the year 2011 is shown. In Wikipedia, contents are divided into twelve: general reference, culture and the arts, geography and places, health and fitness, history and events, mathematics and logic, natural and physical sciences, people and self, philosophy and thinking, religion and belief systems, society and social sciences, and technology and applied sciences.


Overview · Lists · Outlines · Portals · **Categories** · Glossaries · Indexes
Reference · Culture · Geography · Health · History · Mathematics · Nature · People · Philosophy · Religion · Society · Technology

edit watch

Wikipedia's contents: Categories

General reference
Culture and the arts
Geography and places
Health and fitness
History and events
Mathematics and logic
Natural and physical sciences
People and self
Philosophy and thinking
Religion and belief systems
Society and social sciences
Technology and applied sciences

Categories (along with other features like cross-references, lists, and infoboxes) help you to find information, even if you don't know what exists or what it's called. The following list of categories of Wikipedia's coverage parallels our other lists by topic.

General reference (see in all page types) edit watch

Main categories: Research and Library science

Reference works • Almanacs • Atlases • Biographical dictionaries • Dictionaries (online) • Directories (online) • Encyclopedias (online) • Glossaries • Handbooks and manuals • Lists • Medical manuals • Reference book stubs • Reference works in the public domain • Style guides • Trivia books • Web sites

Further research tools and topics • Academic disciplines • Archives • Books • Clients • Colleges • Curricula • Databases (online) • Distance education • Grammar • Government agencies • Indices • Information • Knowledge • Libraries (digital) • Library cataloging and classification • News agencies • Periodic table • Prefixes • Reading • Research • Search engines • Suffixes • Topics • Universities • Writing

QUICK LINKS
The topmost category
List of all categories
About Wikipedia's categories
Category help
Random category

Category index
! 0–9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Culture and the arts (see in all page types) edit watch

Main categories: Culture and Arts

Culture *and* ***Humanities*** – Classics • Critical theory • Cultural anthropology • Folklore • Food culture • Gastronomy • Languages • Literature • Museology • Mythology • Philosophy • Popular culture • Science and culture • Traditions

Arts *and* ***Entertainment*** – Arts and crafts • Celebrity • Censorship in the arts • Festivals • Humor • Literature • Museums • Parties • Poetry

Performing arts – Dance • Film • Music • Opera • Storytelling • Theatre

1051 x 67

Figure 2-4 Categorization in Wikipedia

Moreover, wiki has technologies to enable searching, self-publishing, and organizing and sharing of personal information (Andersen, 2004). These tools included in wikis can underpin knowledge visualization and provide options for variability in the process. A combination of blog and wiki (bliki) mentioned by Andersen was an example of this in a personal form.

2.4.3 Overview of Wiki software

These following wiki software products are considered the most common example by Louridas (2006) and Ebersbach et al. (2006).

Confluence is a wiki with an ability to provide different spaces in structuring its content. This ability means that it has multi-linked wikis as part of its features, alongside space and page-level security constraints and multiple content searching. Confluence, released in 2004, is one example of proprietary wiki software.

MediaWiki was born as a necessity in the project of Wikipedia in 2002. It facilitates numerous extensibilities, also flexibility enabled by the use of PHP Hypertext Preprocessor (PHP) script.

MoinMoin is another wiki software. Written in Python, this software enables user registration and does not require a database connection. It is supported with a plug-in system for feature augmentation and other functions.

TWiki, which has many plug-ins and features, is the most all-inclusive one. TWiki, created to be a company intranet, is utilized progressively for commercial purposes, due to its significant stature of development. Created in July 1998, it allows easy reporting and access control.

UseModWiki, released initially in 1999, is one of the oldest and the most widely used wiki software. It has had a prominent effect on other wikis' development. One example of a product of that effect is MediaWiki, whose syntax firmly takes after that of UseModWiki.

2.5 Knowledge Repository

According to Zack (1999), for explicit knowledge to be managed there are four primary resources needed: repositories, refineries, organization roles, and information technology. Information technology infrastructure should be seamless for providing a flow of explicit knowledge through stages of refining process to enable knowledge capture, handling digital objects that points to knowledge unit, presenting content to make it usable in different cases, and reaching relevant needed contents (Zack, 1999).

Repository is defined by Bernstein & Dayal (1994) as a shared, structured set containing information on an enterprise that is mainly about managed artifacts that the enterprise produces or uses. These artifacts include documents, maps, information systems, software and components. Knowledge as an object is composed by two components: structure and content. Zack sees the design of a knowledge repository to be based on those components. Knowledge structure gives the background for contents accumulated inside. This background would be needed to interpret those contents in many different views.

As a structure gets more complicated, a scheme for connecting and cross-referencing units of knowledge becomes a necessity. These knowledge units have different attributes whose values vary depending on the context and kind of explicit knowledge that is stored. Those units act as base element of the structure; a bundle of content open to manipulation, storage, retrieval, labeling, and indexing. In doing all these, the mentioned schemes are useful because they symbolize conceptual associations, ordered sequences, and/or cause-effect relationship (Zack, 1999).

Explicit organizational knowledge has a span that makes recording relevant knowledge content a challenge to strive for repositories. In order to capture noteworthy and meaningful concepts, the design of a repository should enable sufficient knowledge recording capability. It is a knowledge engineer job to itemize the repository in accordance to those concepts. Moreover, several repositories can be joined together inside a knowledge platform, regardless of the structures of each repository. The structure can be different because type of knowledge or content in each repository can vary. A logical linkage can be made across those repositories with content from each serving as context for explicating other repositories' content (Zack, 1999).

Dingsoyr and Royrvik (2003) point out that:

In creating a knowledge repository, knowledge is collected, summarized, and integrated across sources.

The repository can either be filled with knowledge by letting workers themselves identify what knowledge has adequate value to be stored in the repository. It can also be done by assigning some people in an organization to scan communication processes to uncover knowledge (van Heijst et al., 1997).

A study has developed a knowledge management system, which includes a repository divided in two parts (Dingsoyr and Royrvik, 2003). The first one stores structured internal knowledge that includes databases for sales and marketing information and employee capability. It also includes examples of documents, templates, software components, procedure information, and research reports. The second one stores informal internal knowledge that includes electronic discussion forums, news and announcement, general review of projects, and technical documents.

Overall, in any system involving knowledge, structure of knowledge repositories determines their meaningfulness. It needs to reflect a structure of shared mental model of contextual knowledge agreed in principle in the organization. However, the structure's definition or sharing is still a common limitation in most organizations. The knowledge unit then must be supported by a sufficient definition, including how it should be grouped for ease of exchange, retrieval, access and integration (Zack, 1999).

2.6 Enterprise knowledge embedding

Repositories can store knowledge using a predefined structure. This structure, however, needs to be customized to set of knowledge that needs to be stored, as section 2.5 described. For enterprise knowledge, possible medium can vary depending on how organizations approach their problems in attempt to solve those problems (Das, 2003). Argote et al. (2003) also underlines the importance of knowledge embedding approach.

Vernadat (2001) mentioned that projecting enterprise knowledge can bring value to the enterprise. Medium of embedding for enterprise knowledge can vary from plant layout model (Berio & Vernadat, 1999), workflow (Goul & Corral, 2007; Kuhn, et al., 2003), business process model, reference model (Kuhn, et al., 2003), database design, data flow diagram, and action diagram (Purvis, 2001). Enterprise modeling in particular deals with depiction and examining automation in the enterprise, including building model that represents activities within an enterprise (Berio & Vernadat, 1999).

Enterprise models are used to help understand specific condition with high intricacy that made it difficult to understand (Ba et al., 1995). They also can assist in performance analysis, explain design environment (Goul & Corral, 2007), and decision support (Ba et al., 1995). Various modeling languages and tools to develop enterprise models exist (Vernadat, 2001).

2.7 Current situation at BiZZdesign

2.7.1 BiZZdesign

BiZZdesign is a company focusing its service lines on enterprise architecture management, business requirements management, business process design and improvement, business process management, and structured implementation and governance. The company has main offices in Netherlands and other offices in Germany (which services Germany, Switzerland, and Austria), in Belgium, in United Kingdom, and in North America. These software tools constitute BiZZdesign’s EA Management suite: Architect (to model, visualize, and analyze enterprise architecture), BiZZdesigner (business process management), RiskManager (risks and governance), and GripManager (implementation projects). The company also provides training and consultancy along its service lines (BiZZdesign, 2012).

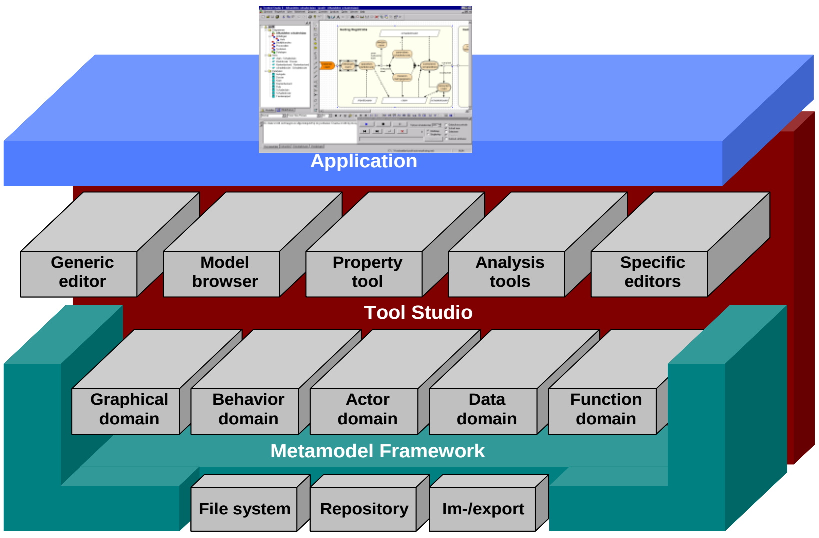
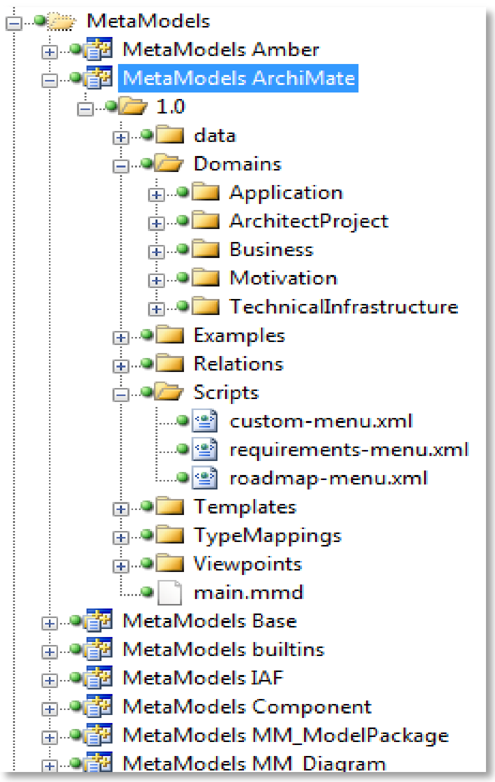


Figure 2-5 BiZZdesign tool architecture (BiZZdesign, 2011)

BiZZdesign's service lines are aligned to the company's mission: to help organizations govern and change (themselves) effectively using enterprise architecture, business process management and governance supported by methods, tools, consultancy, and training (BiZZdesign, 2012). The enterprise knowledge in BiZZdesign is mostly involved in its software tools and their content portal, InSite. InSite is a viewer of the contents of repository from both Architect and BiZZdesigner, the modeling tools. BiZZdesign's modeling tools enables creation and/or modification of architecture models and business process models.

2.7.2 BiZZdesign repository

The repository is used with every comprehensive use of modeling tools and the InSite portal. Figure 2-5 shows how BiZZdesign's tool architecture is structured. Enterprise knowledge contained in business process and architecture models from the two modeling tools are stored in a repository as metamodels. A sample structure can be seen in Figure 2-6. BiZZdesign repository is currently developed to provide services to be used by external applications, changing from currently single-purpose repository client to a repository enable to couple with multiple tools for sharing knowledge. This coupling to multiple tools will be enabled with a service.



```
MetaModels
    MetaModels Amber
    MetaModels ArchiMate
        1.0
            data
            Domains
                Application
                ArchitectProject
                Business
                Motivation
                TechnicalInfrastructure
            Examples
            Relations
            Scripts
                custom-menu.xml
                requirements-menu.xml
                roadmap-menu.xml
            Templates
            TypeMappings
            Viewpoints
            main.mmd
    MetaModels Base
    MetaModels builtins
    MetaModels IAF
    MetaModels Component
    MetaModels MM_ModelPackage
    MetaModels MM_Diagram
```

Figure 2-6 Sample repository structure (BiZZdesign, 2011)

The enterprise knowledge will then be captured in a metamodel with a hierarchy depicted in Figure 2-7. Red square indicates static metamodels and the green one shows the dynamic metamodels. Further on the hierarchy, Built-ins are data types of a programming language (long, bool, etc.). Base includes metamodel object, metamodel relation, and metamodel reference object. MM_Diagram (metamodel diagram) is comprised of metamodel graphic, metamodel edge, and metamodel compound. On the same level, MM_ModelPackage can include metamodel model package, metamodel model, and metamodel module.

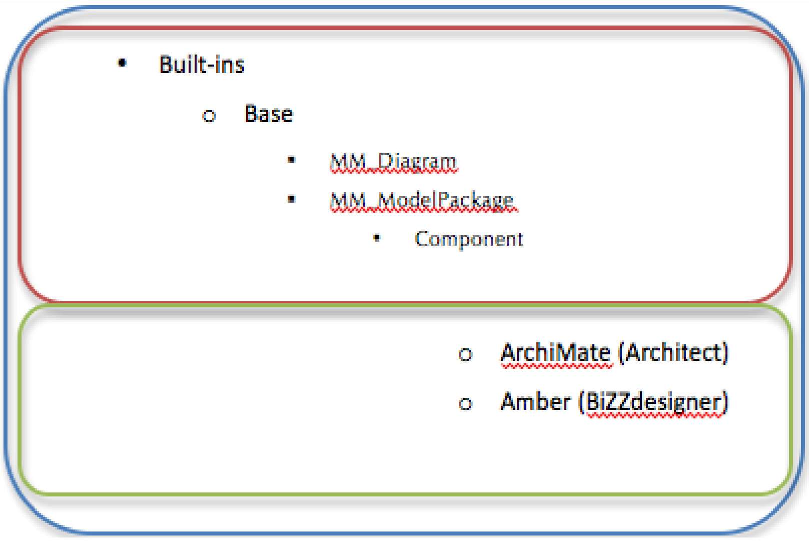


Figure 2-7 Metamodel hierarchy

BiZZdesign tool architecture is a generic tool framework for graphical modeling tools. This tool architecture is extensible and configurable by model engineers. Main functions are developed as pluggable tool components (BiZZdesign, 2011). Among the technologies used by this tool architecture are Windows GUI front ends (C++), Repository (shared file, relational database), Web application server (Java), and Browser client (Flash).

2.8 Summary

In this chapter, we have discussed literature on knowledge, knowledge visualization, repositories, and enterprise knowledge embedding. Repositories play an important role in the relation between knowledge visualization tools and knowledge management. This chapter closes with description of current situation at BiZZdesign, a company in the Netherlands with focus related to this research.

3. Reference Architecture for Enterprise Knowledge Visualization

After reviewing literature from previous chapter, we continue with designing reference architecture to visualize enterprise knowledge. In this chapter, we define the aspects of designing reference architecture for knowledge visualization. We then specify the required components and why they are required in a reference architecture for visualizing enterprise knowledge.

3.1 Preliminaries

In section 2.1, we mentioned that we relate externalization and combination conversion type from Nonaka (1994) to knowledge organization and embedding. Knowledge combination and externalization result in the spread of explicit knowledge within an enterprise. Meanwhile, Zack (1999) argues that explicit and tacit knowledge need to have the right balance in an organization to maintain competitive advantage. Enterprise knowledge is then embedded in a knowledge format. This knowledge format is then visualized with the help of a visualization tool.

3.1.1 Managing enterprise knowledge to be visualized

Interactions and activities in an enterprise defined by Firestone (2000) can have complexities that require the presence of a knowledge management method. Andersson et al. (2004) discuss about integration of business process support (BPS) with knowledge management and come up with three assumptions below.

For the knowledge management to be of use in an organization it should be seamlessly incorporated in everyday business activities, for a business support system to be able to automatically gather and distribute knowledge, both the business and support system should be process-oriented, and important prerequisite for a knowledge management system (KMS) will function in practice is that a majority of the organization staff actually use the system in a major part of their daily work.

The above assumptions are then derived into an integrated KMS/BPS that consists of historical database, principle of planning, and navigation system (Andersson et al., 2004). Section 1.1 provided definition of management of enterprise architecture that includes regulating and advancing IT support. Knowledge visualization, including managing knowledge to be visualized, is considered part of both processes. Section 2.6 provided description and examples of how enterprise knowledge can be embedded and then, later, stored in a repository. Enterprise knowledge facilitated in this design is that contained in models. Enterprise model is described by Fox and Gruninger (1998) as:

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise. It can be both descriptive and definitional-spanning what is and what should be.

The definition above implies that enterprise model can contain information that is related to interactions and activities in an enterprise. This will be further discussed in subsection 3.2.1.

3.1.2 Visualizing models containing enterprise knowledge

As discussed in section 2.3, Kurfess (2008) argues that visualization is one way to present knowledge. As a mean capable of visualizing knowledge, wiki has also been discussed in subsections 2.4.1 and 2.4.2. However, in this chapter, the reference architecture designed is not solely oriented to wiki software products. Instead, we believe that any visualization tools intended for models that contain enterprise knowledge should be capable of accessing stored knowledge. As section 2.5 underlined, we consider repository of knowledge to be a generic element that can facilitate storage of enterprise knowledge. It can also have the flexibility to handle knowledge embedding in form of models.

Based on those viewpoints, models are observed. They can consist of objects, relations, texts and more importantly graphical representation, as explanation in section 2.6 implies. These properties are complemented with explanations that can also be enclosed in those models. How those properties can be generated and subsequently visualized will be discussed in section 3.2.

3.2 Enterprise knowledge visualization reference architecture

In previous section, we discussed relation of architecture elements and visualization, and how knowledge management works with visualization. This section will start with definitions of reference architecture. We start with a definition by Bass et al. (2003) who stress on software elements and their functionality:

A reference architecture is a reference model plotted onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them. While a reference model splits the functionality, a reference *architecture is the plotting of that functionality onto a system decomposition. The plotting may be (although not necessarily) one to one. A software element may implement part of a function or several functions.*

Gallagher (2000) coins another definition with emphasis on domains and instantiation aspects.

A reference architecture is the generalized architecture of several end systems that share one or more common domains. The reference architecture defines the infrastructure common to the end systems and the interfaces of components that will be included in the end systems. The reference architecture is then instantiated to create a software architecture of a specific system. The definition of the reference architecture facilitates deriving and extending new software architectures for classes of systems. A reference architecture, therefore, plays a dual role with regard to specific target software architectures. First, it generalizes and extracts common functions and configurations. Second, it provides a base for instantiating target systems that use that common base more reliably and cost effectively.

We then define reference architecture for enterprise knowledge visualization as depiction of reference for elements and functionality of architecture components with concern on instantiation for systems carrying out the task of visualizing enterprise knowledge embedded in models.

The reference architecture is designed with ArchiMate, which has an underlying framework that consists of business, application, and technology layers (Iacob et al., 2004). Moreover, components for each layer are identified based on the literature study on enterprise knowledge, knowledge visualization, and knowledge repository. We design this reference architecture with consideration of its quality. The followings are quality goals that this reference architecture should have:

High availability (QG1)

To manage knowledge in an enterprise that is highly dynamic, a repository needs to be ready to be called at anytime. It should have a system that has a high uptime. To achieve this, hardware redundancy and software redundancy as suggested in a case study by Bass et al. (2003) can be chosen as a solution. In an ArchiMate model, this will affect technology layer including infrastructure, external infrastructure services, and application components and services. Software redundancy and hardware redundancy on enterprise knowledge visualization can be achieved by using fault detection, fault recovery, and fault prevention methods mentioned by Bass et al (2003).

Capability to handle and manage models and properties (QG2)

As mentioned in section 2.5, design of a repository should enable sufficient knowledge recording capability. This design of repository should be tailored specifically to handle models. Functionality of repository should include exchange, retrieval, and access of models.

Not only models, a repository should also be able to manage objects that models can contain. It should be able to give information on specific properties of models. This can include relation between objects, sequences, layers, pointers, etc. Goul and Corral (2007) defines goal of model management as:

The goal of model management is to develop a generic infrastructure that offers an order-of-magnitude productivity improvement to builders of model-driven applications, such as database tools, application design tools, message translators, and customizable commercial applications.

Properties included by models should include graphical representation. A separate function of repository to generate this should be allocated. Graphical representation of models will then be passed to or obtained by visualization tool(s). In order to have both capability to handle models and capability to manage models and properties, the models themselves need to have sufficient definition so that it can be used and presented in different scenarios (Zack, 1999). A method to achieve those is by having specially assigned element of repository for models.

Visualization process discussed in this chapter and used as a base for reference architecture is based on a framework for knowledge visualization suggested by Eppler & Burkhard (2004). Based on that framework, we specify the knowledge type (enterprise knowledge), visualization goal (codifying the knowledge), and visualization format (conceptual diagrams) used in this reference architecture. Eppler and Burkhard define conceptual diagrams as:

Conceptual diagrams, by contrast, are abstract, schematic representations used to explore structural relationships among parts. They help reduce complexity, amplify cognition, explain causal relationship and to structure information. The type of knowledge that is conveyed by conceptual diagrams is analytic and their format is thus highly structured and systematic.

Therefore, repository should provide a graphical representation quality that is sufficient and able to be handled by visualization tools. This leads to the next quality goal: flexibility of the system.

Flexibility (QG3)

Repository system built based on this reference architecture should be able to couple with various visualization tools. Variety of visualization tools requires graphical representation to have necessary requirements. For this reference architecture, a selection from a set of requirements stated by Bobrik et al. (2006) is sufficient. First, repository system should be able to facilitate use of symbols and provide functions for color, fonts, and symbol adaptation. Next, it should provide precise rules to handle possible clash when a process element is being accessed and it should select relevant visualization symbols based on elements of model.

Applicability (QG4)

This reference architecture will need to be applicable in different enterprises and on different repositories. Therefore, modularity of repository system, including its functions and services is important in order to visualize enterprise knowledge and models. Bass et al. (2003) suggest template providing and cautiously handing over responsibilities to modules to achieve this applicability quality goal.

3.2.1 Requirements for repository

Aiming for quality goals of high availability, capability to handle and manage models, flexibility and applicability would mean a generic repository design is needed. This variety of model can include data model, reference model, business process model, and other types of model containing enterprise knowledge. Fox and Gruninger add that from practical point of view enterprise model must be able to correspond to things that are planned, anticipated, and that has occurred in an enterprise. Activities inside an enterprise should be covered by knowledge contained in enterprise models (Fox & Gruninger, 1998).

Repository system is fundamental in our enterprise knowledge visualization reference architecture. In order to produce a generic repository system design to facilitate enterprise knowledge visualization, following requirements related to business domain (van Wasbeek, 2006) are necessary.

Increase system quality (RR1)

A repository should have necessary functions or services that can lead to overall system quality. In practice, a repository can be used by more than one visualization system. Thus, this requirement is closely related to support for system expandability, the next requirement of repository in business domain.

Support for system expandability (RR2)

An enterprise with different activities and interactions is highly open to being expandable. Therefore, it is important to consider a repository design to support expandability in the future. This requirement and increase system quality requirement correspond to applicability quality goal of reference architecture.

Increase system lifecycle (RR3)

Contents inside a repository can change as an enterprise or an organization change their activities or business. They can also change when the organization change the way they carry out its business. Albeit those prospective changes, a repository needs to be designed so that a certain system or a number of systems that use it can be maintained easily. Therefore, it should have high modularity and simplicity. This requirement corresponds to high availability, flexibility and applicability quality goals.

Decrease task completion time (RR4)

Explanation in section 2.5 provides arguments to support that repository can help to manage knowledge. Referring to the argument by Nonaka (1991) in section 2.1, management of knowledge will then be crucial to an organization's competitive advantage. This management of knowledge can thus relate to completion time for finishing projects or tasks. Therefore, a repository needs to be designed with consideration of task completion time. To quality goals of reference architecture, this requirement corresponds with high availability, capability to handle and manage models and its properties, and flexibility.

Van Wasbeek (2006) provided other supporting requirements for repository to manage artifacts that we consider relevant for our reference architecture. A repository capable to provide management for any type of models should enable metadata information addition, provide facilitation on relation between knowledge units, and provide container for model definitions.

Metadata information addition (RR5)

To add metadata information in a model, basic information from where those metadata are obtained has to be understood. Metadata information can be used to identify models and will also be useful for repository search function. This is covered by a literature describing a tool for enterprise modeling by Rospocher et al. (2009). The literature provides explanation of aspects that can be added as metadata information. Rospocher et al. state that enterprise model should describe:

The domain, the processes and the competencies of an enterprise.

Those three properties will be described one by one. Domain of an enterprise covers explanation of business domain that an enterprise is involved in. Operations within an enterprise contain inherent knowledge that should be stored. This knowledge is consisted of conceptualization of connections between units within an enterprise, including information that can be derived from those units. The business domain has a set of underlying constraints and set of working procedure, including which things should be addressed when carrying out certain tasks. This needs to be captured in relations, objects, and concepts of an enterprise model (Rospocher et al., 2009).

A business domain of an organization consists of tasks that are defined or being planned. Process of an enterprise includes steps of tasks that are carried out by previously assigned roles through task assignment breakdown. Sequences or steps of tasks that are carried out also exist in enterprise models. Rospocher et al. address this and mentioned that procedures and outline in a business domain can complement metadata information of enterprise models (Rospocher et al., 2009).

An enterprise cannot be separated from people involved in organizational activities.

The matter concerning this involvement is related to how capable those people are. In order for organization to fulfill their purpose, its employees need to have certain capability and attitude. Rospocher et al. also add that there is a relation between domain and process information of an enterprise model. They state that this connection is facilitated by information of competency. That kind of metadata information is crucial because it covers skills of a person of group of people and his/her/their level of knowledge to perform a certain assignment. This metadata information is motivated by the fact that an individual or group of people undergoes a learning process in carrying out an assignment of set of tasks (Rospocher et al., 2009).

Those three metadata information can be added to an enterprise model. The level of width and/or depth of that information depend on how an organization would like to have their goals fulfillment breakdown and how detailed an organization want to evaluate their performance. An enterprise model is a medium for those, as the definition from Fox and Gruninger (1998) implies. To the quality goals of reference architecture, this requirement corresponds to capability to handle and manage models and its properties.

Relation between knowledge units (RR6)

Zack (1999) states that a sufficient knowledge recording capability is important for a knowledge repository. As the summary enterprise domain from Rospocher et al. underlines, models containing enterprise knowledge can contain relation of knowledge units. Zack adds that these knowledge units have a variety of attribute that reflects the context and the kind of explicit knowledge stored.

Since an organization can store explicit knowledge in many different formats, it is crucial for the repository to have the adaptability and flexibility to be able to take in those different formats. A supporting application or function to validate those formats can also be helpful. For an enterprise, this can provide more orderliness in their knowledge management. For people concerned with repository and its supporting components, this will be beneficial for further possible uses of repository contents.

Orderliness in knowledge storage is also indirectly implied by Dingsoyr and Royrvik (2003) who realizes that knowledge repository gathers its content from collection and with integration from many different sources of knowledge.

An issue of knowledge input process should also be taken into account. Van Heijst et al. (1997) mention two kinds of possible input process. The first is a judgment of adequacy from involved workers on related knowledge and which of those should be stored within a repository. Second approach is selecting knowledge by assigning a group of people that have necessary required knowledge to sort from abundance of knowledge medium formats and gather them.

The last issue gives another point of view on how a repository can properly store relation between knowledge units. The first input process choice might be a solution for defining relation. However, an organization needs to make sure workers with related knowledge can allocate a portion of their human resource to be able to sort which knowledge is related to which domain(s). To optimize the first input process, a role to administer knowledge division in respective organizational division should be considered. It would be better if this input process is thoroughly planned, so that workers can have knowledge medium formats that are already sorted or tagged to ease their task of selecting which knowledge to be stored in repository.

The second input process choice requires a separate process for selecting the right group(s) of people. These people need to have comprehensive understanding of the enterprise. This understanding should include not only in depth comprehension on specific domain or group of domain, but also an understanding of how different domains collaborate and assist each other. This kind of understanding is usually present in people with a significant amount of experience in the organization. People who built their career within the organization would make a good candidate.

Another possibility for the second input process choice is to outsource an expert on specific domain or group of knowledge. In choosing the expert(s) to outsource to, a background of repository knowledge can also be taken into consideration. He or she or they might come from a sector that has close relation to that of the organization. A combination between internal personnel and external expert to form a group of people to sort required knowledge is also an option, although this might require them to spend a preliminary time to work with each other.

Overall, in facilitating relation between knowledge units, it is fundamental for a repository to have adaptability, flexibility, orderliness, and relevant choice of input process. These are needed to ensure the repository contain knowledge that will be useful for its further uses and ready for any adjustment in knowledge medium formats in the future. This requirement therefore corresponds with capability to handle models and its properties of the quality goals of reference architecture.

Properly defining relation between knowledge units as an input to repository is prominent for future uses of repository because it will determine the structure of repository. Since a repository needs to provide its contents to application or stakeholders, a relevant structure or set of structures would be needed. An enterprise can be different to other enterprise in the level of complexity of its activities and processes. This level of complexity leads to variety of knowledge and knowledge medium formats (Dingsoyr & Royrvik, 2003). A case of a repository having different structures for different set of stored knowledge is therefore possible.

Structure decision needs to be taken carefully. The reason for this is because it will determine efficiency of repository access. This decision will also determine the cost for building a repository. Not only that, extra role might need to be added if repository structure is too complex. The more complex a repository structure, the more resources need to be allocated to administer or manage repository. Because structure decision is prominent, stakeholders need to allocate certain resources to repository design and knowledge medium formats that it contains or will contain. Examples of variety of different knowledge medium formats are provided in section 2.5.

Container for model definition (RR7)

The variety of interactions and activities within an enterprise are purposed for solving problems in an organization. To solve those problems, certain decisions should be made. Definitions from Fox and Gruninger (1998) shows that enterprise models contain set of properties that can aid in making those decisions.

However, those set of properties can be contained in a way that makes decision making more difficult. Along with metadata information, this repository requirement is also important for identification. Difficulties in identifying can lead to decision making constraints and eventually can cost more time and eventually would not be beneficial for an organization.

In order to support the decision making, a repository should contain definition of those properties so that users can understand models better. To do so, a container for model definition should support checking and enclosing of properties. For example, a model can belong to specific organizational body or department and might only be viewable by users with specific privileges.

Time information is also an important part of model definition (Fox & Gruninger, 1998). An organization can run its business better if there is clarity on which models are related to activities in the past, which models are related to current activities, and which models are related to future plans. Related to time information, a proper management of past models can be useful for the management to perform evaluation and plan future activities. Overall, to the quality goals of the reference architecture, this requirement corresponds to capability to handle and manage models and its properties.

Moreover, a generic repository should be independent to any database or server, should be able to include support for inconsistency inspection, should not be difficult to configure, and should support impact analysis (van Wasbeek, 2006).

Independency (RR8)

Repository system in our reference architecture should not be restricted to a specific database or server. It should be able to couple with any kind of server installed on any platform. Format of database should also not be a constraint for future systems basing their implementation on this reference architecture.

For this requirement to be fulfilled, the person or group of people who have the responsibility to design the repository should anticipate prospective server and database change and/or configuration. Expansion of current system and current enterprise scope should also be taken into account. This requirement of independency corresponds to the flexibility and applicability quality goals of the reference architecture.

Support for inconsistency inspection (RR9)

An inconsistency in repository contents can cause them to be unable to be accessed by users or adjacent applications. Although methods of backing up the repository can be taken as an approach to prevent this, it still needs to be complemented with an inconsistency inspection. This requirement corresponds to capability to handle and manage models and its properties and also the quality goal of high availability.

A reason for this because backing up, however automatic it would be, will still consume an amount of time. In enterprises with little or zero fault tolerance, this can lead to organizations not being able to provide service properly and/or not being able to conduct their business to their tightly measured plan and budget. Thus, inconsistency inspection is also necessary for repository system.

Ease of configuration (RR10)

A repository system can have complexities behind its interface. A certain set of procedures and packaging of functions need to be conducted before a repository system is installed. It is prominent so that people assigned with the task to implement or configure the repository can have their job performance enhanced. The procedures and packaging can also improve applicability of overall reference architecture (fourth quality goal).

Furthermore, an ease of configuration can help the job of person or group of people assigned to manage the repository. The argument from Dingsoyr and Royrvik (2003) that implies on variety of knowledge medium formats also corroborate the need for configuration ease. Enterprise with various knowledge medium formats can use the help of reference architecture with a repository system that is easy to set up.

Ease of configuration can also cover structures of repository. A generic repository design should be able to facilitate various knowledge storage structures. Therefore, a configuration interface to help choose relevant repository structure is important in this reference architecture.

Support for analysis on change request (RR11)

Another aspect of flexibility is anticipation support for effects produced by change requests. A repository is made to be accessed by multiple users and to handle change requests, regardless of their numbers and their simultaneity. This is crucial so that future systems will have better load management and able to prioritize on which change to carry out first. A queuing system can also be taken as a possible approach, so that repository system can prioritize changes that are requested to it.

In order to fulfill this requirement, a repository should include a list of requirements for a change to happen. Not only that, it should also store lists of which files, which components of systems and which items that can be affected by a change request. For overall quality goal of reference architecture, this requirement corresponds to capability to handle models and its properties.

To complement previous requirements, a generic repository should also be able to provide interface for repository manager to edit its contents and configurations. It should also provide support for documentation of models (van Wasbeek, 2006).

Support for documentation (RR12)

As the example in section 2.4 provides, a wiki implementation example in ArchiXL shows the presence of documentation. Fulfillment of this requirement will enhance the performance of user related to visualization tool use. Although documentation of models can be simple, it is still important to facilitate the use of symbols, as Bobrik et al. (2006) underline.

A documentation can be embedded as properties of models (Rospocher et al. 2009). This documentation is a mean to understand a model better. As mentioned in container for model definition requirement, users that have certain tasks related to making decision can benefit from the fulfillment of this requirement. Therefore, this requirement also supports the second quality goal of the reference architecture (capability to handle and manage models and its properties).

Editing interface (RR13)

To manage its contents, there should be a role assigned specifically. This role will have special privileges to handle models and its properties. Therefore, it corresponds with the second quality goal of reference architecture.

Models, including its properties, come from different sources of knowledge across an organization, as Dingsoyr and Royrvik (2003) point out. For that reason, the role assigned to edit the repository also need to have a comprehensive understanding, at least in regard to structure or set of structures of the repository.

Quality goals	Corresponding repository requirements
High availability (QG1)	RR3, RR4, RR9
Capability to handle and manage models and its properties (QG2)	RR4, RR5, RR6, RR7, RR9, RR11, RR12, RR13
Flexibility (QG3)	RR3, RR4, RR8
Applicability (QG4)	RR1, RR2, RR3, RR8, RR10

Table 3-1 Mapping of quality goals of reference architecture and requirements of repository

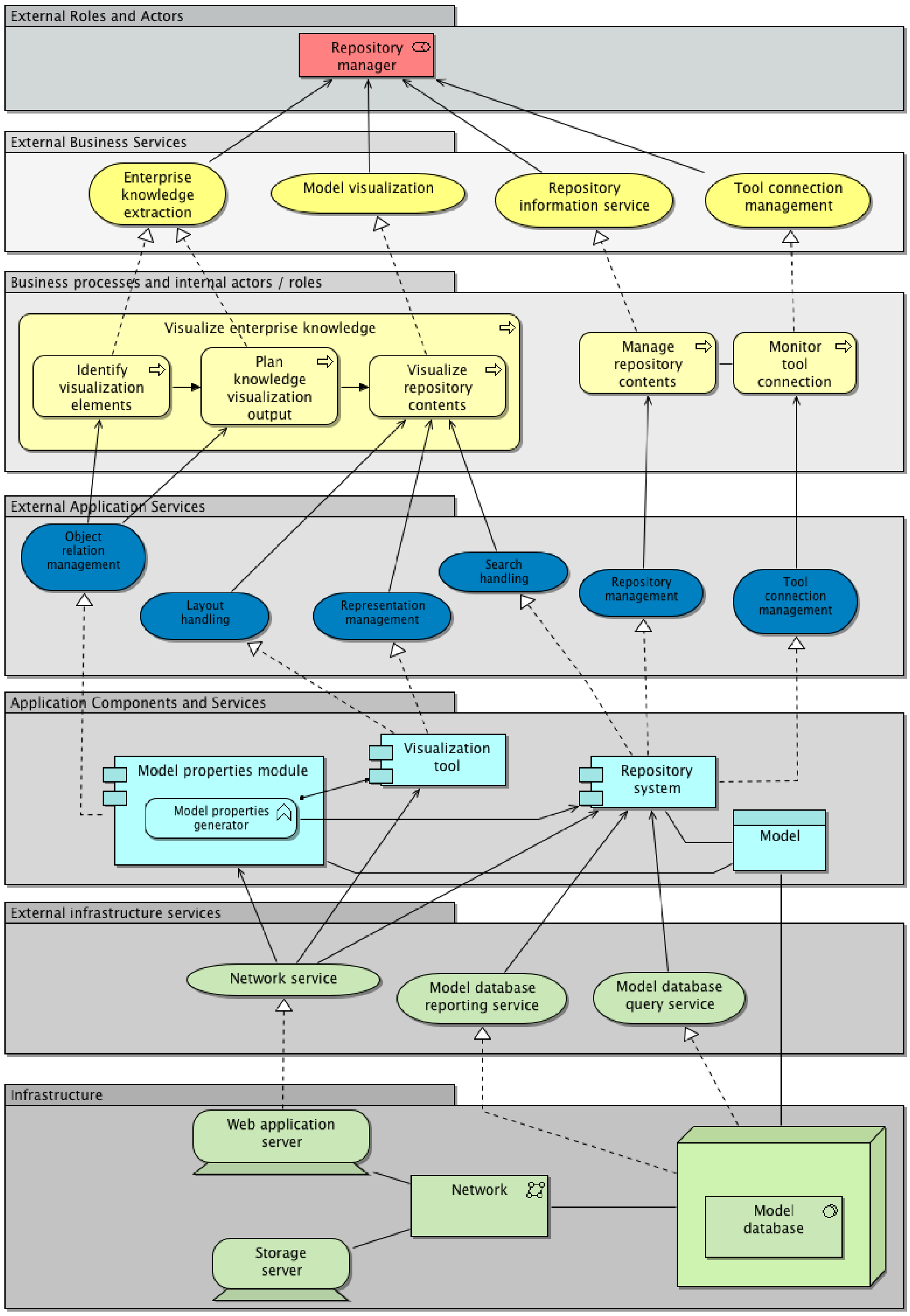
Symbol	Requirement name
RR1	Increase system quality
RR2	Support for system expandability
RR3	Increase system lifecycle
RR4	Decrease task completion time
RR5	Metadata information addition
RR6	Relation between knowledge units
RR7	Container for model definition
RR8	Independency
RR9	Support for inconsistency inspection
RR10	Ease of configuration
RR11	Support for analysis on change request
RR12	Support for documentation
RR13	Editing interface

Table 3-2 Requirements for repository

Table 3-1 provides a mapping of quality goals and corresponding repository requirements. List of repository requirements can be found in Table 3-2.

3.2.2 Business layer of reference architecture

To be able to properly manage and handle models and to monitor the uptime of repository system, a role needs to be assigned to administer the repository. This is the base for defining repository manager role. For enterprise knowledge visualization, external business services used by repository manager would be related objects/models from repository, additional enterprise knowledge in form of attributes/properties, and repository information itself. These business services are based on the first three quality goals of reference architecture.



External Roles and Actors
Repository manager
External Business Services
Enterprise knowledge extraction
Model visualization
Repository information service
Tool connection management
Business processes and internal actors / roles
Visualize enterprise knowledge
Identify visualization elements
Plan knowledge visualization output
Visualize repository contents
Manage repository contents
Monitor tool connection
External Application Services
Object relation management
Layout handling
Representation management
Search handling
Repository management
Tool connection management
Application Components and Services
Model properties module
Model properties generator
Visualization tool
Repository system
Model
External infrastructure services
Network service
Model database reporting service
Model database query service
Infrastructure
Web application server
Network
Storage server
Model database

Figure 3-1 Enterprise knowledge visualization reference architecture

In relation to the quality goals of reference architecture, repository information service corresponds to high availability. Furthermore, model visualization corresponds to capability to handle and manage models and enterprise knowledge extraction corresponds to flexibility quality goal.

Requirements for repository include those that require a role to manage or administer the repository system. Among them are inconsistency checking, repository ease of configuration, and analysis of change request. Thus, we define repository manager role for the external actors and roles on the business layer and model visualization, enterprise knowledge extraction, tool connection management and repository information service as external business services.

To be able to monitor repository, its connection with tools, and overall visualization process, there needs to be separate relevant business processes. We believe that in any visualization system of enterprise knowledge, elements of visualization need to be identified first before visualization output is planned and the actual visualization is conducted. Both identification of visualization elements and planning of its output are realized by enterprise knowledge extraction business service, which is a component summarizing the bridge between repository and visualization process.

For the repository and visualization tool to be able to be monitored by repository manager, business processes related to repository content and tool connection are needed. Visualization process, manage repository content and monitor tool connection should be included as business processes in business layer.

3.2.3 Application layer of reference architecture

In section 2.5, we discussed knowledge repository and one of its necessity is that it needs to have sufficient knowledge recording capability. To support the need, the repository must be able to store object relation and representation. Both require different instances from overall management of the repository, because we assume that object relations are stored with different method than representations and that object relation will relate directly to navigation system in the visualization system, as Andersson et al. (2004) suggest. Therefore, repository management, object relation management, and representation management are identified as required external application services. Those three application services correspond with capability to handle and manage models and its properties, our second quality goal for the reference architecture.

Repository management application service and manage repository contents business process will play their part for privilege-related concerns. To make sure conceptual diagrams in models are visualized properly, components layout handing and search handling are needed as external application services of application layer. Repository system should also be able to manage tool connection.

Generally speaking, a repository system is essential in providing extraction of enterprise knowledge. The repository system in this reference architecture is designed to be built based on requirements for repository in subsection 3.2.1. Those thirteen requirements for repository (see Table 3-2) are necessary to be fulfilled to optimize repository system's functionality.

In this reference architecture, visualization tool and model properties module as application components, realized by layout handling and representation management, and object relation management respectively have assignment relation. A data object of Model is needed as resource for generation of enterprise knowledge to be visualized. This data object is associated with repository system and model properties module and should also be part of this reference architecture's application layer.

3.2.4 Technology layer of reference architecture

Repository system will need solid database infrastructure and services. For external infrastructure services, both database reporting service and database query service will be needed to support repository system in that case. Additionally, a network service would be used by both visualization tool and repository system for network-related access. All three of network service, database reporting service, and database query service are used by repository system application component. In some systems basing its architecture on this reference architecture, network service might also have possibility to be used by either visualization tool application component or model properties module application component, or both. Therefore, we depict its relation to those two components.

Meanwhile, for infrastructures, web application server (to provide the necessary input for especially repository system), model database, and network are the main infrastructure components for this reference architecture. Model database has association with Model data objects, which is essential for generating enterprise knowledge. Also, a storage server is needed for flexibility of web application server and possibility to connect with separate model databases.

3.3 Discussion and summary

A reference architecture comprised of component for knowledge visualization was provided in this chapter, supported with arguments from related literature. We started the discussion in this section by asking whether the reference architecture built indeed can be referred. We do this mainly with looking into elements and their functionality, so that future systems can be built to visualize enterprise knowledge.

Regardless of size of organization that has a repository storing enterprise knowledge, we assume that there has to be a repository manager role. The supporting business processes realizes four main business services that repository manager should at least address in achieving the task of visualizing enterprise knowledge. Those business services matches with quality goals defined in section 3.2.

An organization can have different types of knowledge inside its repository. A system built based on our reference architecture will require to have specially assigned part of repository to handle models if that repository does not only contain models.

Domokos & Varro (2002) also briefly discuss the role of layouter and renderer components. The mentioned components were thus translated in the components for business layer and application layer, namely identify visualization elements and knowledge visualization output planning business processes, metamodel generator and image generator applications, and layout management and representation management application services.

Moreover, application services, application components, and infrastructures follow the concepts from our four quality goals. We consider technology layer (application components, external infrastructure services, and infrastructure) in particular to suffice the need to give facilities for system with repository and model with objects, relation and, graphical representation. The consideration is based on findings from Yan et al. (2009).

Overall, quality goals and tactics has been defined and chosen for this reference architecture. These four quality goals and their respective tactics to achieve them can be taken as approach for future systems basing their implementation on this reference architecture. A set of requirement for repository system, a key component in this reference architecture, have also been described and mapped to the quality goals.

In this chapter, we have provided a reference architecture for enterprise knowledge visualization. The reference architecture is built based on quality goal defined in section 3.2. Overall, the reference architecture designed gives generalization of possible several end systems, provides a base for future implementation, and facilitates derivation of those future systems. Thus, we conclude that using our reference architecture, a system carrying out the task of visualizing enterprise knowledge can be built on repositories storing model with enterprise knowledge contained.

4. Enterprise Knowledge Visualization at BiZZdesign

In this chapter, an implementation of enterprise knowledge visualization reference architecture is described. The implementation, carried out at BiZZdesign, is based on the reference architecture provided in chapter 3.

4.1 Wiki and MediaWiki

Interest in wiki-based systems is growing steadily since Wikipedia was founded. The number of wiki software is growing and not only in commercial organizations but also in the software engineering community (Farenhorst & van Vliet, 2008).

In this research, we use MediaWiki due to its extensibility (Boulain et al., 2006) and ability to handle external knowledge formats. MediaWiki enables those with series of extensions made by developers. Updates of architectural views or any models and/or contents are handled by an external service, so that MediaWiki can focus on handling visualization of enterprise knowledge. Generally, there are extensions for enterprise knowledge extraction. These extensions can accommodate various external knowledge formats and bridge them to pages in MediaWiki. One of them is capable to extract XML tags and will be part of implementation section (section 4.3).

MediaWiki is different from a structured wiki. A structured wiki aims at combining wiki with advantage of a database application. MediaWiki's shortage on that feature is balanced with simplicity and look and feel (Louridas, 2006).

MediaWiki can support the case at BiZZdesign (see section 2.7) by complementing the use of repository by InSite portal and tools mentioned in subsection 2.7.1. It can cover two assumptions (see subsection 3.1.1) from Andersson et al. (2004). Assumptions covered are the first (integration in business activities) and the last (actual use of the system). Louridas (2006) supports this coverage by arguing that it can be integrated with ease and have a tendency to be used by users in an organization.

4.2 Business process models and architecture models

Models can aid us in understanding the enterprise and activities inside it. A business process model depicts task or series of tasks that are to be completed, including sequence of those tasks (Dijkman & Joosten, 2002). An example of business process model is available in Figure 4-1.

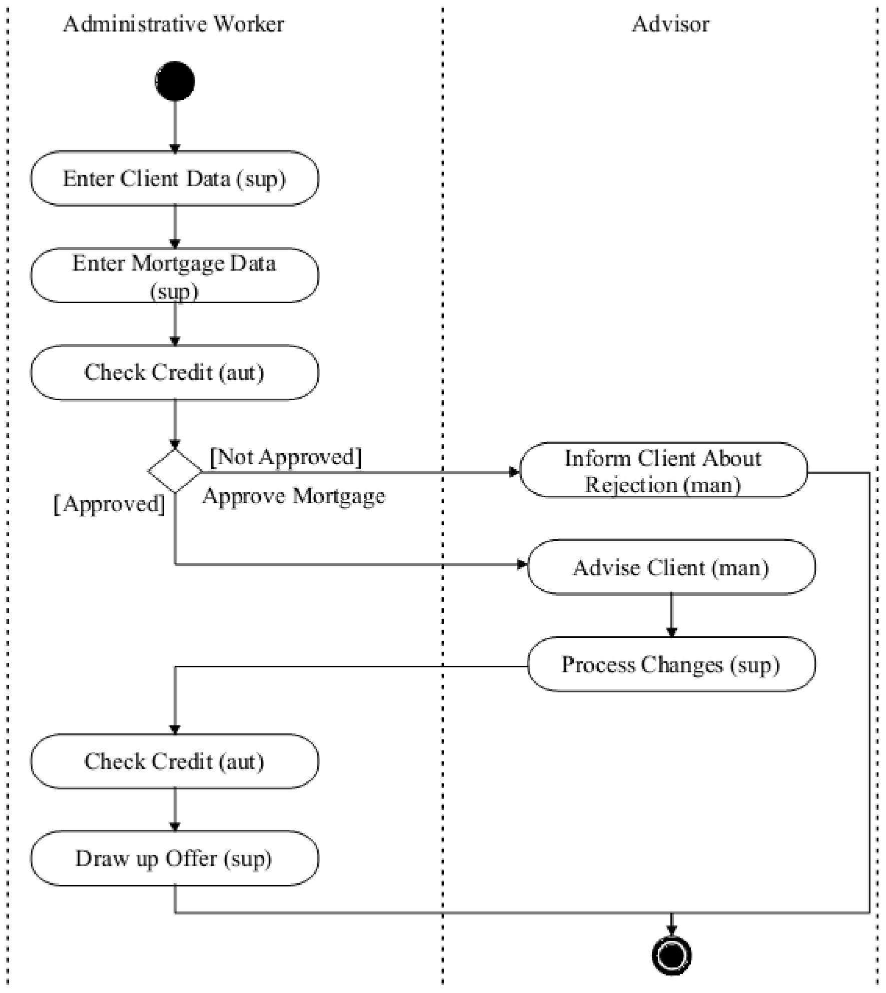


Figure 4-1 Example of a business process model (Dijkman & Joosten, 2002)

Architecture models, along with specification documents, constitute a set of architecture artifacts. It also provides a holistic view of structure and function of a part of an enterprise (Iacob et al., 2007b). It is mainly used to depict strategy from a set of organization components or elements. The model of an architecture also facilitates communication of its components (Gorton, 2011). Parts of an enterprise can be facilitated by architecture models. An example of an architecture model that is a reference architecture can be seen in Figure 4-2. Business process models and architecture models are needed to communicate to intended stakeholders of the enterprise. A good structure, storage, and visualization can support that communication.

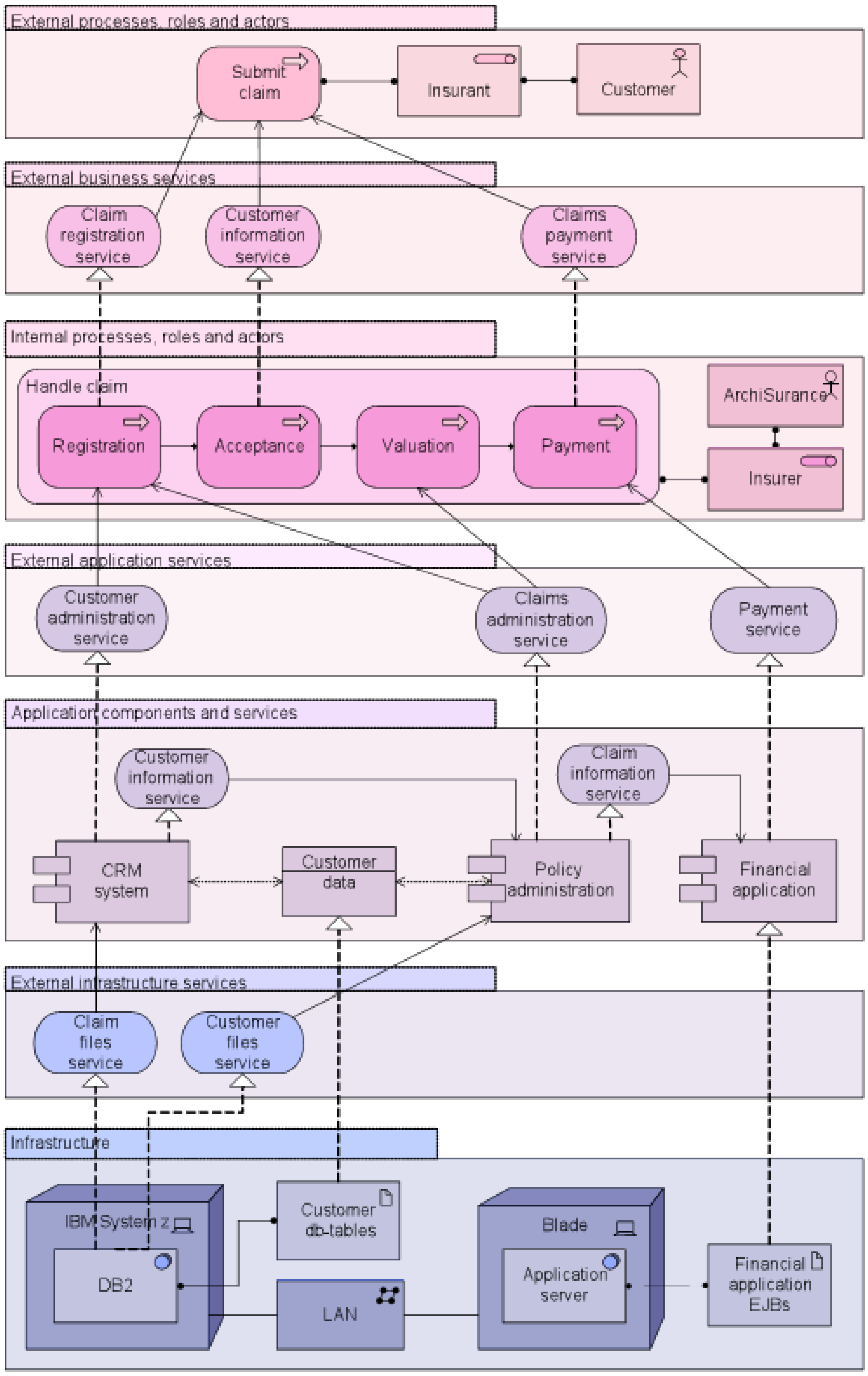

External processes, roles and actors
Submit claim
Insurant
Customer
External business services
Claim registration service
Customer information service
Claims payment service
Internal processes, roles and actors
Handle claim
Registration
Acceptance
Valuation
Payment
ArchiSurance
Insurer
External application services
Customer administration service
Claims administration service
Payment service
Application components and services
Customer information service
Claim information service
CRM system
Customer data
Policy administration
Financial application
External infrastructure services
Claim files service
Customer files service
Infrastructure
IBM System z
DB2
Customer db-tables
LAN
Blade
Application server
Financial application EJBs

Figure 4-2 Example of an architecture model (Lankhorst at al., 2009)

4.3 Implementation of enterprise knowledge visualization using wiki software

Implementation of enterprise knowledge visualization at BiZZdesign is done referring to the waterfall software engineering method (See Figure 4-3). This implementation is based mainly on external application services and business processes and internal actors/roles from the reference architecture on chapter 3. Referring to the definition by Gallagher (2000), our end system will be called prototype in the remainder of this chapter.

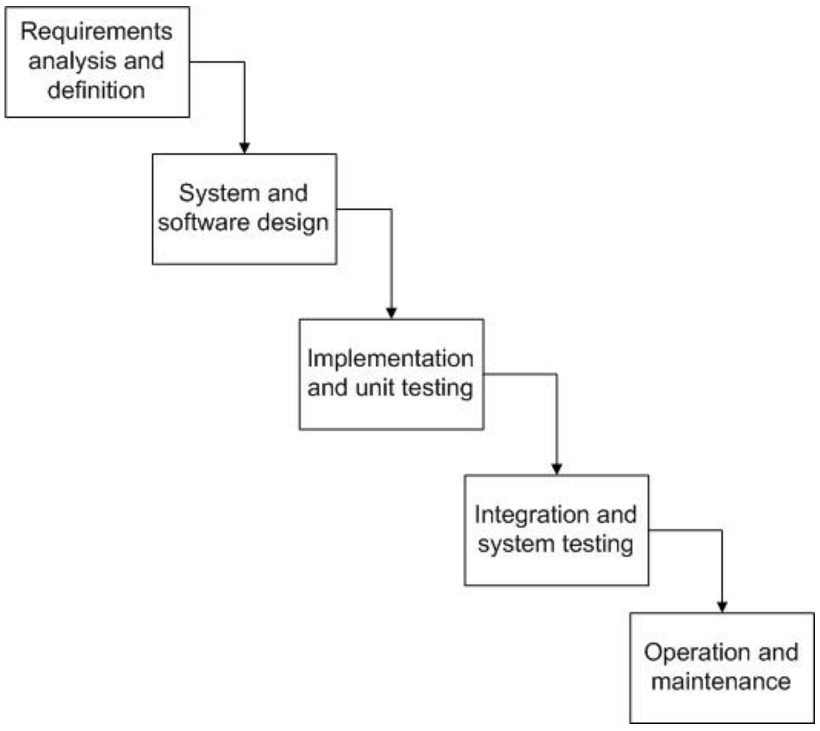

Requirements analysis and definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 4-3 Waterfall software development life cycle (Huo et al., 2004)

4.3.1 Requirements analysis and definition

4.3.1.1 Stakeholders

Related stakeholders in this implementation include BiZZdesign repository team (programmers, designers) and BiZZdesign management. Repository team and management aims to test repository structure and preliminary functions in form of a prototype for use with external tools and to have a prototype that visualizes repository contents using wiki software.

4.3.1.2 Requirements

BiZZdesign needs a complementary package in the EA management suite that includes tools and portal that uses repository. For this implementation case, besides input from repository team, requirements were extracted from other exploration. This exploration is conducted by the author of current research regarding the functions that BiZZdesign's current InSite portal has (A) and a wiki application of an IT reference architecture of an external party (B). The following requirements were then gathered:

[RI1] The wiki system shall be extensible. This means that the wiki software shall have or enable the use of available extensions for additional functionality. (A)

[RI2] The visualization shall be near real-time. A repository stores the most recent model packages and its contents. Latest stored content in the repository will be visualized. (B)

[RI3] The visualization shall accommodate model package's contents. This includes all objects in the model package. (A, B)

[RI4] The visualization shall provide a layout of the objects, information & image representation in and of them. The wiki software shall enable layout arrangement. (A, B)

[RI5] The visualization shall be able to search the contents of all existing model packages in repository. This is needed particularly when repository consists of many model packages with probably similar names in their contents. (A)

[RI6] The visualization shall provide navigations for repository browsing and for search function. (A, B)

4.3.2 System and software design

4.3.2.1 Software architecture

Components in software architecture are assignments of responsibilities in subdivision of an application. Therefore, a certain role is played by each component and series of component work together to supply the needed functionality (Gorton, 2011).

Architecture of prototype (see Figure 4-4) consists of extractor of enterprise knowledge (the repository system), application components using the service (model framework, enterprise modeling component, a representation engine to generate image representation), and the elements built using MediaWiki. The last is realized by three application services: Initializer, Visualizer, and Search Bridge.

In this architecture, repository system, infrastructure and visualization tool application components from application layer of reference architecture in chapter 3 are heavily represented. Here, Enterprise Modeler and data objects constitute the repository system. Enterprise Modeler is supported by application components of Model Framework, Representation Engine and Enterprise Modeling Component.

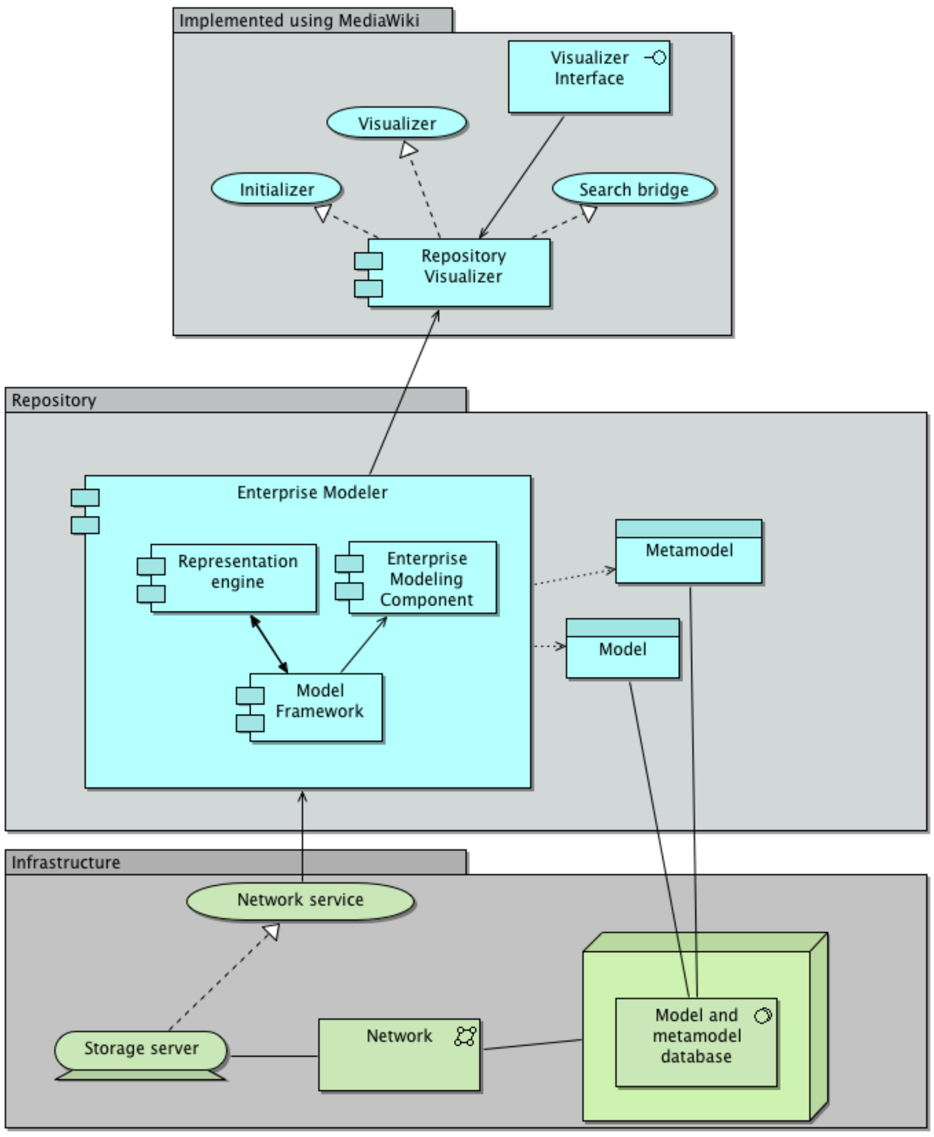


Figure 4-4 Architecture of the prototype

Model Framework has the responsibility of controlling lifecycle of derived objects. Besides this main responsibility, it also has generic tracing and error control functionality. Visualization tool can ignore whether a model is consistent or not, because it is completely taken care of by Model Framework (BiZZdesign, 2011). Enterprise Modeling Component is an element that is used by BiZZdesign tools (see section 2.7) to provide handling functionalities of business process models and architecture models (BiZZdesign, 2011).

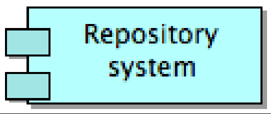
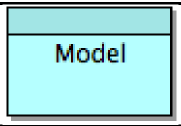
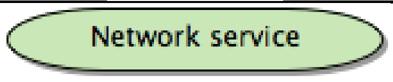
The two data objects are accessed by Enterprise Modeler for the necessity of delivering right representations for Repository Visualizer. A Model is an instance of Metamodel. Properties within Model are used for generating relations of objects inside a model and extracting the documentation texts. Thus, Model data object is the one accessed most frequently in this system. Entities of model flow from Model and metamodel database that access Storage server through Network. Subsequently, those models and their properties are accessed by Enterprise Modeler, which is a component that corresponds with Repository System in the reference architecture. Enterprise Modeling Component give meanings to those properties and together with Model Framework and Representation engine provide elements of extraction of model properties that in turn will be captured by Repository Visualizer.

In section 2.5, we discussed the importance of knowledge units for complicated knowledge structure. In business process models and architecture models, these knowledge units correspond to properties of Model data objects. In this research, contents of repository are assumed to be valid.

Properties of Model that correspond to graphical representation accessed by Enterprise Modeler are then passed on to Representation Engine before being stored in a structured web address. Meanwhile, the tool for visualizing enterprise knowledge is depicted by the top gray box in Figure 4-4. Five elements inside the smaller gray box are the ones implemented using MediaWiki. As an application component, Repository Visualizer realizes three application services: Initializer, Visualizer, and Search Bridge.

4.3.2.2 Mapping software architecture to reference architecture

Repository Visualizer is the visualization tool in this prototype's architecture. It has necessary functions to provide visualization to models contained in BiZZdesign repository. Modified from reference architecture, Repository Visualizer in our prototype is realized with three application services, as mentioned in subsection 4.3.2.1.

Reference architecture layer	**Software architecture element**	**Reference architecture element**
Application	Repository Visualizer	Visualization tool
Application	Enterprise Modeler	
Application	Metamodel and Model	
Technology	Network service	

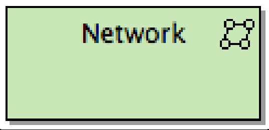
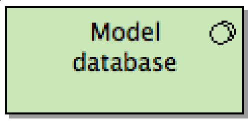
Technology	Network	
Technology	Model and metamodel database	

Table 4-1 Mapping of prototype architecture to reference architecture

At BiZZdesign, Enterprise Modeler is the instantiation of Repository system. It is composed of three components that are essential for extracting the properties of data objects. Still in application layer, metamodel and model are accessed to obtain properties of models. Metamodel data object is a necessity because the way a model stored in the BiZZdesign repository includes different metamodels. This will be explained later in subsection 4.3.2.3.

A database containing model and metamodel holds contents that are associated with metamodel and model data objects. In practice, database can vary for different types of those that implements SQL. This database is associated to Network and subsequently Storage server. Those two elements are that of what the reference architecture depicts, whereas a network service complements prototype's infrastructure and used by Enterprise Modeler to connect with data and infrastructure.

4.3.2.3 Data model

The repository stores model as depicted in Figure 4-5.

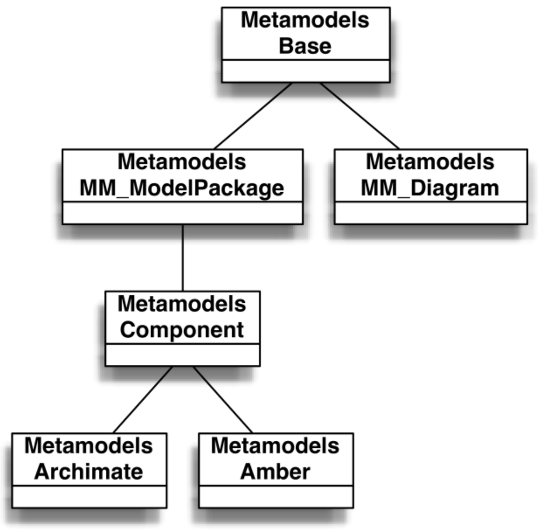


Figure 4-5 Data model of BiZZdesign repository

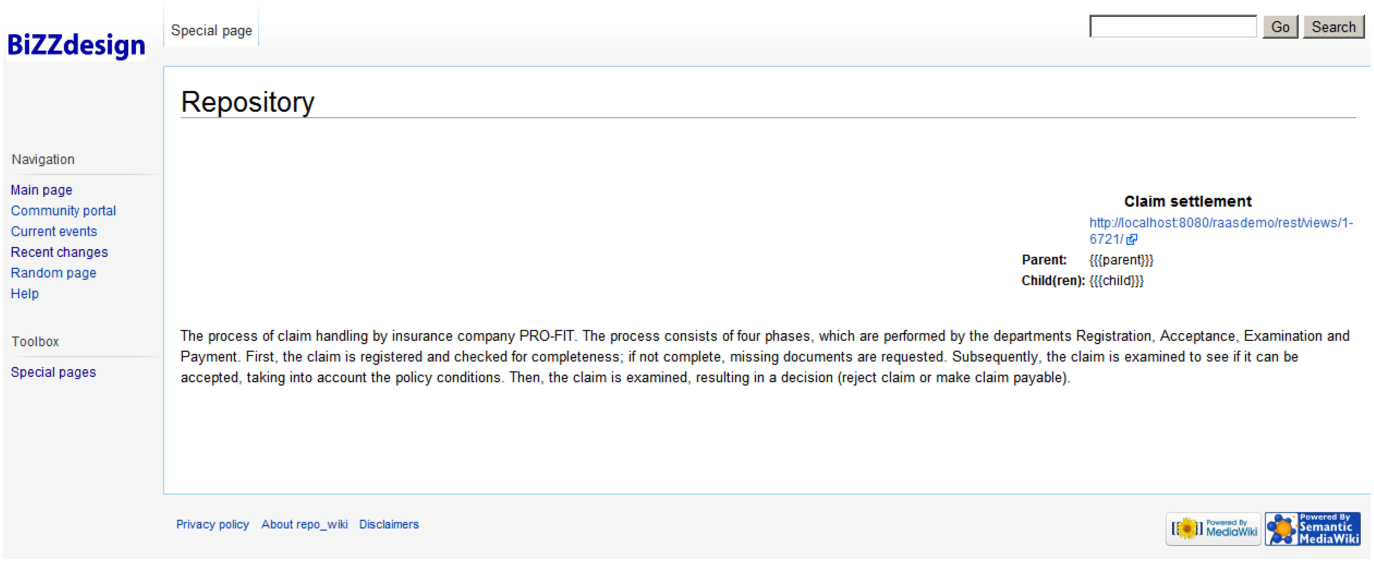
From Base metamodels, the repository stores two metamodels: MM_ModelPackage and

MM_Diagram. Generally, metamodel is instantiated with models, with properties containing that inside metamodels. A ModelPackage has a collection of Models. Among them are the ones containing components. This has two kinds of Metamodel inside, each for their own language structure. Metamodels Amber is mainly used by BiZZdesigner and Metamodels ArchiMate is mainly used by the tool Architect (BiZZdesign, 2011).

4.3.2.4 Interface design

An interface design to show how MediaWiki can fulfill [RI1] to [RI5] and part of [RI6] (navigations excluded) can be seen in Figure 4-6. The interface design shows layout that is similar to a layout of a component from ArchiXL's wiki discussed in subsection 2.4.1. Our prototype is designed to provide placement for included objects and linking of them, something that MediaWiki can provide with the use of special page calling and external image embedding.

Placement of objects is presented with a borderless table on top right part of Figure 4-6. The design shows brackets that are part of MediaWiki's syntax for linking/calling contents or other pages. Furthermore, above the object linking is a place for image representation of business process model or architecture model. Arrangement designed in this phase is such that placement of borderless table for image representation and linking of objects won't collide with the space allocated for documentation below them.



Repository

Claim settlement

http://localhost:8080/raasdemo/rest/views/1-6721/

Parent:	{{{parent}}}
Child(ren):	{{{child}}}

The process of claim handling by insurance company PRO-FIT. The process consists of four phases, which are performed by the departments Registration, Acceptance, Examination and Payment. First, the claim is registered and checked for completeness; if not complete, missing documents are requested. Subsequently, the claim is examined to see if it can be accepted, taking into account the policy conditions. Then, the claim is examined, resulting in a decision (reject claim or make claim payable).

Figure 4-6 Interface design

4.3.3 Implementation and unit testing

At the time of development, the most recent MediaWiki version is 1.17.0. This is subsequently the version used in implementation. MediaWiki is supported by 1) PHP and 2) database server that implements SQL. The language used in developing the prototype is PHP.

4.3.3.1 Main components

The components below are the main components used to visualize contents of BiZZdesign's repository. They are depicted in the prototype architecture as application services that Repository Visualizer realizes. These were made from scratch using the concepts of special pages in MediaWiki. The components are built with PHP scripting language and MediaWiki syntax.

The first is intializer, which lists model packages present (most recently saved) in repository. These model packages are sourced from models made with design tools or exported from a recognized format. The second one, responsible for visualizing enterprise knowledge to wiki pages in MediaWiki, is Visualizer. Visualizing includes handling of layout of pages, handling of possible empty content, and specifying image boundaries. The third main component is Search Bridge. This component handles the search input and function to repository search. Search function can facilitate one of the AND or OR operator, and can also be combined with wildcard character (*). This wildcard character can be placed at the front of, between letters, or as the last character of a search term and a combination of two or all of three placements. Wildcard character can also be used without the AND or OR operators.

Each of the main components will be addressed and will be described briefly.

Initializer

This component is the one called first when a user calls for the repository address. What this component does it that it sets base address generated by repository system with relevant address of the deployment. It then sets containers of model names, references for navigation, and identifiers for objects and its sub-objects that are located one level below by loading the XML input to a structured container using simplexml functions. The component then checks if container of model name(s) is empty before visualizing it to a MediaWiki page. In case the checking showed it to be not empty, a bullet listing the name and a link for each model name is made.

Visualizer

All model packages and objects in the repository are handled by this component for each of their page generation in MediaWiki. It initially sets base address of repository system and its appending values for addressing. Then it sets containers required for model names, identifiers, and references for navigation, and also containers of contained objects and/or views names, references for navigation, and identifiers.

After those containers are set, Visualizer loads the XML input to a structured container with the help of simplexml. It will then handle layout cases, including empty documentation, containment of views and objects in a page, containment of views only in a page, containment of objects only in a page, while handling objects links and identifiers, including that of object relations. Moreover, Visualizer also handles layout robustness for zooming in browsers, handles navigation to each root modelpackage, and handles display from a search query link.

Search Bridge

An action of clicking the search button will trigger this component. It will first set base address and its appending values for addressing. Subsequently, it then handles search query input, including operators that users might add before, between or after alphanumeric value. When server-side searching process is done, it stores matched results in a container. After generating MediaWiki page for displaying search results, it then prints a search message after checking the number of search results gathered. After the search message/notifications are printed, search results are also printed, given the container is not empty. This component also automatically assigns links to the objects of search results. Those links will call Visualizer component to display pages of corresponding objects. In case of empty search results, a notification that there are no objects or modelpackages inside the repository will be shown.

4.3.3.2 Supporting components

The remaining components listed below are 1) obligatory extensions required for the coupling to work and 2) additional extensions to be used for future development. They are:

1. simplexml

   This is a PHP extension that provides a toolset to convert XML to an object that can be processed with normal property selectors and array iterators.
2. libxml

   A PHP extension that serves as a preliminary for simplexml in this implementation
3. ExternalData

   A MediaWiki extension able to handle different types of knowledge format as an input to be displayed in a MediaWiki page

Display basically differs with each main component. On Initializer, the system displays the model package names as a bulleted list. Those names will then be clickable and will lead to displays handled by Visualizer component. On any page, users can search repository contents by entering a search term on the form to the left of the wiki page’s left border. That search term will then be handled by Search Bridge to generate a page containing results of the search.

Current display style is based on Vector theme embedded in MediaWiki 1.17.0. It has been modified with layout on the extensions used for visualizing. This layout is based on placement position on page and sequence of objects’ information and extracted properties. The form for repository search input is placed within PHP file of Vector theme (\skins\Vector.php), therefore same placement needs to be done on other skin(s) if a display change is needed.

4.3.3.3 Modification steps from a MediaWiki installation

MediaWiki installation in this development is configured with default option with no regard to database type preference, an issue that many MediaWiki installations use to address.

Installation of obligatory extensions that are libxml (PHP extension), simplexml (PHP extension), and ExternalData (MediaWiki extension) is done by modifying installation of installed package. The first two (libxml and simplexml) are placed within 'ext' folder of PHP package in the server. To enable them (if not already enabled in a default PHP installation), modification instructions on http://www.php.net/manual/en/libxml.setup.php and also http://www.php.net/manual/en/simplexml.setup.php shall be followed.

To be noted when choosing the PHP package is the required minimum version of PHP (5.2.3) for MediaWiki 1.17.0. The third one (ExternalData) and remaining supporting extensions are placed inside 'extension' directory in MediaWiki. Installing a MediaWiki extension is done through modifying the file 'LocalSettings.php' located inside the root folder of MediaWiki installation. This file modification is basically done to tell MediaWiki to include extension that is located in a certain location, using include_once or require_once PHP functions.

Creation of three main components that are Initializer, Visualizer, and Search Bridge, all inside the folder 'extension' of MediaWiki is done in four steps: creating the files needed as per the instructions on Special pages manual of MediaWiki, setting up relevant parameters from the guideline above, modifying the file LocalSettings.php to include three main components, and filling required code to the file 'Special[extension name].php'.

Overall, from a server package or PHP installation the system needs to be modified based on the following series of steps. Two PHP extensions, libxml and simplexml, will need to be enabled initially. After that, the three main components and ExternalData supporting component can be copied to 'extension' folder of MediaWiki. This will need to be followed with setting of the parameters for those components on MediaWiki's local setting. Accordingly, the base address of the repository system's XML pages can be added inside the code of both Initializer and Visualizer components. After that, prototype is ready to be tested.

4.3.3.4 Discussion on implementation

Due to limitation of extraction of enterprise knowledge in models to the form of XML, only a few pages have complete documentation and/or object nesting information. The limitation is that of repository system which is still in development. However, in next phase (testing), relevant pages can still be shown to test that system development abides the requirements defined.

Other boundaries are also defined. It is advised to see wiki pages in font ranging from 11-12 using standard Arial font size. Next, documentation of an object that can be handled by prototype is a maximum of 4979 characters. On XML pages generated by enterprise modeler, a “View” has to have its name ending with the word 'view' (case-insensitive) when stored in repository in order to be able to be classified in the object listing in a wiki page generated by the repository. This prototype is also developed to have a certain name length boundary. It can handle object names up to 960 characters, which is also the maximum number of characters when searching the repository.

A restriction is also made on a wiki page layout. First, image width allocated is 4.625 inches. Beyond that measurement, a scroll bar will appear to assist viewing the remaining part of the image. Second, documentation space width of 6.16 inches is allocated. These allocations requires browser to take full width (occupies the full screen) to optimize viewing of wiki pages.

Components (main components and supporting components) in this implementation stand for the unit of this prototype’s implementation. Considering the fact that layout of a wiki page includes collaboration of those components, unit testing part in this implementation phase is done along with integration and system testing.

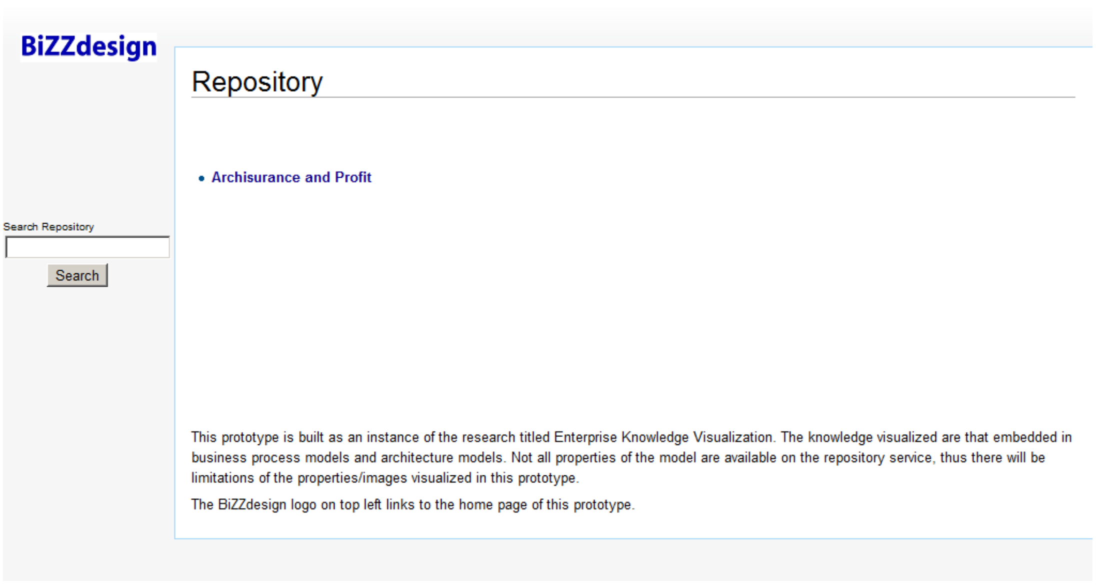
In subsection 3.2.1, we discussed requirements for a repository of an end system basing its implementation from our repository. Repository exploration especially on Model data object and Enterprise Modeling Component during implementation proves that BiZZdesign repository fulfills RR2, RR6, and RR7. Both Model and Metamodel data object also shows the functionality to support metadata information addition (RR5). Capability to use different databases and its knowledge storage proves that BiZZdesign’s repository fulfills RR8 and RR12. Overall, this implementation alone shows that the repository covers six of the thirteen requirements.

4.3.4 Integration and system testing

This phase is conducted to verify result of development phase. We conduct the testing with examining pages utilizing the system’s main components. Figure 4-7 shows a testing of Initializer component. The only model package present in the repository is listed and given a link, two of the component’s main purpose. Continuing to Visualizer component, Figures 4-8 through 4-10 shows the placement, layout and link assigning. Horizontal object relation, and consequently the implementation of object relation management external application service, is evaluated with wiki pages in Figure 4-8 and Figure 4-9, with ‘Claim settlement’ being an object contained in the ‘profit’ model package.

The third main component, Search Bridge, is tested with visualization of its result depicted in Figure 4-11 and Figure 4-12. The figure shows that search results had been successfully given links, in addition to the facilitation of search using wildcard operator, as indicated by the sentence above "Search Results:" on both search execution.

Different from design phase, development phase resulted in a change of placement of search box, from top right of a MediaWiki page to middle left of the page. Unneeded native MediaWiki links are also hidden, so that current system can have a cleaner interface.



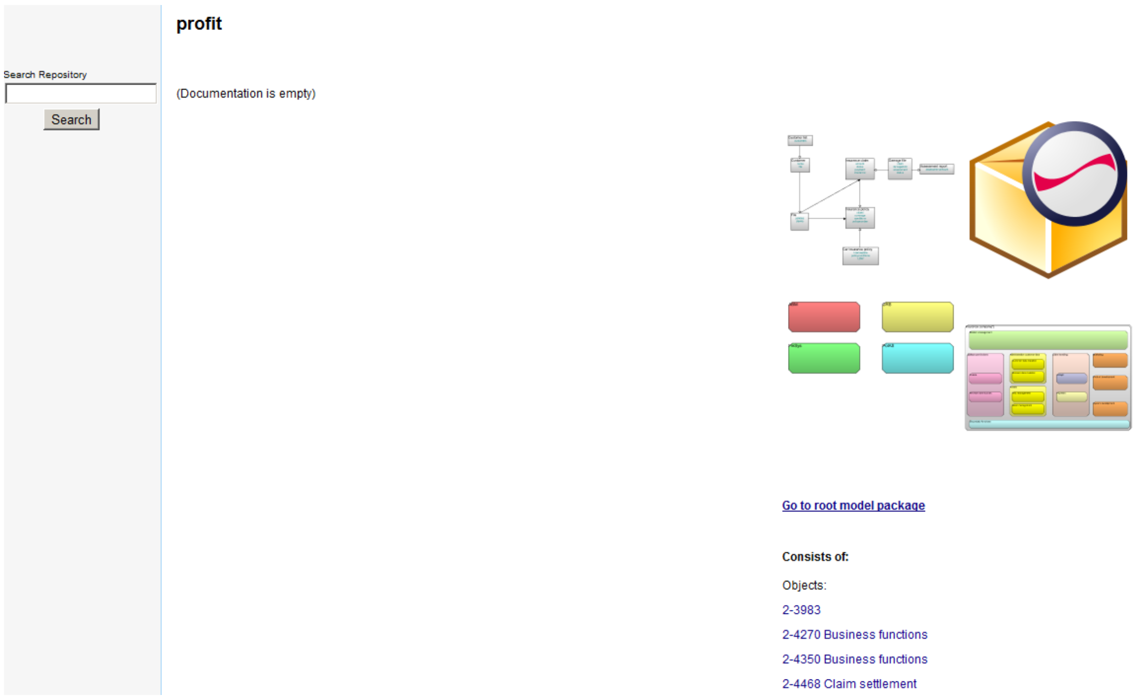
Repository

- Archisurance and Profit

This prototype is built as an instance of the research titled Enterprise Knowledge Visualization. The knowledge visualized are that embedded in business process models and architecture models. Not all properties of the model are available on the repository service, thus there will be limitations of the properties/images visualized in this prototype.

The BiZZdesign logo on top left links to the home page of this prototype.

Figure 4-7 Testing of initial page of enterprise knowledge visualization using repository (Initializer component)



profit

Search Repository

Search

(Documentation is empty)

Go to root model package

Consists of:

Objects:

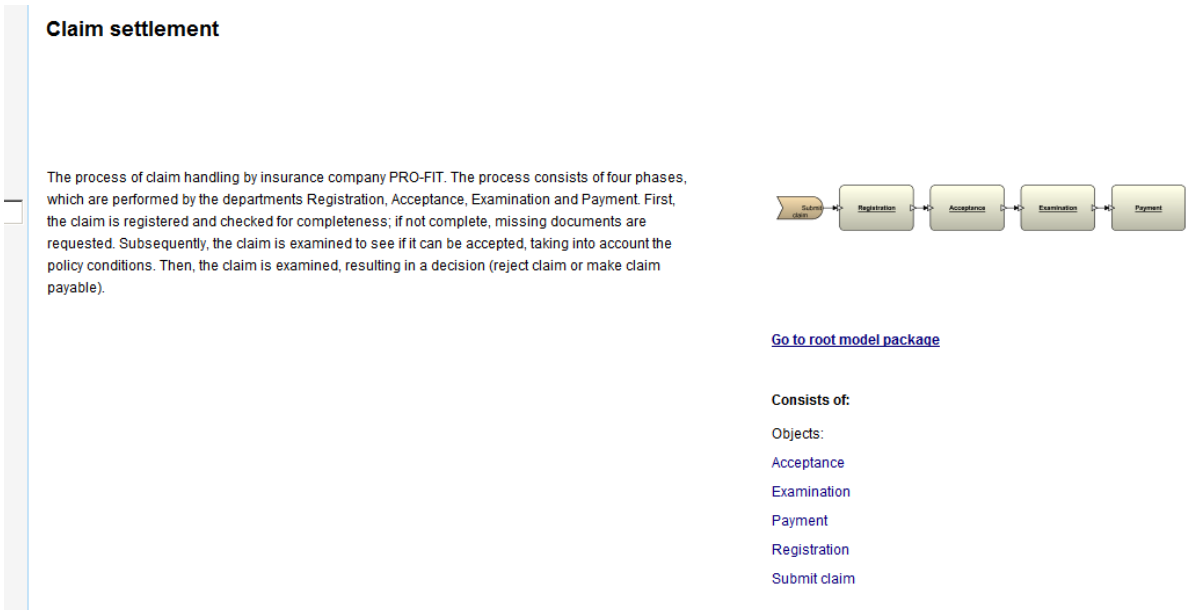
2-3983

2-4270 Business functions

2-4350 Business functions

2-4468 Claim settlement

Figure 4-8 Testing visualization of 'profit' model package



Claim settlement

The process of claim handling by insurance company PRO-FIT. The process consists of four phases, which are performed by the departments Registration, Acceptance, Examination and Payment. First, the claim is registered and checked for completeness; if not complete, missing documents are requested. Subsequently, the claim is examined to see if it can be accepted, taking into account the policy conditions. Then, the claim is examined, resulting in a decision (reject claim or make claim payable).

Go to root model package

Consists of:

Objects:

Acceptance

Examination

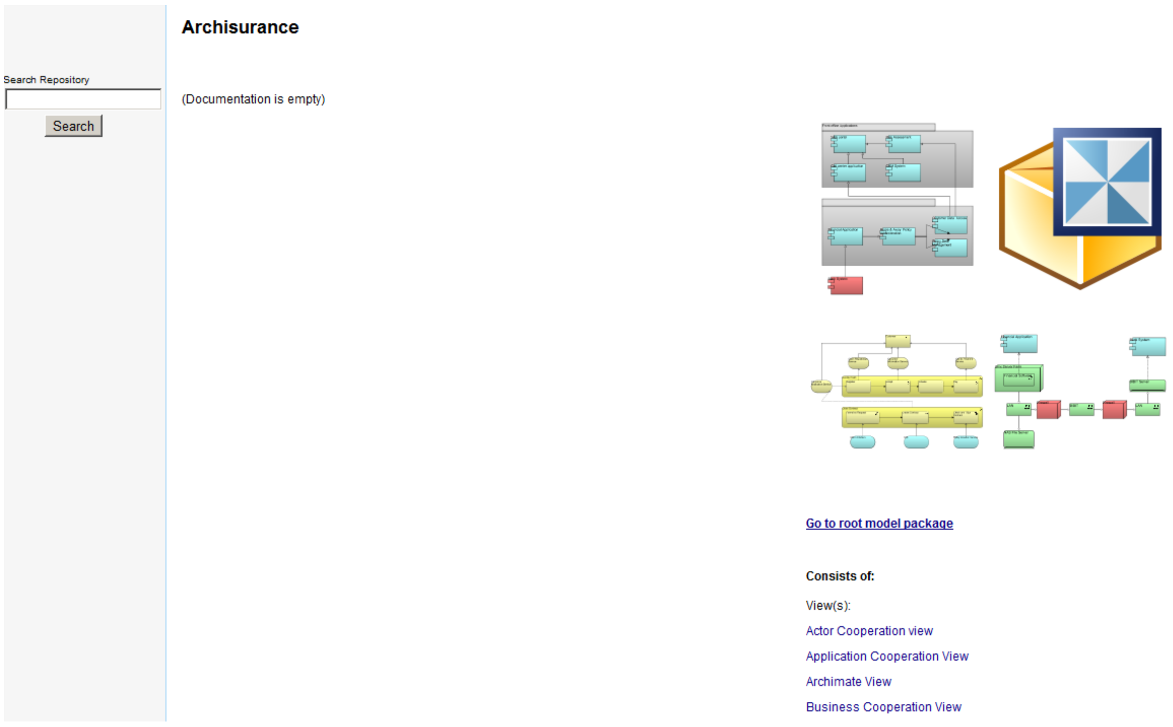
Payment

Registration

Submit claim

Figure 4-9 Testing visualization of 'claim settlement', inside 'profit' model package

BiZZdesign


Search Repository

Search

Archisurance

(Documentation is empty)

Go to root model package

Consists of:

View(s):

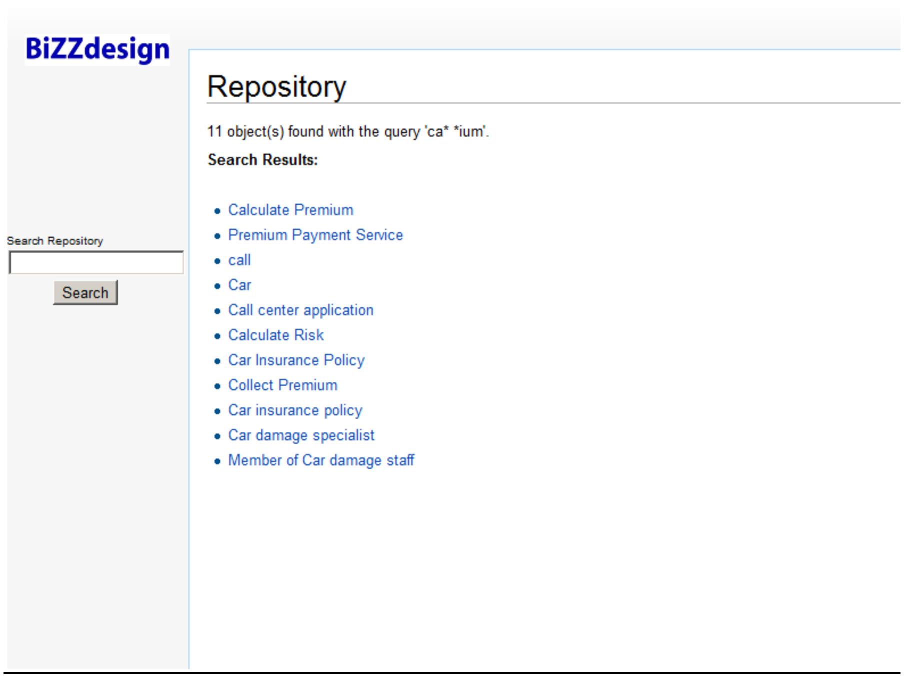
Actor Cooperation view

Application Cooperation View

Archimate View

Business Cooperation View

Figure 4-10 Testing visualization of 'Archisurance' model package


BiZZdesign

Search Repository

Search

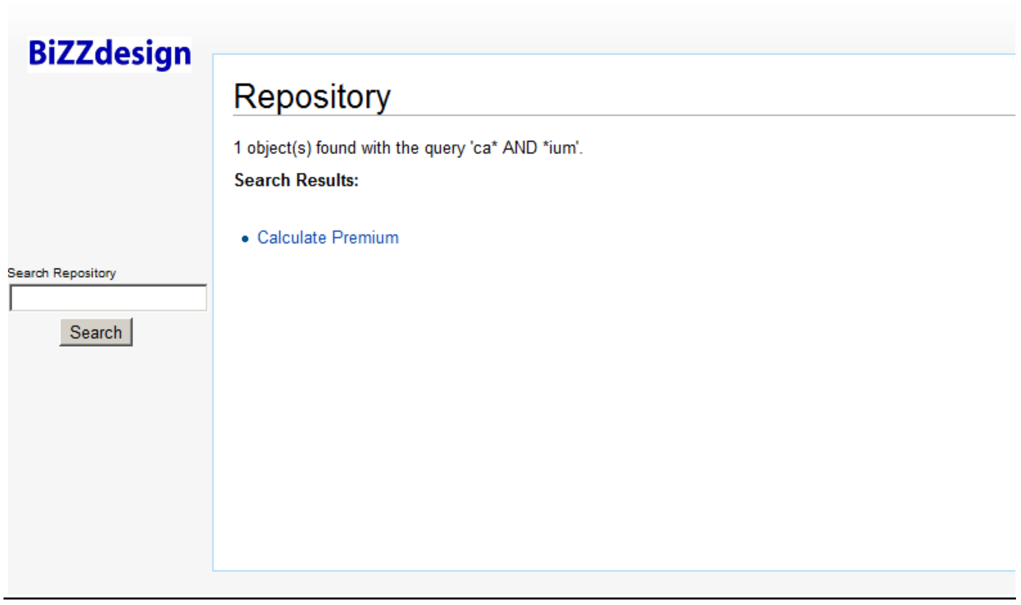
Repository

11 object(s) found with the query 'ca* *ium'.

Search Results:

- Calculate Premium
- Premium Payment Service
- call
- Car
- Call center application
- Calculate Risk
- Car Insurance Policy
- Collect Premium
- Car insurance policy
- Car damage specialist
- Member of Car damage staff

Figure 4-11 Testing of search function of the prototype with search query that includes wildcard character



Repository

1 object(s) found with the query 'ca* AND *ium'.

Search Results:

- Calculate Premium

Figure 4-12 Testing of search function of the prototype with search query that includes both wildcard and operator

A remote testing was done in Amersfoort and proven to have no significant delay in loading objects and images. The prototype has been tested in laptop computers with 13 inches screen size.

Images captured in this testing phase were obtained on a laptop computer with following environment: Microsoft Windows 7 operating system, WAMP server package, and version 12 of Mozilla Firefox web browser. This prototype was also tested to visualize models with same output using version 14 of Google Chrome web browser. Both results are obtained using 13 inches screen size laptop computers with 1280 by 800 pixels screen resolution.

4.3.5 Operation and maintenance

Deployment is conducted on a server at BiZZdesign Enschede. This server has relevant server package containing PHP and SQL package installed to facilitate deployment of prototype. This server is connected with repository system through an internal network. After deployment, there are no change requests being made in this software development. This is due to time limitation of this research and the development stage of BiZZdesign's repository.

4.4 Impact to BiZZdesign's situation

This section includes the possibilities enabled through implementation of this prototype, how BiZZdesign's repository system can be improved, in terms of generation of enterprise knowledge representation and handling and providing certain functionalities.

Description in early part of section 3.1 is specified in this chapter through narrowing enterprise knowledge embedding format to business process models and architecture models. Properties of those models are transformed into contents of XML tags, and visualized with the help of MediaWiki and its extensions.

4.4.1 Features of the prototype

From requirements in subsection 4.3.1 and through subsequent phases afterwards, our prototype has a set of features as Table 4-2 included:

No	**Features**	**Requirements addressed**
1	Near real-time coupling with BiZZdesign repository	[RI2]
2	Listing the model packages present in the repository	[RI3]
3	Providing links of model packages existing in the repository	[RI1] [RI3]
4	Going through each model package's contents (all the objects included in the model package)	[RI3]
5	Projecting image representation of the objects in the repository	[RI1] [RI3]
6	Extracting the documentation included in each object	[RI1] [RI3]
7	Indicating absence of documentation in a visualization	
8	Providing links of objects/views contained in a model package or object	[RI3]
9	Providing a layout of the object information, image representation, objects/views contained, and documentation in form of a MediaWiki page	[RI4]
10	Indicating absence of objects in a containment	
11	Filtering out the objects contained in the repository that has no name	
12	Separating views from objects in a model package's or an object's containing views and other objects	[RI4]
13	Searching the contents of all existing model packages in the repository	[RI3]
14	Accommodating operators from repository search functionality	[RI5]

15	Providing message of invalid search and/or empty search result	
16	Listing and providing the links to the items of the search results	[RI4]
17	Providing simple navigations to the repository browsing and search function	[RI5] [RI6]

Table 4-2 Features of the prototype

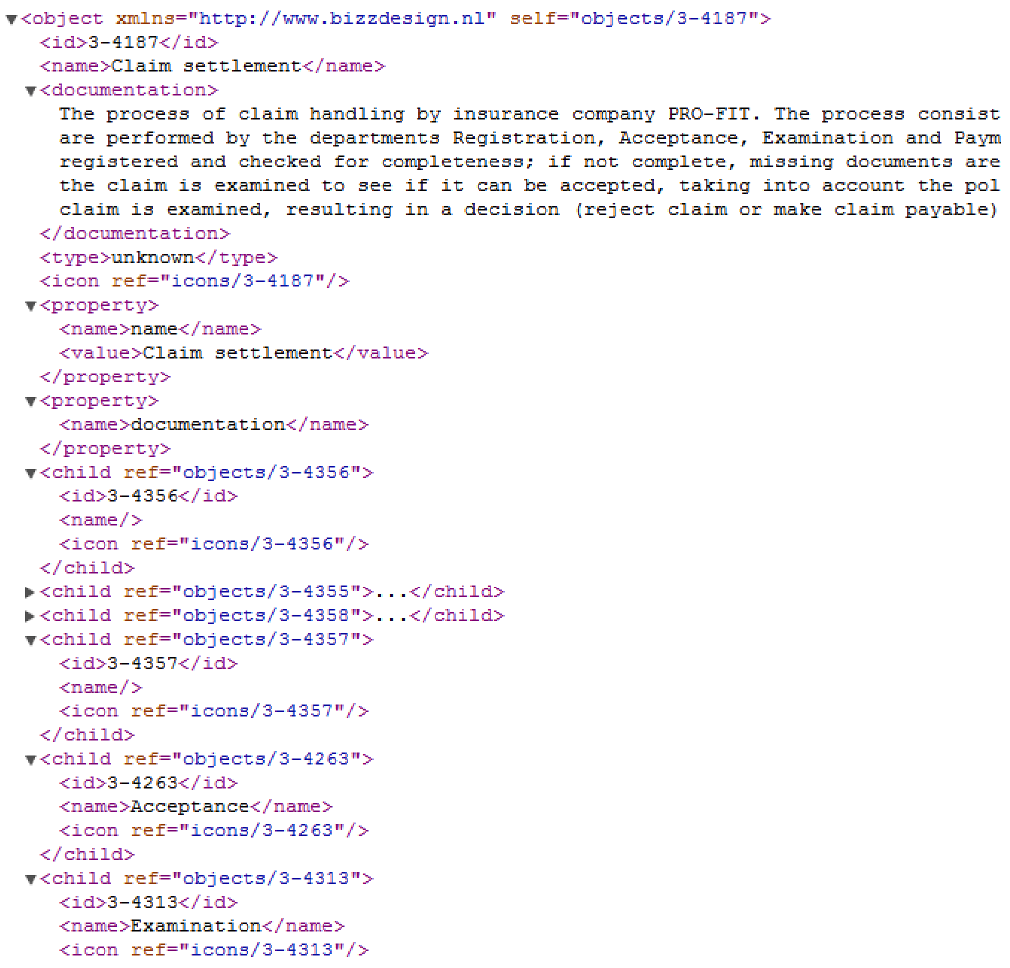
Extra information in square brackets after each feature indicates addressed requirement(s) of that feature (see subsection 4.2.1.2). In listing a content of a model package and object and providing links for them (feature 4 and feature 8), the system displays a relation inside the same space for objects in a MediaWiki page layout. This is to show that current repository needs to define a difference of an object and a relation, which is a limitation of current repository. Some features that do not come from list of requirements are provided to make the prototype more user-friendly.

Current repository system has a service (see subsection 2.7.2) with which XML pages are extracted. A sample XML page can be seen in Figure 4-13. Using attributes/tags inside each object, coupling and linking to the MediaWiki, as evidenced with this prototype implementation, is made possible. Additional information and functions from the repository needed for its further development are summarized in subsection 4.4.2. Extra information and services/functions were derived from repository's current state.

4.4.2 Additional information

More of attributes/tags in similar function would be useful for providing more functions that will improve visualization of models inside BiZZdesign's repository. Those additional attributes/tags in a generated XML, i.e. additional information stored from the models, can provide reference to an object's root model package in form of XML tag, to be used in search (search within a model package) and search results. An object will then have a <rootmdlpkg> XML tag in its XML extraction, for example. This will provide information of to which model package an object belongs to, which will also be useful to group search results.

Different resolutions of image representation to be used relevantly in displaying a model package or an object of single component/concept can also be provided. A component with image can sometimes need a larger resolution if it is a model package or a sequence of process. To make sure it can be seen clearly per object, a set of images with relevant resolution will be needed.



```
▼<object xmlns="http://www.bizzdesign.nl" self="objects/3-4187">
   <id>3-4187</id>
   <name>Claim settlement</name>
 ▼<documentation>
     The process of claim handling by insurance company PRO-FIT. The process consist
     are performed by the departments Registration, Acceptance, Examination and Paym
     registered and checked for completeness; if not complete, missing documents are
     the claim is examined to see if it can be accepted, taking into account the pol
     claim is examined, resulting in a decision (reject claim or make claim payable)
   </documentation>
   <type>unknown</type>
   <icon ref="icons/3-4187"/>
 ▼<property>
     <name>name</name>
     <value>Claim settlement</value>
   </property>
 ▼<property>
     <name>documentation</name>
   </property>
 ▼<child ref="objects/3-4356">
     <id>3-4356</id>
     <name/>
     <icon ref="icons/3-4356"/>
   </child>
 ▶<child ref="objects/3-4355">...</child>
 ▶<child ref="objects/3-4358">...</child>
 ▼<child ref="objects/3-4357">
     <id>3-4357</id>
     <name/>
     <icon ref="icons/3-4357"/>
   </child>
 ▼<child ref="objects/3-4263">
     <id>3-4263</id>
     <name>Acceptance</name>
     <icon ref="icons/3-4263"/>
   </child>
 ▼<child ref="objects/3-4313">
     <id>3-4313</id>
     <name>Examination</name>
     <icon ref="icons/3-4313"/>
```

Figure 4-13 An XML page generated by repository system

For previewing purposes, repository system can add thumbnails in separate resource address, in purpose to be used with a MediaWiki extension. This extension can create a hyperlink within wiki text to a template (or to a normal wiki article) with parameter passing. In search results, this can be useful in case there are still objects with similar name, even though they are already grouped by their root model package name (by reference to an object's root model package). Another information to be added is object information of "appears in" stored in form of XML tag. This can be useful in knowing in which other objects/views within another place (related view/sequence) an object is also part of.

4.4.3 Additional functions

Apart from additional attributes/tags, repository system should provide additional functions that will enable to search using an object type identifier. Search-input handler can be modified to handle, for example, additional words as identifier. Those additional words will correspond to the "type" tag in an XML page (currently still has the same value of "unknown" for all objects, see Figure 4-13). This should also be accommodated with object type support in repository's search function. Another function is image generation as an object with numeric values and identifiers. Those values will then be used as ‘coordinates’ of the object(s) inside, in order to make it linkable. In turn, those ‘coordinates’ will be able to be assigned a link using another MediaWiki extension.

Separation of words will also need to be added. This function will separate words inside documentations that are, for example, not prepositions and are not conjunctions from documentation to be used later with another MediaWiki extension. Those free-of-prepositions, free-of-conjunctions documentation words will then be parsed by a parser function to provide links. This parser function can be made with PHP, or can be embedded in repository as an extra function. Documentation with links will contain back the previously omitted prepositions and conjunctions. This linking feature in documentation can be optional for a user, thus a toggle to turn it on or off might be useful, in particular for the performance of near real-time coupling. In a bigger scope, this function might be embedded in repository or its system as a more general word-filter function.

Last is a function capable of replacing of the content of the documentation tag in objects/model packages that will go hand-in-hand and will still enable word separation/filtering. This will be used to handle editing of the documentation. The editing itself can be done with a temporary container of edited documentation.

4.5 Summary

Previous three sections comprise steps of implementation of prototype, concentrating on parts of reference architecture from chapter 3. MediaWiki as the chosen wiki software product, supported with technologies and application components at BiZZdesign, proved that it is more than adequate to facilitate presentation of enterprise knowledge. The implementation, carried out with waterfall software development lifecycle, also show that parts of reference architecture designed in chapter 3 are sufficient to support enterprise knowledge visualization.

5. Validation

In IS field, design-science research should result in IT artifact created to address an important difficulty within an organization (Hevner et al., 2004). The first part of this chapter describes how validation will be conducted. The second part shows the validation of system prototype built based on parts of the reference architecture.

5.1 Validation approach

The prototype implemented as described on chapter 4 might or might not be valid to the users of the tools. Therefore, we conduct a survey to validate system prototype based on aspects of visualization. Aspects that are used for validating the prototype are generated from three properties of design artifact described in design-science research guideline (Guideline 3 in Table 5-1) from Hevner et al.

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 5-1 Design-science research guidelines (Hevner et al., 2004)

We consider guideline 1, 2, and 4 to 7 to be not applicable to current validation/design approach because time limitation of this research. Thus, we refer only to guideline 3. Efficacy of prototype can be measured by its functionality; thus we refer to the functional requirements in subsection 4.2.1.2. Key areas mentioned by Gallagher et al. (2008) on visualization are used as a base for validation questions.

In order to measure functionality, we believe our prototype needs to be valid for how it will facilitate its prospective users. For this, Gallagher et al. (2008) include four sub-aspects: static representation, dynamic representation, views, and navigation and interaction. In this research, static representation and dynamic representation sub-aspects are merged because there are no dynamic representations. This sub-aspect is measured with how image formats are visualized as seen by respondents of survey. Next, questions 3 through 5 will measure how our prototype facilitates views and how users can navigate and interact with prototype easily. Utility/usefulness is measured with how our prototype makes the user complete the task better or how it eases their job and how relevant the platform is. For this sub-aspect, how it enhances user's job performance will be measured with three questions related to transfer from repository, viewing documentation, and understanding the models. They can measure user's performance because the more a visualization system provides complete properties of models, the more a user can understand them.

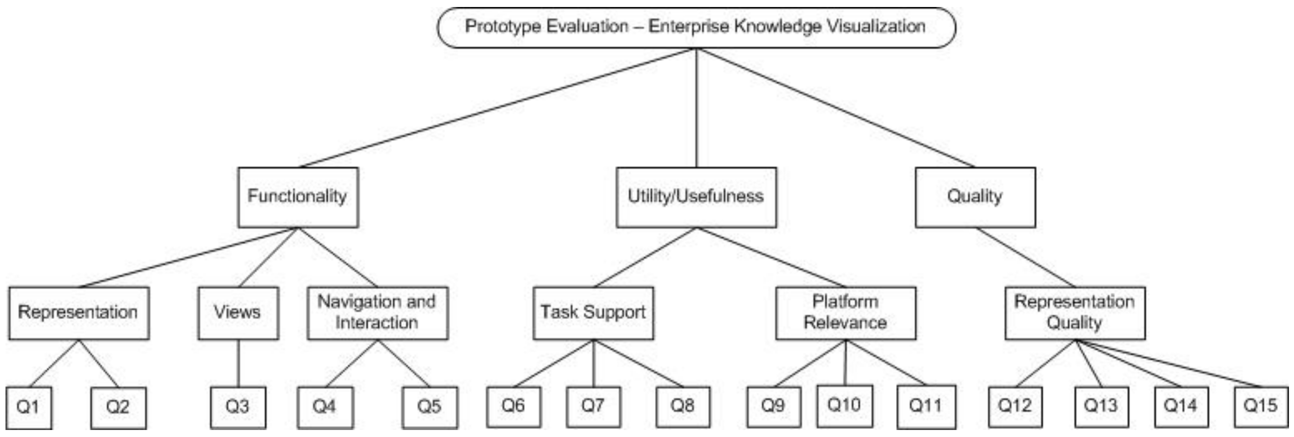


Figure 5-1 Map of EK visualization prototype's validation question

Platform relevance is the next sub-aspect on usefulness. This sub-aspect has three questions that measure whether visualization system is relevant to the platform and components of MediaWiki. As the wiki software chosen for our prototype, MediaWiki's extensibility means that it has a growing number of components/extensions. Question related to this measures whether the respondent feels that relevant extensions of MediaWiki has been used. Last validation sub-aspect is representation quality. Questions 12 and 13 measures quality of generated representation with the factors mentioned by Moody (2009):

color, shape and size. Subsequently, layout and the usage of technology that supports generation of models and their representation are measured using last two questions (Q14 and Q15).

Q1	The visualization system supports both business process models and architecture models.
Q2	The visualization system supports image formats of business process models and architecture models.
Q3	The visualization system accommodates architectural views in a proper way.
Q4	The visualization system can be used to search for objects in architecture models or business process models.
Q5	The visualization system enables navigation between wiki pages.
Q6	The visualization system helps me in understanding architecture models and business process models.
Q7	The visualization system helps in viewing the documentation of models and objects.
Q8	The visualization system supports the transfer of models from repository to user.
Q9	The visualization system can be properly executed on a platform with a repository.
Q10	The visualization system, plus MediaWiki features, support viewing of models' properties (name, included objects, image, documentation) through connection with repository.
Q11	The visualization system has used relevant components from MediaWiki and repository to visualize enterprise knowledge.
Q12	The representation generated has proper use of colors.
Q13	The representation generated has proper use of object shape and size.
Q14	The layout of models and objects facilitate the understandability of models.
Q15	The technology that supports the generation of models and their representations are sufficient to generate high quality representation.

Table 5-2 Survey questions

Survey questions (see Table 5-2) are answered with following possible answers: Strongly agree – Agree – Neutral – Disagree – Strongly disagree. These questions have numeric value of 5 – 4 – 3 - 2 - 1, respectively. These questions will be addressed to people working in the company that this research is conducted at. Those respondents are considered to have sufficient capabilities on the topics addressed in this research. This includes business process models and architecture models that are the main embedding media of enterprise knowledge visualized by our prototype.

Except for four questions (Q9, Q10, Q11, and Q15), respondents are asked to take the perspective of an end user. The number of respondent is limited to eight people. Period of survey spans from June 25th, 2012 to July 16th, 2012. Answers to this survey are analyzed in section 5.2. In validating, we also consider the relation to implementation requirements in section 4.3.1. The second, fifth, and sixth requirements ([RI2], [RI5], [RI6]) relates to functionality. Utility/usefulness covers the first, fourth, and sixth ([RI1], [RI4], [RI6]) requirements. Meanwhile, representation quality relates to the fourth requirement ([RI4]).

5.2 Validation result

In this section, result of the survey conducted to validate our prototype is presented. As mentioned in previous section, there are three validation aspects: functionality (validation aspect 1), utility/usefulness (validation aspect 2), and quality (validation aspect 3). In Figure 5-3 to Figure 5-17, y-axis is the answer and x-axis is the number of respondent on that answer.

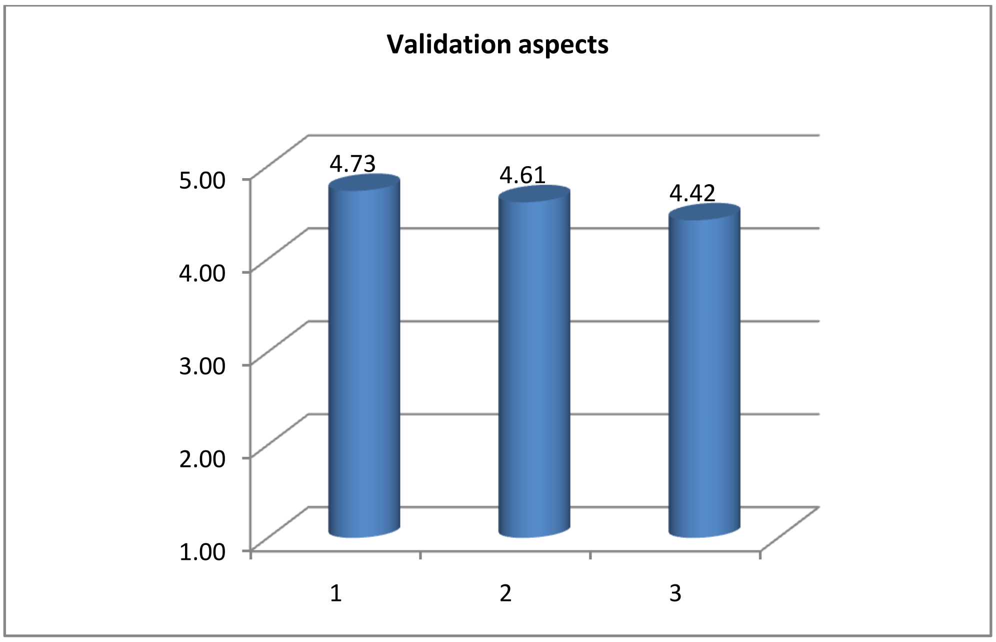


Figure 5-2 Score of validation aspects

The results in this chapter are average of the scores from respondents on each aspect, sub-aspect, and question. We will describe result of each aspect in following three subsections.

5.2.1 Functionality

This validation aspect covers three sub-aspects (representation, views, and navigation and interaction). Answers to questions covered by this aspect can be seen in Figure 5-3 to Figure 5-7.

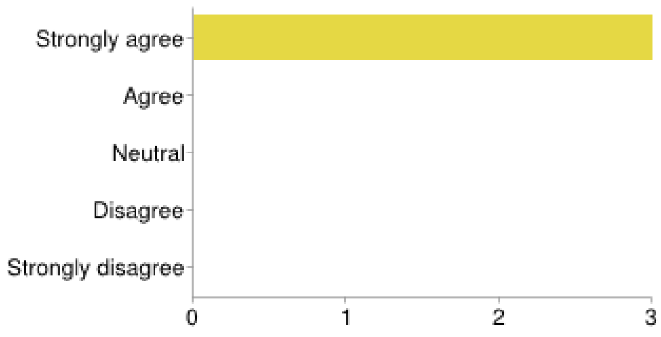


Figure 5-3 Answers to Q1

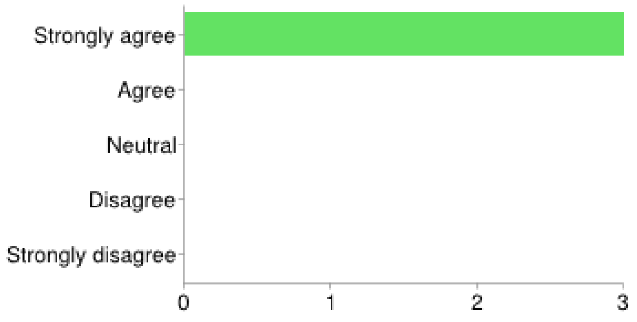


Figure 5-4 Answers to Q2

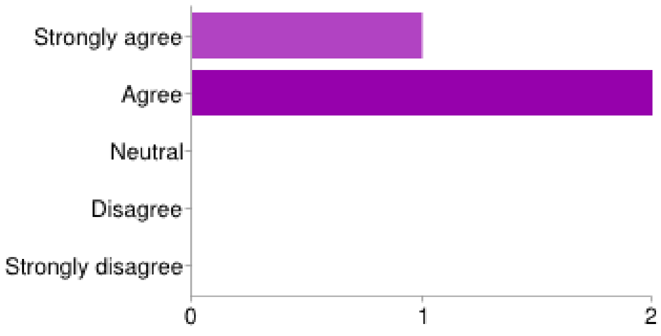


Figure 5-5 Answers to Q3

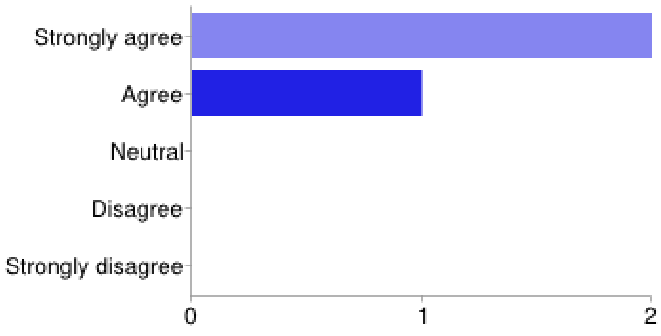


Figure 5-6 Answers to Q4

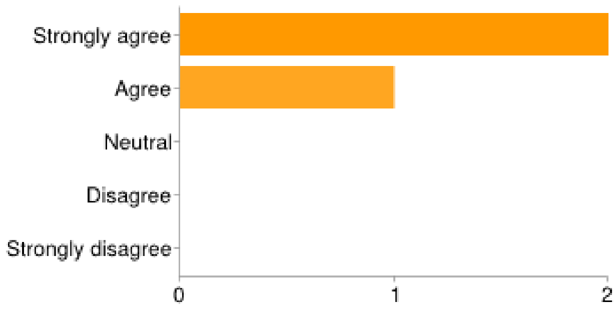


Figure 5-7 Answers to Q5

Average score of this answer is 4.73. Each of the sub-aspects results in high scores, especially Representation sub-aspect. In detail, average Representation score is 5, while Views is 4.33 and Navigation and Interaction is 4.67. This means that respondents think that functionality of the prototype satisfy their expectation. The score for this validation aspect is the highest among three validation aspects.

5.2.2 Utility/Usefulness

Detailed score for utility/usefulness validation aspect is 4.44 for Task Support and 4.78 for Platform Relevance. Those two scores compose the average score of 4.61 for utility/usefulness. The answers to six questions in this validation aspect can be seen in Figure 5-8 to 5-13. Figure 5-12 gives the most positive response with all three respondents selected 'Strongly agree' as their answer for Q10. The score for this validation aspect is the second highest of three validation aspects.

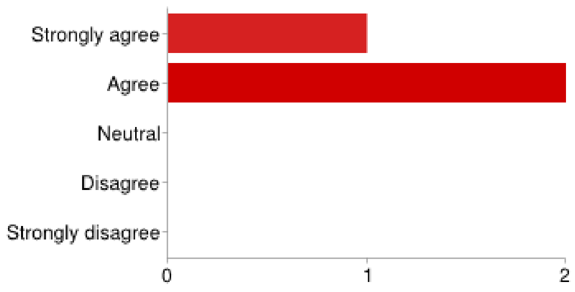


Figure 5-8 Answers to Q6

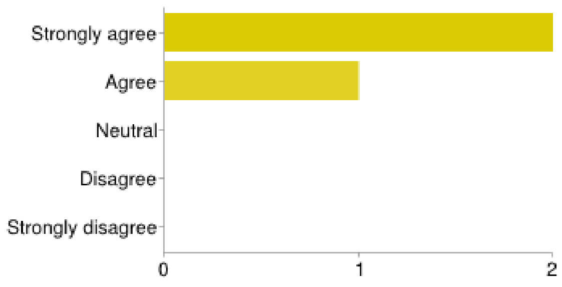


Figure 5-9 Answers to Q7

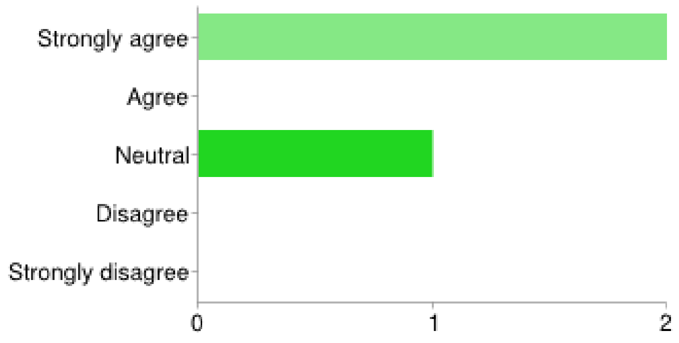


Figure 5-10 Answers to Q8

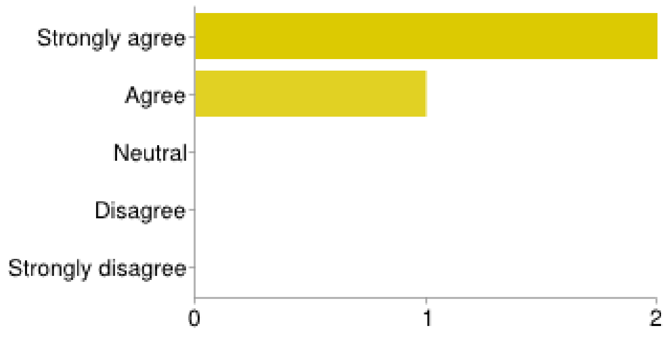


Figure 5-11 Answers to Q9

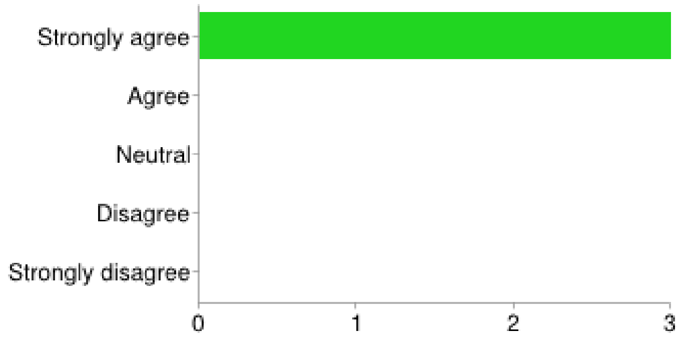


Figure 5-12 Answers to Q10

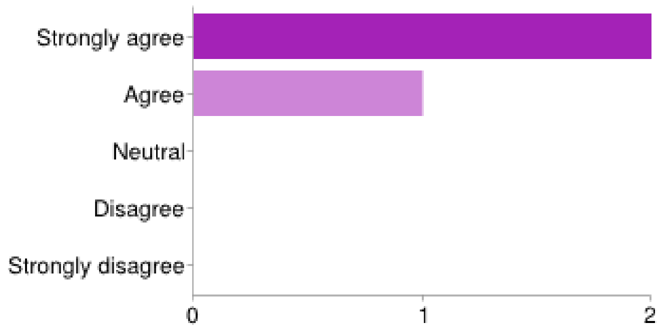


Figure 5-13 Answers to Q11

5.2.3 Quality

This validation aspect only covers representation quality. Answers to questions covered by this aspect can be seen in Figure 5-14 to Figure 5-17. Average score for this aspect is 4.42. The answers show that most respondents think that representation quality is not high. One respondent underlines this in a remark that there are many areas to be improved in terms of representation quality.

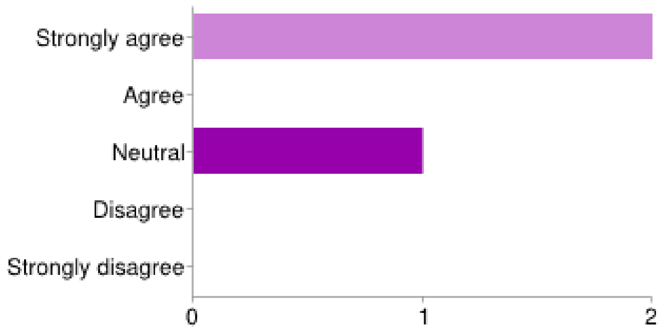


Figure 5-14 Answers to Q12

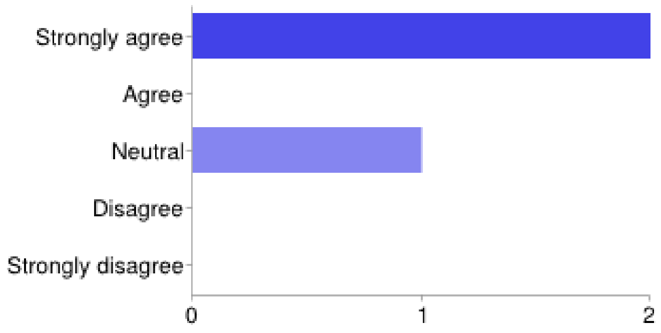


Figure 5-15 Answers to Q13

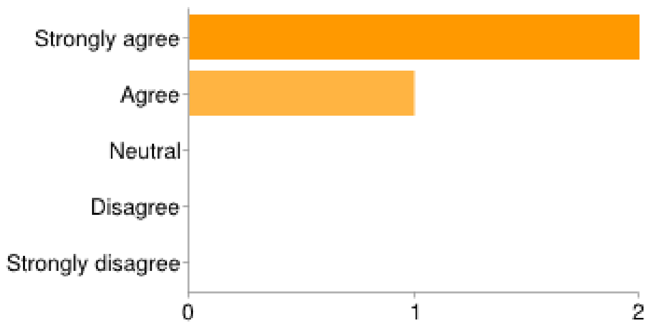


Figure 5-16 Answers to Q14

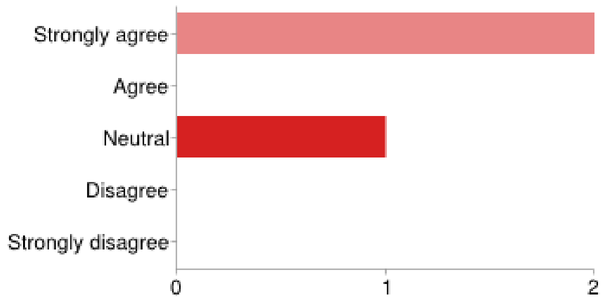


Figure 5-17 Answers to Q15

5.3 Summary

During the survey period (see section 5.1), three out of eight respondents gave their reactions. Overall, the survey gives a quite clear impression of what the users think about our prototype. One issue is that the number of people answering the survey that is limited to three respondents. A remark on overall prototype is that it gives a good first impression on basic coupling although is limited in terms of content retrieval. The latter is understandable by the respondent considering preliminary state of repository system. Overall scores of survey questions' answers are provided in Figure 5-18.

Another remark is found regarding Q8. The respondent mentioned that the question needs to address more on the ability to discover models and its properties using the prototype. Furthermore, a remark is made on location indicators. This can help a user by informing which part of model a user is looking at.

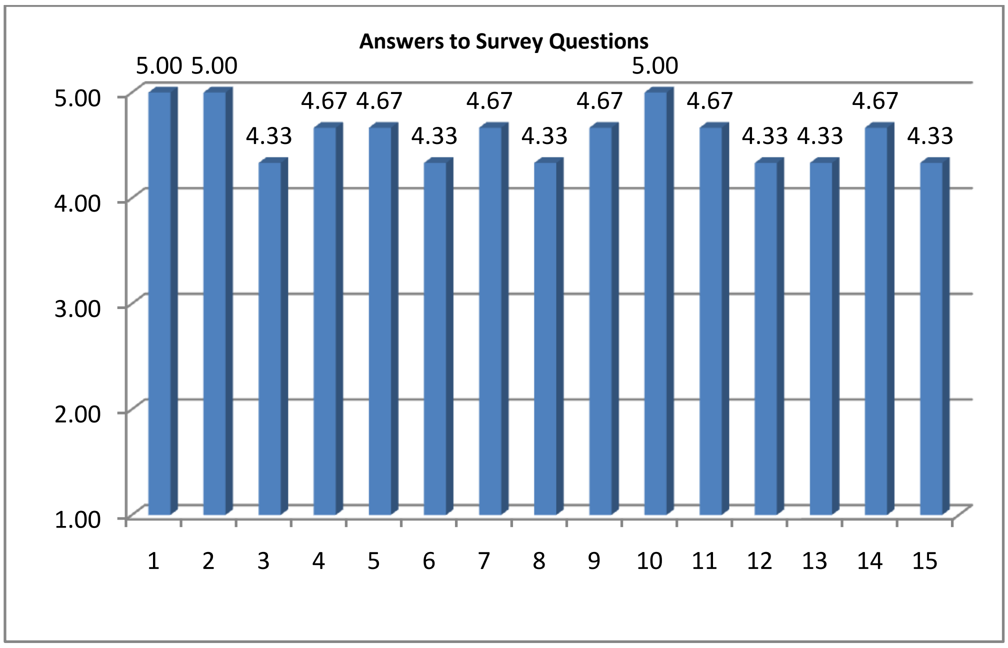


Figure 5-18 Answers to survey questions

Another comment is about things to improve given a better repository system is available. The respondent mentioned another image format and extra navigation links. Those comments can be useful to our reference architecture in terms of properties needed for visualization tool and model properties module. A comment on Q8 was made with remark on ambiguity of the question, including agreement if the transfer of models refers to ability to explore contents of models to users with the prototype. Comments on our prototype can be used to improve reference architecture design in terms of the specifications for visualization tool, model properties module, and repository system (see Figure 3-1). The remarks can be used to detail which properties to be generated and to be put onto external page or format of information generation.

Overall, answer to survey questions shows positive response on our prototype. This is reflected in Figure 5-18 with all questions' answer score above 4, which means that each respondent, by average, answers agree or strongly agree on our survey, including average scores for functionality, utility/usefulness, and quality. Consequently, this prototype is sufficient to facilitate prospective users based on criteria of visualization.

6. Final Remarks

6.1 Conclusion

We first revisit research questions defined in chapter 1.

What is the knowledge stored in a repository of an organization?

What type of knowledge is stored?

As discussed in subsection 2.7.2 and later in section 3.1, enterprise knowledge is the type of knowledge stored in the repository. It is based on interactions within an organization or an element of organization.

What are the formats of the knowledge stored?

Enterprise knowledge stored in the repository is stored in different formats. Generally, formats mentioned in section 2.6 (workflow, database design, business process models, data flow diagram, etc.) are used. For this research, section 4.2 specifies the formats to embed enterprise knowledge used in a repository: architecture models and business process models. In business process models and architecture models, knowledge units inside a repository are specified in form of properties of models, as explained in subsection 4.3.2.1.

How is wiki supporting enterprise knowledge visualization?

What are the advantages of a wiki?

Wiki (TWiki and MediaWiki) can provide necessary functions for knowledge management purposes (Liang et al., 2009). On MediaWiki, this is evident through the number of extensions that it has. Other than that, simplicity (Louridas, 2006) is also an advantage that a wiki has.

How can wiki support enterprise knowledge visualization?

Wiki can support enterprise knowledge visualization by acting as a visualization tool. In our enterprise knowledge visualization implementation (see section 4.3), visualization tool's role is taken by a wiki software. By means of external data service and handling, wiki can visualize enterprise knowledge from repository.

What is the design of a system that transfers enterprise knowledge from a repository to a wiki?

How to design reference architecture for visualizing enterprise knowledge?

To design a reference architecture for visualizing enterprise knowledge, we define quality goals to be achieved. These quality goals are then realized through a set of components for each layer of reference architecture.

A set of requirements for repository is also defined. These requirements are based on a previous research related to knowledge repository. Combined with quality goals, those repository requirements are necessary for a system basing its implementation on our reference architecture. Furthermore, we added relevant components to each layers of reference architecture. These components are added based on related literature.

How to implement a prototype based on the reference architecture?

In section 4.3, we describe how to conduct prototype implementation based on our reference architecture. Waterfall software development method was used to implement our prototype. PHP programming language is used for source code. We chose MediaWiki as visualization tool for this prototype implementation due to its ability to handle external knowledge formats, extensibility, and simplicity (see section 4.1). Testing and operation was conducted at BiZZdesign, a company whose case is being used in this research.

In implementing the prototype, enterprise knowledge embedding form is narrowed to business process models and architecture models. Those models are stored in a repository that facilitates their properties, including graphical representation. Those properties are extracted to XML format and in form of web pages with structured address. From there, MediaWiki as the chosen wiki software is coupled with repository system that generates those XML pages.

After defining enterprise knowledge embedding in models, we continue with specifying that embedding. We then defined goals and requirements for our reference architecture, especially for knowledge repository. These requirements (subsection 4.3.1.2) come from exploration of InSite, a portal of object and model viewer at BiZZdesign and a wiki presentation from ArchiXL. Business process model and architecture model were chosen as knowledge format in chapter 4. This was also the chapter where a case of enterprise knowledge visualization implementation at BiZZdesign is described. The implementation resulted in a prototype that visualizes business process models and architecture models using XML pages and MediaWiki. On a broader view, the reference architecture and implementation are parts of regulating and advancing IT support processes. Both processes were discussed in section 3.1.

Our prototype is then validated with a survey**.** This survey measured functionality, usefulness, and quality of representation. Responses to this survey were positive (see section 5.2). This shows that our prototype had succeeded in fulfilling functional requirements specified in chapter 4.

Chapter 3 provides us a reference architecture containing guidelines and requirements to implement an end system. Using the reference architecture, a system developer can apply those requirements and guidelines to be used to implement an end system. The reference architecture is designed so that the end system can support many knowledge visualization tools and be applicable to repositories containing any kind of models. Meanwhile, product of chapter 4 is a prototype based on that reference architecture. The prototype can give a preliminary illustration on how to implement a system to present enterprise knowledge in an organization environment. BiZZdesign could base its further development of the repository system on our prototype implementation. This master thesis contributes to people reading it by providing new insight on enterprise knowledge, its presentation and how users relate to it. It could also be a practical example for knowledge visualization purposes.

6.2 Limitations

This research has some limitations. First, repository system development stage that is still early provided some constraints. Albeit repository system's address structure is sufficient for navigation, there are other constraints such as empty XML tags or tags with 'unknown' as a value (see Figure 4-13). Also only a few XML pages have documentation, as mentioned in subsection 4.3.3.4. Other limitations related to repository system are non-functional requirements such as security and cost. The former was discussed shortly in subsection 3.2.3. Navigation in our prototype is done by giving link to parent modelpackage. To go to previous page, users need to use buttons on their keyboard that functions to go to previously visited page.

The prototype based on our reference architecture was not yet tested thoroughly in terms of layout stability. Layout stability testing includes testing on browsers set at small window size, and testing with computers with smaller screen size. Due to time limitation, our repository system and coupling with wiki haven't been tested with different simultaneous access. Time limitation also prevents us to validate the product of chapter 3 and check BiZZdesign repository's fulfillment of the thirteen repository requirements thoroughly. Thus, we focus to validate the prototype implemented based on that reference architecture. This eventually also has its own constraint, since only three respondents out of eight gave their reactions. Moreover, documentation limitation restricted us from exploring BiZZdesign's repository system. It also gives constraints on specifying InSite features and ArchiXL's wiki presentation features in relation to requirements in subsection 4.3.1.

6.3 Future work

In the future, simultaneous access checking should be added to make sure that repository system and selected MediaWiki extension can handle it. Validation in chapter 5 is based more on functional requirements. On this limitation, an evaluation consisting both functional and non-functional aspects can be conducted on future research. Validation of reference architecture itself would complement this research. Moreover, future research of end systems with different wiki software or different visualization tool(s) can be done. Case studies containing those different software or tools can then be used to compare perception of users and cost of instantiating the reference architecture, for example. Models other than business process models and architecture models can also base future works.

Currently, layout on a wiki page is fixed at certain measurements. This means that when a browser’s window size is adjusted, image and documentation text can collide. Flexible layout using other MediaWiki extensions should be added for more stability of layout.

Moreover, research needs to be done to examine integrity of extracted knowledge format. Britton and Bye (2004) discussed this and mentioned that the usage of XML for knowledge/data formats consume more time and more bandwidth. Different knowledge formats as a bridge between repository and a visualization tool needs to be tested. Discussion on decision making exists in subsection 3.2.1. However, we think there is at least one further step needed for the prototype implementation to be of help in decision making in an enterprise or organization. This can be achieved with features like regular process performance report, different privileges for different user group, and better graphical representation.

Furthermore, other future work includes modification from current prototype (subsections 4.4.2 and 4.4.3). This covers documentation editing, use of other MediaWiki extensions, extra information in XML tags for navigation purposes, and images with different resolution for better representation and previewing purposes. Extra information in XML tags can also be used to group search results. On search function, an identifier of object should be added to enable user to search with an identifier. Graphical representation is also an issue in which coordinate assignments for certain objects’ image need to be done. Documentation editing would need a function to replace the content of a tag in an XML page. This can be done with a temporary container in the repository system.

Acknowledgements

With the completion of this thesis, I would like to thank my supervisors Maria Iacob, Klaas Sikkel, Harm Bakker, and Wilco Engelsman.

Without their insights and guidance, it would not be possible for my master study to come to this point.

I would also like to thank my family and lecturers for every advice and support that they gave to me during my study. Also colleagues and contributing people that I can’t mentioned here one by one.

Sulistyo U Wicaksono

Depok, August 2012

References

Andersen, E. (2004). Using wikis in a corporate context. Working Draft (Norwegian School of Management BI), Dated November 10, 2004.

Andersson B., Bider, I., & Perjons, E. (2004). Integration of Business Process Support with Knowledge Management - A Practical Perspective. Proc. of the 5th Int. Conf. of Practical Aspects of Knowledge Management, Vienna, Austria, p. 227-238.

Argote, L., McEvily, B., Reagans, R. (2003). Managing Knowledge in Organizations: an integrative framework and review of emerging themes. Management Science.49: p.571-582.

ArchiXL (2012). ArchiXL IT Referentie Architectuur. http://www.wikixl.nl/wiki/itrefarch2, Retrieved August 11, 2012.

Ba, S., Whinston, A. Lang, K.R. (1995). An enterprise modeling approach to organization decision support, Proceedings of the 28th International Conference on System Sciences, p. 312 – 320.

Bass, L., Clements, P., & Kazman, R. (2003). Software Architecture in Practice. Addison-Wesley.

Bernstein, P.A. & Dayal, U. (1994). An overview of repository technology. In Proceedings of VLDB, Santiago de Chile, Chile.

Berio, G & Vernadat, F. (1999). New developments in enterprise modeling using CIMOSA. Computers in Industry, 40:p.99–114.

BiZZdesign. (2011). Model-Driven Engineering at BiZZdesign. Lecture slides.

BiZZdesign. (2012). Company info, from http://www.bizzdesign.com, Retrieved May 27, 2012.

Bobrik R., Bauer T., & Reichert M. (2006). Proviado-personalized and configurable visualizations of business processes. In: Proc. EC-WEB'06, LNCS 4082, p. 61-71.

Boulain, P., Parker, M., Millard, D., & Wills, G. (2006) Weerkat: An extensible semantic wiki. In: Proceedings of 8th Annual Conference on WWW Applications, Bloemfontein, Free State Province, South Africa.

Britton, C. and Bye, P. (2004). IT Architectures and Middleware, Strategies for Building Large Integrated Systems, second ed., Addison-Wesley.

Buffa, M. (2006). Intranet wikis. In Proceedings of Intraweb workshop, WWW'06.

Das, A. (2003). Knowledge and productivity in technical support work. Management Science. 49(4), p.416-431.

Davenport, T.H. Eccles, R.G., & Prusak, L. (1992). Information Politics. Sloan Management Review. 34(1): p.53-65.

Di Iorio, A. & Zacchiroli, S. (2006). Constrained wiki: an oxymoron? In WikiSym '06: Proceedings of the 2006 international symposium on Wikis, pages 89{98, New York, NY, USA, 2006. ACM Press.

Dijkman, R. & Joosten, S. (2002). Deriving use case diagrams from business process models. Technical Report 02-08, CTIT, Enschede.

Dingsoyr, T. & Royrvik, R. (2003). An Empirical Study of an Informal Knowledge Repository in a Medium-Sized Software Consulting Company. in:Proceedings of the International Conference on Software Engineering, Portland, OR, United States, 2003, pp. 84–92.

Domokos, P. & Varro, D. (2002). An open visualization framework for metamodel-based modeling languages. In: Proceedings of International Workshop on GraphBased Tools (GraBaTs'02), Band 72 (2) der Reihe ENTCS, Seiten 78–87, Barcelona, Spain.

Ebersbach, A., Glaser, M., & Heigl, R. (2006). Wiki-Web Collaboration. Springer, p.18.

Eppler, M., & Burkhard, R. (2004). Knowledge Visualization. Towards a New Discipline and its Fields of Application. ICA Working Paper 2/2004, University of Lugano.

Ernst, A.M., Lankes, J., Schweda, C.M., & Wittenburg, A. (2006). Tool Support for Enterprise Architecture Management-Strength and Weaknesses. In 10th IEEE International Enterprise Distributed Object Computing Conference, Hong Kong., p.13–22.

Farenhorst, R. & van Vliet, H. (2008). Experiences with a Wiki to Support Architectural Knowledge Sharing. In: Proceedings of the 3rd Workshop on Wikis For Software Engineering.

Firestone, J. M. (2000). Enterprise knowledge portals: what they are and what they do. In: Knowledge and Innovation: Journal of the KMCI 1, Nr. 1, p. 85–108.

Fox, M.S., Gruninger, M. (1998). Enterprise modeling. AI Magazine 19(3) (1998) p.109–121.

Gallagher, B.P. (2000). Using the architecture tradeoff analysis method to evaluate a reference architecture: a case study. Technical Report CMU/SEI-2000-TN-007, Carnegie Mellon University, Software Engineering Institute.

Gallagher, K., Hatch, A., & Munro, M. (2008). Software architecture Visualization-An Evaluation Framework and its Application. IEEE Trans. Software Eng., vol. 34, no. 2, p. 260-270.

Gonzalez-Reinhart, J. (2005). Wiki and the Wiki Way: Beyond a Knowledge Management Solution. Information Systems Research Center, p. 1–22.

Gorton, I. (2011). Essential Software Architecture. Springer-Verlag Berlin Heidelberg. p.1-15.

Goul, M. and Corral, K. (2007) "Enterprise model management and next generation decision support," Decision Support Systems, vol. 43, pp. 915-932.

Hasan, H. and Pfaff, C.C. (2006). The Wiki: a tool to support the activities of the knowledge worker. In Proceedings at the Transformational Tools for 21st Century (TT21) 2006 Conference. Central Queensland University, Rockhampton, Queensland, Australia.

Hevner, A., March, S., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly (28:1), 2004, p.75-105.

Hoogeboom, G. (2005). Architecture & Visualization. Master Thesis. Radboud University Nijmegen: The Netherlands.

Huo, M., Verner, J., Zhu, L., Babar, M. A. (2004). Software Quality and Agile Methods. In Proceedings of COMPSAC 04, IEEE Computer Soc., 2004, p. 520-525.

Iacob, M., Jonkers, H., & Wiering, M. (2004). Towards a UML profile for the ArchiMate Language. Telematica Instituut & Leiden Institute for Advanced Computer Science.

Iacob, M., Franken, H., & Berg, H. V. (2007a). Enterprise Architecture Handbook. BiZZdesign Academy Publishers: Enschede, The Netherlands.

Iacob, M., Franken, H., Berg, H., Bakker, H. (2007b). Capturing Architecture for the Adaptive Enterprise-creating the blueprint for change. Via Nova Architectura.

Keller, T. and Tergan, S. (2005) Visualizing Knowledge and Information: An Introduction, In: Tergan, S. and Keller, T. (Eds). Knowledge and Information Visualization: Searching for Synergies. Springer-Verlag: Heidelberg. 2005. p. 1-23.

Kurfess, F. (2008). Knowledge Presentation and Visualization. Lecture slides of Knowledge Presentation. California Politechnic State University, United States.

Kuhn, H., Bayer, F., Junginger, S., Karagiannis, D. (2003). Enterprise Model Integration. In: Bauknecht, K., Tjoa, A., Quirchmayr, G. (Eds.): Proceeding of the 4th International Conference EC-Web 2003 – Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer Verlag, pp. 379-392.

Lankhorst, M., Proper, H., and Jonkers, H. (2009). The Architecture of the ArchiMate Language. In Enterprise, Business-Process and Information Systems Modeling: 10th International Workshop, Bpmds 2009, and 14th International Conference, Emmsad 2009, Held at Caise 2009, Amsterdam, The Netherlands, June 8-9, 2009, Proceedings, p. 367. Springer.

Liang, M., Chu, S., Siu, F., & Zhou, A. (2009). Comparing User Experiences in Using Twiki & Mediawiki to Facilitate Collaborative Learning. Proceedings of the 2009 International Conference on Knowledge Management.

Louridas, P. (2006). Using wikis in software development. IEEE Software, 23(2), p. 88-91.

McMahon, C. (2008). A synthesized knowledge mapping framework to embed a KM strategy using topic maps and wikis: the Tao of Wiki. Master Thesis. Dublin University of Technology, Scotland.

Moody, D. (2009). The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Trans.Software Eng., vol. 35, no. 6, p. 756–779.

Nonaka, I. (1991). The knowledge-creating company. Harvard Business Review. 69(6):p.96-109.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. Organization Science, Vol. 5: p.14-37.

O'Leary, D. E. (1998). Enterprise Knowledge Management. Computer Magazine, 31(3) p.54-61.

Purvis, R. L., Sambamurthy, V. Zmud, R.W. (2001). The assimilation of knowledge platforms in organizations: An empirical investigation. Organ. Sci. 12(2): p.117–135.

Rospocher, M., Ghidini, C., Pammer, V., Serafini, L., and Lindstaedt, S. (2009) MoKi: the modelling wiki. In Proceedings of the Forth Semantic Wiki Workshop (SemWiki 2009), co-located with 6th European Semantic Web Conference (ESWC 2009), volume 464 of CEUR Workshop Proceedings.

Quigley, E. J. & Debons, A. (1999). Interrogative Theory of Information and Knowledge. in Proceedings of SIGCPR '99, ACM Press, New Orleans, LA., p. 4-10.

Stenmark, D. (2001). The Relationship between Information and Knowledge. in Proceedings of IRIS 24, Ulvik, Norway, August 11-14.

Van Heijst, G., van der Spek, R., & Kruizinga, E. (1997). Corporate memories as a tool for knowledge management. Expert Systems with Applications.13(1): p.41-54.

Van Wasbeek, E. (2006). A repository for the management of artifacts in system engineering. Master Thesis. University of Twente, Enschede.

Vernadat, F. (2001). UEML: Towards a Unified Enterprise Modelling Language. Proc. 3ème Conférence Francophone de Modélisation et Simulation (MOSIM'01), Troyes, France, 25–27 April 2001, p. 3–13.

Yan, Z., Dijkman, R. & Grefen, P. (2009). Business Process Model Repositories-Framework and Survey. Technical report, Eindhoven University of Technology, The Netherlands.

Zack, M. (1999). Managing Codified Knowledge. Sloan Management Review. 40(4): p.45-58.

Appendix A. Prototype source code

Initializer

```
<?php

class SpecialVisInit extends SpecialPage {

       function __construct() {

              parent::__construct( 'VisInit' );

              wfLoadExtensionMessages('VisInit');

                                                    $this->mIncludable = true;

       }

                                                    //Initializer. One of the main components of enterprise knowledge visualization prototype.

       function execute( $par ) {

              global $wgRequest, $wgOut;

                             /*Preliminary PHP code starts*/

        $this->setHeaders();

        $param = $wgRequest->getText('param');

                             /*Preliminary PHP code ends*/


                             /* Content type. This can be XML or binary or other types of data */

                              header( "Content-type: text/x-wiki; charset=utf-8" );

              // Start visualizing modelpackages on root level of the repository


                                                    $base = 'http://showcase.bizzdesign.nl/rest/';                                   //Address    of
the repository system's set of XML addresses

                                                    $url = $base.'/modelpackages';
              //The root address of the repository system's set of XML addresses

                                                    $mdlarray = array(); $mdlid = array(); $mdlref = array();//Setting the containers for models
and their properties

                                                    $init = true;

                                                    $wgAllowExternalImagesFrom = array ($base);
              //Activating the use of external image from address

                                                    $sxe = simplexml_load_file($url);
              //Containing XML tags and contents in a simpleXML container
```

```
if ($sxe==null) {                                                                    //Handling
error in search query
                                        $wgOut->addWikiText("Search query is currently not supported for repository
searching. Please search use alpha-numeric characters and please use the OR operator instead if you use '||'.<br>");
                                        exit;
                                }
                                foreach ($sxe->modelpackage as $mdl) {
        //Assigning content of simpleXML structure tables of name, id and references.
                                        $mdlarray[] = $mdl->name;
                                        $mdlid[] = $mdl->id;
                                        $mdlref[] = $mdl->attributes()->ref;
                                }

                        //Spacer, for layouting purposes
                                $wgOut->addWikiText("





                                ");


                                if ($init==true){
        //Checking the emptyness of container of modelpackage
                                                if (sizeof($mdlarray)==0){$wgOut->addWikiText("(There are no
modelpackages in the repository)");}
                                                else {
                                                        $i = 0;
                                                        for ($i=0; $i<count($mdlarray); $i++) {        //Listing the
modelpackages

                                                                $mdlno = substr($mdlref[$i], -6);
                                                                $wgOut-
>addWikiText("*[[Special:Visualize/$mdlref[$i]$mdlno|'''$mdlarray[$i]''']]");


                                                        }
                                                }
                                }
```

```
                    if ($init==false) {
          //Handling empty case
                         $wgOut->addWikiText("





                              <span style=\"font-size: large;\">'''$mname'''</span>

                         ");

                    }
                         //Start filling spacer for front page notification below

                    $output="


                         {{#get_web_data:

                         url=$url

                         |format=XML

                         |data=doc=documentation,mid=modelpackage->attributes()->ref

                         }}

















                         {{#external_value:doc}}

{{#clear_external_data:}}
```

This prototype is built as an instance of the research titled Enterprise Knowledge Visualization. The knowledge visualized are that embedded in business process models and architecture models. Not all properties of the model are available on the repository service, thus there will be limitations of the properties/images visualized in this prototype.

```
                         ";
```

```
        $wgOut->addWikiText( $output );                                      //Displays spacer to wiki page


                                        //End notification

                                $output1="The BiZZdesign logo on top left links to the home page of this prototype.

                                        ";


                                $wgOut->addWikiText( $output1 );
         //Displays end notification to wiki page



    }

}

?>
```

Visualizer

```
<?php
class SpecialVisualize extends SpecialPage {
      function __construct() {
            parent::__construct( 'Visualize' );
            wfLoadExtensionMessages('Visualize');
                                                  $this->mIncludable = true;
      }
                                                  //Visualizer. One of the main components of enterprise knowledge visualization prototype.


      function execute( $par ) {
            global $wgRequest, $wgOut;
                              /*Preliminary PHP code starts*/
            $this->setHeaders();


       $param = $wgRequest->getText('param');
                              /*Preliminary PHP code ends*/


                              /* Content type. This can be XML or binary or other types of data */
                               header( "Content-type: text/x-wiki; charset=utf-8" );
                              $base = 'http://showcase.bizzdesign.nl/rest/';            //Address of the repository system's set of XML address
                                                  $pathso = 'objects/';
               //Assigns path for objects to a container
                                                  $pathsm = 'modelpackages/';
               //Assigns path for modelpackage to a container
                                                  $pathsimg = 'views/';
               //Assigns path for graphical representation to a container
                                        $pathsdoc = 'documentation/';                                              //Assigns      path      for
documentation to a container
                                                  $bool = false;
                                                  $prntid = ""; $prntids = "";                                               //Setting
parent id
```

```
                    if (substr($par, 14, 6)=="search") {                    //The case of requested
path is for searching

                         $objectid = substr($par, 8, 6);

                         $url = $base.$pathso.$objectid;

                         $bool = true;

                         $prntids = substr($par, 20, -1);

                         $prntids=str_replace("_", ' ', $prntids);

                    }

                    else if (substr($par, 0, -13)=="objects") {             //The case of requested
path is for calling an object address

                         $objectid = substr($par, 8, 6);

                         $url = $base.$pathso.$objectid;

                         $bool = true;

                         $prntid = substr($par, -6);

                    }

else if (substr($par, 0, -13)=="modelpackages") {   //The case of requested path is for calling a modelpackage

                         $objectid = substr($par, 14, 6);

                         $url = $base.$pathsm.$objectid;

                         $bool = true;

                         $prntid = substr($par, -6);

                    }
```

```
            $ch = $sxe->child;            //Assigning content of simpleXML structure
        tables of name, id and references of child(ren) which is named.

            $i = 0;

            foreach ($ch as $chd) {

                if ($chd->name!="") {

                    $nmdchref[] = $chd->attributes()->ref;

                    $nmdchnm[] = $chd->name;

                    $nmdchid[] = $chd->id;

                }

            }

            //Assigning content of simpleXML structure tables of name, id and references
of child(ren) which has id.

            $chld = $sxe->child;


             foreach ($chld as $chd) {

                if ($chd->id!="") {

                    $iddchid[] = $chd->id;

                    $iddchnm[] = $chd->name;

                    $iddchref[] = $chd->attributes()->ref;

                }

            }

            //Filtering contents of tables of name, id and references of child(ren) which are
named and are views.

            $nmview = array(); $idview = array(); $nmviewref = array();

            for ($i=0; $i<count($nmdchnm); $i++) {

                    if (strtolower(substr($nmdchnm[$i], -4))=="view"){

                        $nmview[] = $nmdchnm[$i];

                        $idview[] = $nmdchid[$i];

                        $nmviewref[] = $nmdchref[$i];

                    }

                }

    //Filtering contents of tables of name, id and references of child(ren) which have id and are not views.

            $nmnonview = array();$nmnonviewref = array();$idnonview = array();


                for ($i=0; $i<count($iddchid); $i++) {

                    if (strtolower(substr($iddchnm[$i], -4))!="view") {

                        $nmnonview[] = $iddchnm[$i];

                        $idnonview[] = $iddchid[$i];
                    $nmnonviewref[] = $iddchref[$i];
                            }
                    }

                    $oname = $sxe->name;
                    $oid = $sxe->id;

                    if ($prntids!=""){     //Indicates that the prototype is visualizing search results
                            $output1="

                            <span style=\"font-size: large;\">'''$oname'''</span><div
style=\"visibility: hidden;\">'''ignore this very unimportant text'''</div>[[Special:VisSearch/$prntids|<u>Go to '$prntids' Search
Results</u>]]
                            ";
                            //Above is spacer for layout purposes
                            $output1point5="";
                            $output2="{{#get_web_data:
                                    url=$url
                                    |format=XML
        |data=doc=documentation,ch=child,ty=type,id=id,chid=icon->attributes()->ref
                                    }}
                                    {|class=\"infobox\" style=\"float: right; margin-left: 0em;
width: 333pt; font-size: 90%;\"
                                    |-
                                    | colspan=\"2\"|$base$pathsimg$oid?.png
```

```
                                |- style=\"text-align:left;\"
                                !



                                |


                                |- style=\"text-align:left;\"


                                !



                                Consists of:
                                |}
                        ";//Above is an info box containing with image of the object
                }
                else if (substr($par, 14, 6)==substr($par, 20, 6) OR substr($par, 14,
6)==substr($par, 8, 6)) {
                        //Indicates that the object to visualize is a root modelpackage
                        //Containing modelpackage name
                        $output1="




                        <span style=\"font-size: large;\">'''$oname'''</span>
                        ";//Checking availability of documentation
                        if ($sxe->documentation==""){

                        $output1point5="




                        <span style=\"font-size: 90%;\">(Documentation is empty)</span>
                        ";
                        }
                        else {$output1point5="";}
```

```
//Containing info box with modelpackage image and link to index page
                                                              $output2="{{#get_web_data:
                                                                       url=$url
                                                                       |format=XML

        |data=doc=documentation,ch=child,ty=type,id=id,chid=icon->attributes()->ref
                                                                       }}
                                                                       {|class=\"infobox\" style=\"float: right; margin-left: 0em;
width: 333pt; font-size: 90%;\"
                                                                       |-
                                                                       | colspan=\"2\"|$base$pathsimg$oid?.png



                                                                       |- style=\"text-align:left;\"
                                                                       ![[Special:VisInit|<u>Go to repository index</u>]]



                                                                       |


                                                                       |- style=\"text-align:left;\"


                                                                       !


                                                                       Consists of:
                                                                       |}
                                                              ";




                                                     }
   else {//Indicates that the object to visualize is an object (not a modelpackage)
                                                              $mdlpkgnm="";
                                                              $output1="
```

```
<span style=\"font-size: large;\">'''$oname'''</span>
";
//Checking availability of documentation
if ($sxe->documentation==""){
$output1point5="



<span style=\"font-size: 90%;\">(Documentation is empty)</span>
";
}
else {$output1point5="";}
//Containing info box with object image and a link to its root modelpackage
$output2="{{#get_web_data:
        url=$url
        |format=XML

        |data=doc=documentation,ch=child,ty=type,id=id,chid=icon->attributes()->ref
        }}
        {|class=\"infobox\" style=\"float: right; margin-left: 0em; width: 333pt; font-size: 90%;\"

        |-
        | colspan=\"2\"|$base$pathsimg$oid?.png


        |- style=\"text-align:left;\"
        ![[Special:Visualize/$pathso$prntid$prntid|<u>Go to root model package</u>]]


        |

        |- style=\"text-align:left;\"

        !
```

```
                                                  Consists of:
                                                  |}
                                          ";



                                  }
                                          //Displays object name, documentation and info box according to
each type handled above
                                  $wgOut->addWikiText( $output1.$output1point5.$output2 );


                                          //Displaying views inside an object
                                  if (count($idview)!=0) {

                                          $wgOut->addWikiText("{|style=\"margin-left:  54em;  display:  inline;
width: 333pt; float:right; font-size: 90%;\"
                                          |View(s):
                                          |}
                                                  ");



                                          //Displaying link to each view inside an object
                                          for ($i=0; $i<count($idview); $i++) {

                                                  $wgOut->addWikiText("{|style=\"margin-left:          54em;
display: inline; width: 333pt; float:right; font-size: 90%;\"
                              |[[Special:Visualize/$nmviewref[$i]$prntid|$nmview[$i]]]
                              |}
                                          ");

                                          }//Displaying objects inside an object
                                          $wgOut->addWikiText("{|style=\"margin-left:      54em;      visibility:
hidden; width: 333pt; float:right; font-size: 90%;\"
                                                          |Objects:
                                                          |}
                                          ");
```

```
                                        if (count($idnonview)!=0) {
                                        $wgOut->addWikiText("{|style=\"margin-left:       54em;
display: inline; width: 333pt; float:right; font-size: 90%;\"
                                            |Object(s):
                                            |}
                                                ");
                                        //Displaying link to each object inside an object
                                        for ($i=0; $i<count($idnonview); $i++) {


                                                $wgOut-
>addWikiText("{|style=\"margin-left: 54em; display: inline; width: 333pt; float:right; font-size: 90%;\"

    |[[Special:Visualize/$nmnonviewref[$i]$prntid|$idnonview[$i] $nmnonview[$i]]]
                                            |}
                                                ");

                                        }
                                    }



                                }
                                else if (count($iddchid)!=0) {     //Displaying    modelpackage    and/or    objects
inside a root modelpackage
                                    if  (substr($par,  14,  6)==substr($par,  20,  6)  OR  substr($par,  14,
6)==substr($par, 8, 6)) {
                                        $wgOut->addWikiText("{|style=\"margin-left:       54em;
display: inline; width: 333pt; float:right; font-size: 90%;\"
                                            |Model package(s):
                                            |}
                                                ");
                                    }
                                    else {

                                    $wgOut->addWikiText("{|style=\"margin-left:  54em;  display:  inline;
width: 333pt; float:right; font-size: 90%;\"
                                        |Objects:
                                        |}
                                            ");          }
                                        //Listing objects contained(child(ren))
                                    for ($i=0; $i<count($iddchid); $i++) {

                                                $wgOut->addWikiText("{|style=\"margin-left:
54em; display: inline; width: 333pt; float:right; font-size: 90%;\"
                                        |[[Special:Visualize/$iddchref[$i]$prntid|$iddchid[$i]
$iddchnm[$i]]]
                                        |}
                                                ");

                                    }

                                }
                                else {
        //Handling the case of no objects contained (no children)
                                    $wgOut->addWikiText("{|style=\"margin-left:  54em;  display:  inline;
width: 333pt; float:right; font-size: 90%;\"
                                                |(None)
                                                |}
                                                        ");
                                }

        //Below is a long spacer for wiki page layout purposes
                                $output3="{|style=\"margin-left: 54em; display: inline; width: 333pt; float:right;
font-size: 90%;\"
                                    |
                                                        |}
                                        ";
                                        $wgOut->addWikiText( $output3 );

        //Containing documentation and clearing external data container below
                                        $output4="











                                        <div style=\"position: absolute; border:thin solid white; margin-right: 0em;
width: 444pt; font-size: 90%;\">{{#external_value:doc}}</div>
                                                                {{#clear_external_data:}}
                                                                ";

        //Displaying documentation
                                        $wgOut->addWikiText( $output4 );







                                  }
   }
}
?>
```

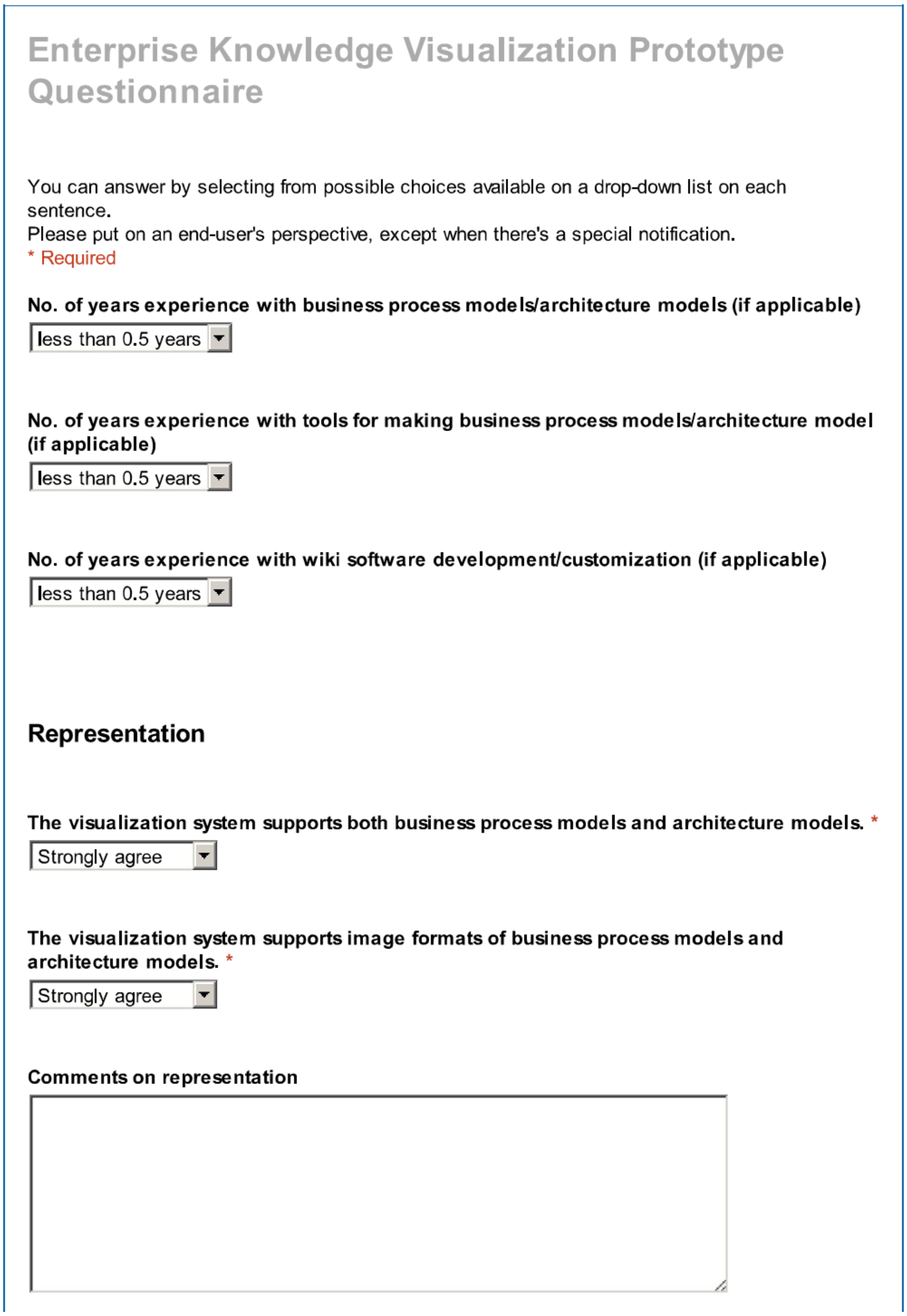
Search Bridge

```
<?php

class SpecialVisSearch extends SpecialPage {

        function __construct() {

                parent::__construct( 'VisSearch' );

                wfLoadExtensionMessages('VisSearch');

                                                                $this->mIncludable = true;

        }


                                                                //Search Bridge. One of the main components of enterprise knowledge visualization
prototype.


        function execute( $par ) {

                global $wgRequest, $wgOut;

                                                                /*Preliminary PHP code starts*/

                $this->setHeaders();




                                                $param = $wgRequest->getText('param');

                                                                /*Preliminary PHP code ends*/


                                                /* Content type. This can be XML or binary or other types of data */

                                                header( "Content-type: text/x-wiki; charset=utf-8" );




                //Start handling search input

                                                                $base = 'http://showcase.bizzdesign.nl/rest/';

                                                                $alpnum = ctype_alnum(trim($par));

                                                                if ($par=="") {


                                                                                $paths = 'search?q=';                                   //Assigns search path to
a container

                                                                                $sterm = $_POST["searchterm"];              //Assigns session handling to a
container

                                                                                $url = $base.$paths.$sterm;                             //Defines URL for calling
repository system's search function by appending paths
```

```
$sxe = simplexml_load_file($url); //Containing XML tags and contents in a simpleXML container
                    if ($sxe==null) {                                       //Handling error in
search query
                    $wgOut->addWikiText("Search query is currently not supported for repository searching.
Please search use alpha-numeric characters and please use the OR operator instead if you use '||'.<br>");
                    exit;
                    }
                    //Allocating tables of search results, table of ids of search results, and table of references
of search results
                    $rsltarray = array(); $rsltid = array(); $rsltref = array();
                    foreach ($sxe->result as $rslt) {
                         $rsltarray[] = $rslt->name;
                         $rsltid[] = $rslt->id;
                         $rsltref[] = $rslt->attributes()->ref;


                    }
                    //Error message for search
                    if (count($rsltarray)==0) {
                         $wgOut->addWikiText("There isn't any of repository contents that matched
your query '$sterm'<br>");
                    //External data extension
                         $output="{{#get_web_data:
                              url=$url
                              |format=XML
                              |data=doc=documentation,mid=modelpackage->attributes()->ref
                              }}









                              {{#external_value:doc}}


                              {{#clear_external_data:}}
                              ";
                         $wgOut->addWikiText( $output );
```

```
                                        }
                                        else {
                                                $jmlh = count($rsltarray);            //Assigns number of search result
to a container
                                                $wgOut->addWikiText("$jmlh object(s) found with the query '$sterm'.<br>");
      //Displays number of search results
                                                $wgOut->addWikiText("'''Search Results:'''<br><br>");        //Displays
text of Search Results heading message
                                                $i = 0;

                                                                  //Lists the search results to a wiki page and
assigns link to them
                                                for ($i=0; $i<count($rsltarray); $i++) {



                                                $wgOut-
>addWikiText("*[[Special:Visualize/$rsltref[$i]search$sterm.|$rsltarray[$i]]]");


                                                }
                                                //External data extension
                                                $output="{{#get_web_data:
                                                      url=$url
                                                      |format=XML
                                                      |data=doc=documentation,mid=modelpackage->attributes()->ref
                                                      }}
                                  else {
      //Search query from another page
                                        $par=str_replace("_", ' ', $par);          //Handling character replacements on search
query
                                        $sterm = $par;
      //Assigns search query to a container
                                        $paths = 'search?q=';                              //Sets path for passing
to repository address
                                        $url = $base.$paths.$sterm;                        //Sets container for
search with appending paths
                                        $sxe = simplexml_load_file($url);          //Containing XML tags and contents in a
                                  simpleXML container
                                        $rsltarray = array(); $rsltid = array(); $rsltref = array();
                                        foreach ($sxe->result as $rslt) {           //Contains search results on containers of
                                name, id and references
                                                $rsltarray[] = $rslt->name;
                                                $rsltid[] = $rslt->id;
                                                $rsltref[] = $rslt->attributes()->ref;


                                        }



                                        $jmlh = count($rsltarray);                                  //Counts and containt
search results
                                        $wgOut->addWikiText("$jmlh object(s) found with the query '$sterm'.<br>"); //Displays
number of objects found with the query
                                        $wgOut->addWikiText("'''Search Results:'''<br><br>");           //Displays Search Result
heading message
                                        $i = 0;


                                        for ($i=0; $i<count($rsltarray); $i++) {
        //Lists search results to a wiki page and assigns links to them
                                        $wgOut->addWikiText("*[[Special:Visualize/$rsltref[$i]search$sterm.|$rsltarray[$i]]]");


                                        }
                                        //External data extension
                                        $output="{{#get_web_data:
                                                url=$url
                                                |format=XML
                                                |data=doc=documentation,mid=modelpackage->attributes()->ref
                                                }}




                                                {{#external_value:doc}}
                                                {{#clear_external_data:}}";
                                        $wgOut->addWikiText( $output );
                                }
        }
    }
?>
```

Appendix B. Survey questions



Enterprise Knowledge Visualization Prototype Questionnaire

You can answer by selecting from possible choices available on a drop-down list on each sentence.
Please put on an end-user's perspective, except when there's a special notification.
* Required

No. of years experience with business process models/architecture models (if applicable)

less than 0.5 years

No. of years experience with tools for making business process models/architecture model (if applicable)

less than 0.5 years

No. of years experience with wiki software development/customization (if applicable)

less than 0.5 years

Representation

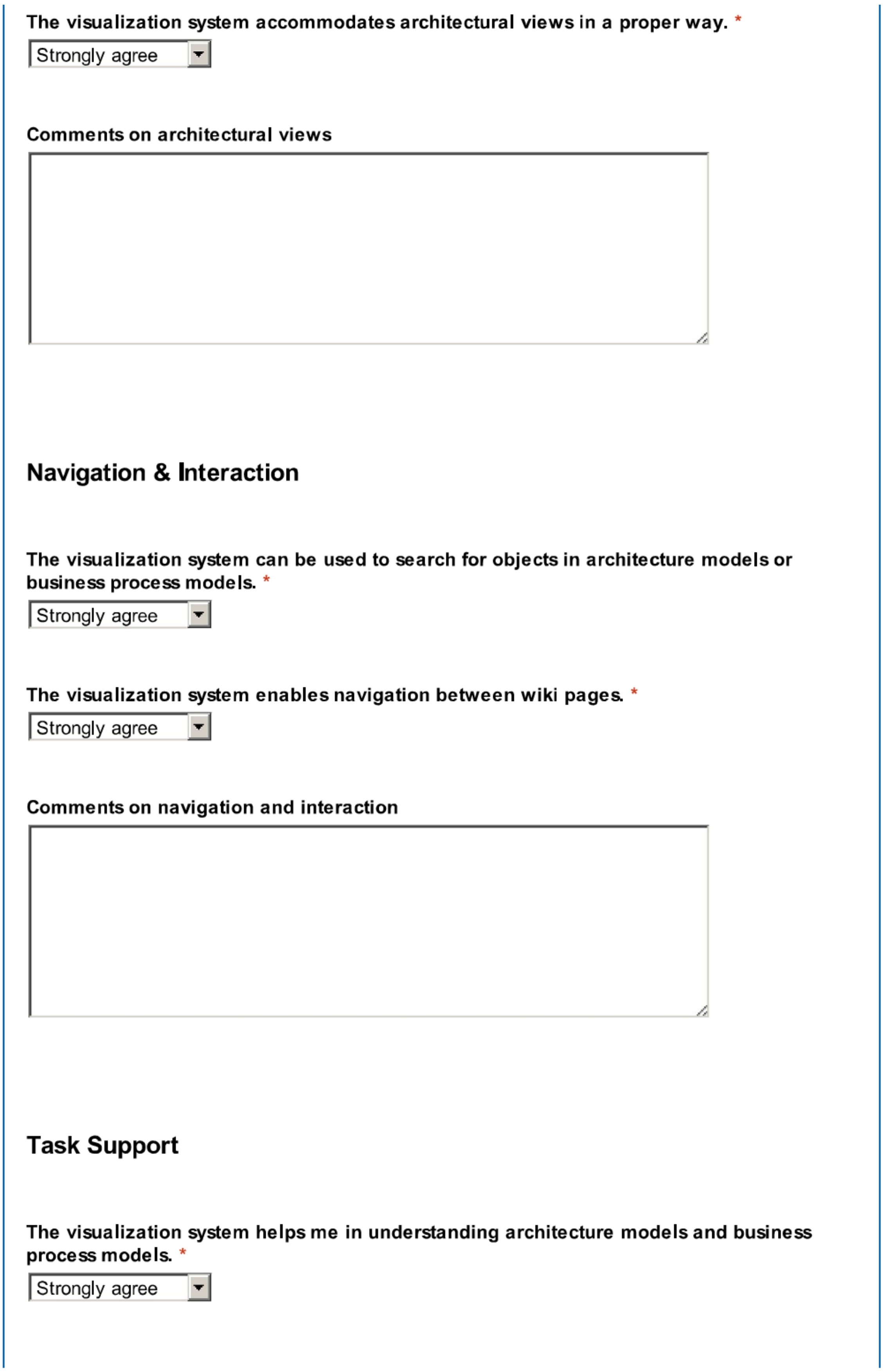
The visualization system supports both business process models and architecture models. *

Strongly agree

The visualization system supports image formats of business process models and architecture models. *

Strongly agree

Comments on representation



The visualization system accommodates architectural views in a proper way. *

Strongly agree

Comments on architectural views

Navigation & Interaction

The visualization system can be used to search for objects in architecture models or business process models. *

Strongly agree

The visualization system enables navigation between wiki pages. *

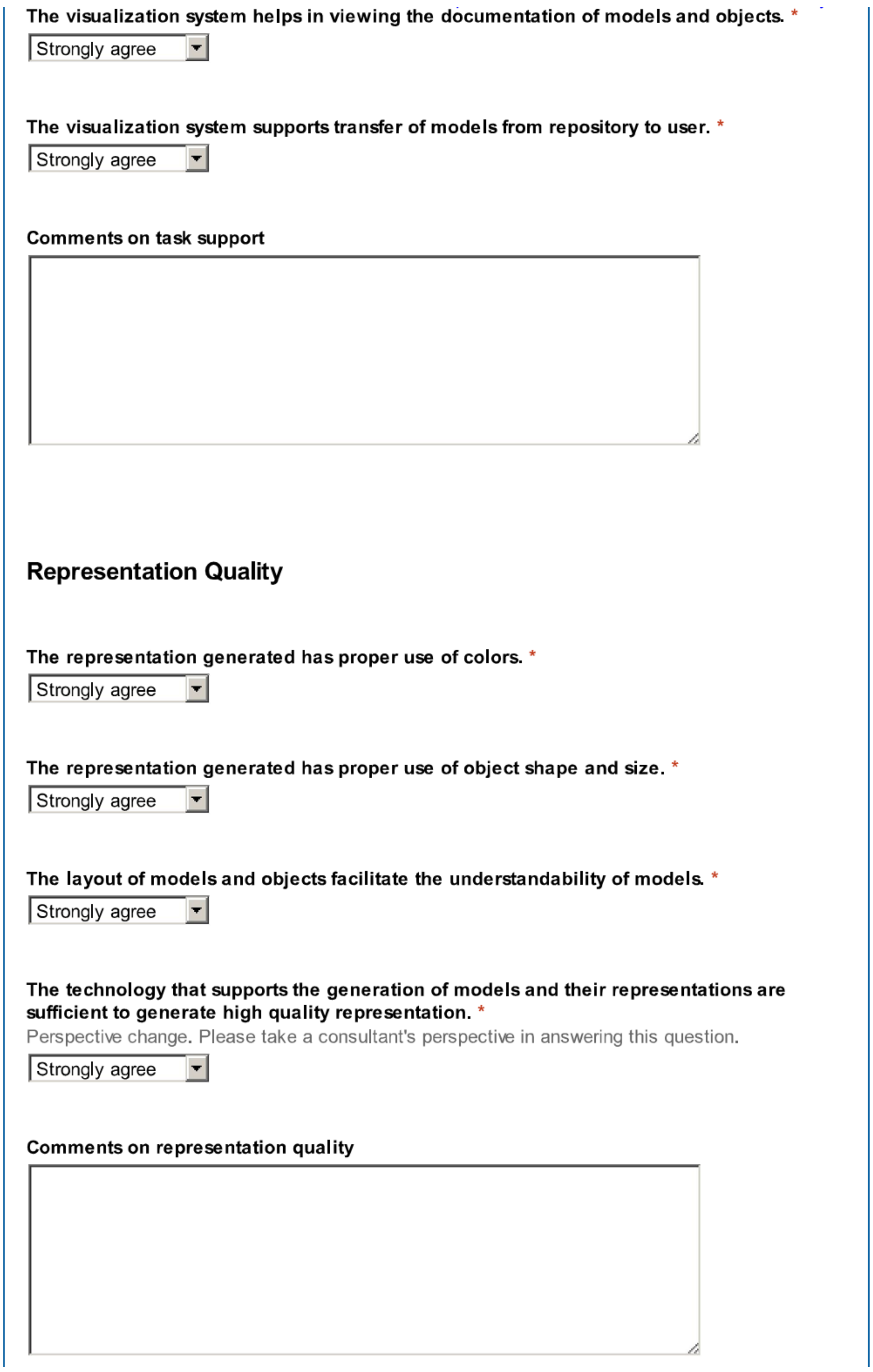
Strongly agree

Comments on navigation and interaction

Task Support

The visualization system helps me in understanding architecture models and business process models. *

Strongly agree



The visualization system helps in viewing the documentation of models and objects. *

Strongly agree

The visualization system supports transfer of models from repository to user. *

Strongly agree

Comments on task support

Representation Quality

The representation generated has proper use of colors. *

Strongly agree

The representation generated has proper use of object shape and size. *

Strongly agree

The layout of models and objects facilitate the understandability of models. *

Strongly agree

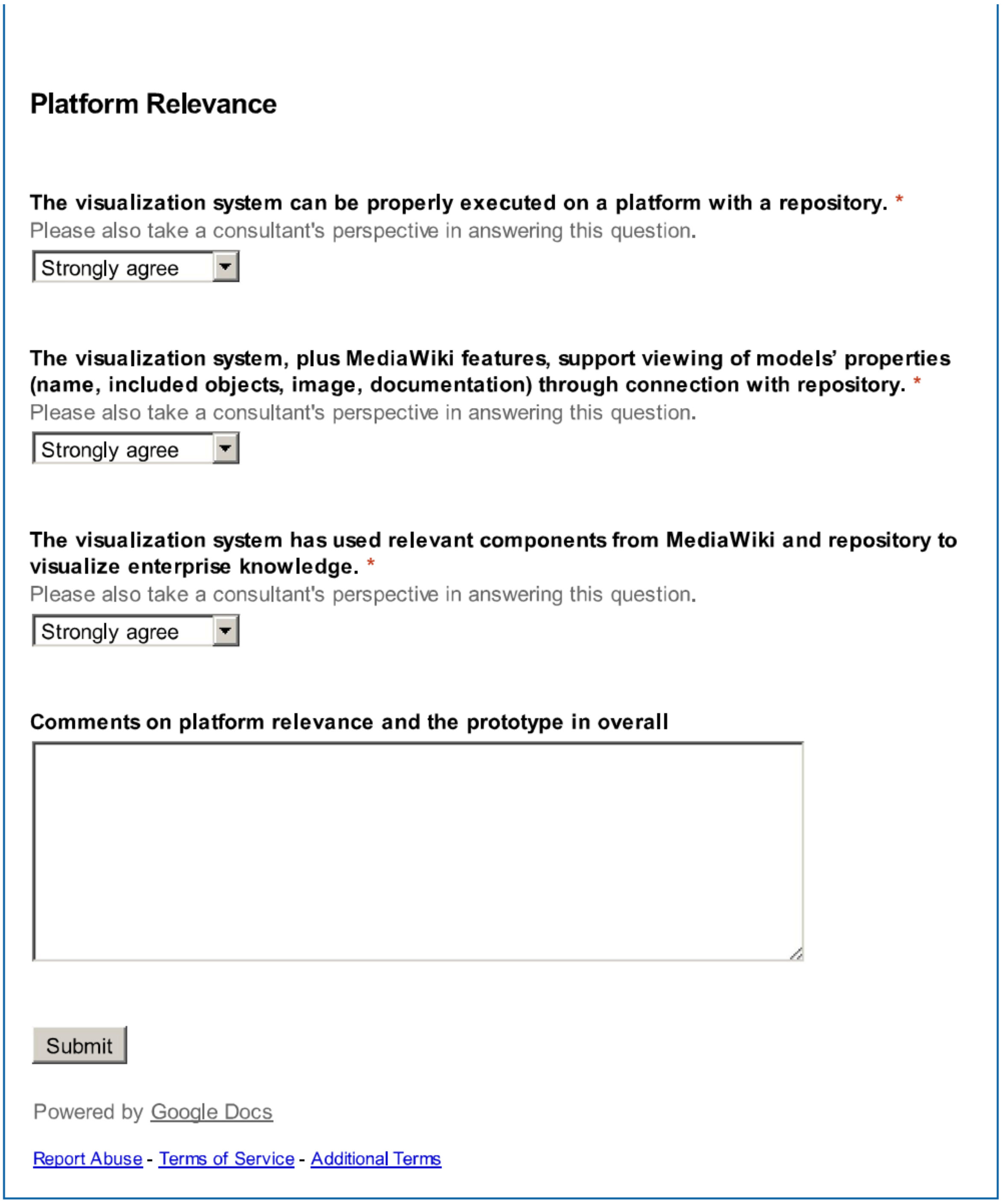
The technology that supports the generation of models and their representations are sufficient to generate high quality representation. *

Perspective change. Please take a consultant's perspective in answering this question.

Strongly agree

Comments on representation quality

BiZZdesign



Platform Relevance

The visualization system can be properly executed on a platform with a repository. *
Please also take a consultant's perspective in answering this question.
Strongly agree

The visualization system, plus MediaWiki features, support viewing of models' properties (name, included objects, image, documentation) through connection with repository. *
Please also take a consultant's perspective in answering this question.
Strongly agree

The visualization system has used relevant components from MediaWiki and repository to visualize enterprise knowledge. *
Please also take a consultant's perspective in answering this question.
Strongly agree

Comments on platform relevance and the prototype in overall