UNIVERSITY OF CALGARY

Ontology-based Retrieval of Software Engineering Experiences

by

Philip Nour

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

AUGUST, 2003

UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies for acceptance, a thesis entitled “Ontology-based Retrieval of Software Engineering Experiences” submitted by Philip Nour in partial fulfillment of the requirements for the degree of Master of Science.

_____

Dr. Frank Oliver Maurer, Supervisor, Chair, Department of Computer Science

_____

Dr. Guenther Ruhe, Department of Computer Science and Department of Electrical and Computer Engineering

_____

Dr. Gail Kopp, “Internal” External, Graduate Division of Education Research

_____

Date

Abstract

Organizations gain knowledge by carrying out more and more projects. Since internal knowledge originates from the organization itself, the suitability of this knowledge for reuse in prospected projects is high. This reason strongly motivates establishing an organizational knowledge management infrastructure. This study approaches ideas, techniques and feasibility of implementing a knowledge base of skills and processes. It focuses on an ontology-based approach and the effort required to initiate and maintain an ontological knowledge base.

The goal of the study is to examine the ontological approach. We compare it to another approach of information retrieval: text-based retrieval. As text-based retrieval is an easier to implement – yet efficient – retrieval system, we explain its capabilities and limitations as compared to the ontology-based approach.

We tested against a proposed development process and used existing efficiency measurements for both approaches. In spite of the high quality retrieval an ontology-based system delivers, results revealed the high effort required to build and maintain it. On the other hand, the results illustrate that good efficiency and low effort are required to build a text-based retrieval system. However, text-based retrieval is found to have limited capabilities in retrieving knowledge within certain contexts. Finally, we discuss the organizational perspective and factors affecting the choice of a specific approach.

TABLE OF CONTENTS

CHAPTER THREE: EXPLORATORY SETUP

CHAPTER FOUR: EXPLORATORY RESULTS

CHAPTER FIVE: CONCLUSIONS

TABLE OF FIGURES

Chapter 1

Introduction

1.1 Introduction

Software processes guide software development inside software development organizations. Over time, these organizations tend to tune their processes. In organizations that operate based on well-defined process models, engineers customize these models to fit their projects better. On the other hand, organizations with no solid definition of a process model tend to come up with their own processes, depending on their capabilities, type of work, and available resources.

As an organization grows, it gains and accumulates different kinds of experience. This experience varies in types; however, it still evolves around the process model and the different activities the organization usually adopts and carries out. In other words, the accumulating pieces of experience gained over time and through completed projects will likely be of a similar structure.

This, however, does not mean in any way that an organization should always commit itself to only one process model. Different models and approaches in software

development are emerging (e.g. Agile Methods). An organization needs to be capable of choosing between many models – according to their capabilities – based on the type of project currently under development. This will cause the type and structure of accumulating experience to differ as the organization grows. However, regardless of type, newly gained experience presents lessons for organizations to learn.

1.2 Learning Organizations

The term *Learning Software Organizations* (LSO) has evolved as a result of new ideas and disciplines introduced into software organizations [Ruhe 01]. Software organizations have introduced disciplines that do not aim directly at production of software, but towards the development of the software process itself, e.g. Experience Factory and knowledge management processes. These disciplines work side by side with the software development process and interact with it.

To utilize past experience, it is essential for software organizations to become learning organizations. This, however, does not imply the existence of a very specific discipline inside the organization. As organizations vary in their capabilities and resources, their ability to possess an infrastructure for implementing a learning loop will also vary. It is difficult to learn new lessons, analyze, and add them to the organization experience; it is also difficult to extract experience that is reusable in other projects. Achieving this goal inside an organization will connect outputs from previous projects to the inputs of prospective ones, establishing an organizational learning loop.

In the next chapter, we will be taking a closer look at disciplines and practices that evolve around establishing a learning loop inside an organization. Supported by ideas described in the next chapter, in further chapters we will also have a look at how skill and process experiences can be reused within the organization learning loop.

However, we have previously mentioned experience gained by the organization and how it may take different forms. There are many aspect of software development, and therefore, more than one type of lessons learned. In fact, experience can also originate from the organization itself, e.g., employees' skills. In the next section, we will talk about the types of experiences an organization can possess. This will represent the type of knowledge that organizations would be compiling as they grow. It will also lead us to know what type of experience we are retrieving when evaluating different types of knowledge retrieval systems, which is the goal of the study.

1.3 Types of Experience

1.3.1 Human Expertise

An important aspect of organization strength is the strength of its personnel. In this context, strength refers to qualifications and skills of employees working at that organization and, at the same time, participating in its growth. These employees have different skills that vary in type, profession, and origin. Skilled employees acquire their skills from many sources. Educational degrees, training, or previous work skills are all examples of different skill sources that employees can possess. We will touch more on employees' skills in Chapter 3 when discussing skill ontology.

1.3.2 Process-Related Experiences

Experience related to software development processes is all about the discovery and application of best practices. While each organization grows and carries out more projects, it is introduced to more than one area of development, newer types of tasks and most probably, newer technology. Throughout these projects, the organization adds to its experience new learned lessons on best practices for carrying out certain tasks. The

organization – through its people – learns these lessons from previous projects, regardless of their failure or success.

The ability of an organization to learn, store and re-use its experience is a matter of discussion. As there are many issues with experience representation, storage and retrieval, the application of such a strategy inside an organization is not a trivial task. This study will shine some light on these issues.

Organizations vary in their ability to introduce tasks or activities that do not lead directly to release of products. Unlike designing, building and testing, process improvement activities do not eventually produce customer deliverables. Instead, their goal is to enhance the development process through providing support for its workflow. On the other hand, process improvement activities consume different resources; therefore, its implementation inside the organization is subject to organizational pre-adoption of such resources.

Along with its growth process, the organization will have a more defined process in place. The process will define a solid base for the organization workflow of tasks. Process improvement activities can directly affect the workflow of the development process. This is normal since the goal of improvement activities is to enhance the process; hence, changes in the process model are expected. The maturity of the process model implemented inside the organization will determine its flexibility and ability to be changed. This reflects on the overall ability of the organization to incorporate new process improvement activities into its current working system.

In the previous discussion, the term *Process Improvement* was used generically. However, the interest of this study will focus more on specific aspects of process improvement. The study will focus on the improvement of retrieval of previous practices towards the intention of better using them in the future. As discussed previously, however, improvements come with a price, since establishing an infrastructure for such activities is resource-costly. Allocation of resources can take the form of employees,

tools, time, effort and the cost of changes on the process currently in effect. Depending on its resources and improvements it wishes to introduce, the organization itself should judge on its own ability towards initiating improvement processes.

Capturing experience is the key to implementing an experience base, as it directs the way towards methods of implementing storage and retrieval of experience. Upon capturing experience, identified pieces of knowledge are expressed in terms holding semantics related to a specific context. Storage is a relatively easy phase, if it provides means of retrieving this knowledge.

Previous process experience is usually of importance to people in charge of a project workflow, e.g., project managers or team leaders. Project managers are the best candidates to use experience retrieved from experience repositories, since they are in a position that allows them to utilize it. From a different perspective, experience obtained from experience bases provides input for managers making decisions; it meets an initial need.

Project managers will need to issue queries against the experience base related to many areas of development. For example, a manager would issue a query to find answers about best practices, available skills or tools for reuse in other projects. The answers provided by returned results demonstrate the value of an experience base system through the insight it provides for the project manager, directing him or her towards a better decision.

There are many aspects of how to better manage a project. Therefore, managers need to issue various types of questions against the experience base. These questions address specific information that is needed from previous projects, regarding either decisions or practices used at that time. Examples of information in experience bases that managers will be interested in retrieving are workflow information, timeframes, techniques used in certain tasks, outcome of a certain practice, etc.

In addition, human skills and experiences affect directly task assignments, quality and speed of development. Hence, such information also stands as a large part of the experience base, providing valuable information for managers when they assign responsibilities for new projects.

1.4 Knowledge Management

Knowledge is information combined with experience, context, interpretation and reflection [Ruhe 01]. As the name indicates, knowledge management (KM) is the discipline of managing the knowledge inside the organization for reusing it. Many technologies and techniques evolve around knowledge management, providing different approaches and solutions for various problems regarding managing knowledge inside organizations.

Experiences discussed earlier in this chapter are part of the organizational knowledge that the organization needs to manage[1]. Being a self-gained experience, this knowledge has a very high potential of reuse in emerging projects inside the same organization. This is because certain practices inside the same organization will not differ much among most projects. For example, the use of the same process model for similar projects, hence, practices within different projects will probably have many similarities.

Experience is concerned with context and interpretations. Therefore, it leaves the door open to project managers on how to make use of previous experiences. Previous practices are not necessarily best practices, as previous practices might result in poor outcomes. Thus, while knowledge management systems provide means of knowledge retrieval, they do not dictate a certain way of using it. One reason for this is that the ability to reuse knowledge in other tasks depends on many factors, including the following:

[1] In this chapter, the terms *knowledge* and *experience* are used interchangeably or specifically according to context needs. They are distinguished from each other in Chapter Two.

- Degree of similarity

  Similarity between different projects is the key for re-use. Similarity contributes, as an important factor, to which comparable practices are reusable. For example, a success story in using a certain tool can indicate another success story in using the same tool, based on the similarity between tasks using it.

  However, it is not easy to define similarity. No one project is identical to another. Factors like tools, model, technology, domain, goals, etc. all contribute in distinguishing one project from the other. Analyzing aspects of similarity between these factors among previous projects can direct managers as how to and when not to reuse previous practices.

- Availability of resources

  Allocation of the same resources used successfully in previous tasks might not always be possible. This can be because of concurrent projects the organization is carrying out or previous administrative decisions limiting resources for a certain project. Tools and personnel availability in addition to time constraints can affect the possibility to re-perform successful stories from previous projects.

- Outcome of stored experience

  Experiences are not always success stories. Failures or poor outcomes of performed tasks are part of the experience repository as proof of how not to do things. Project managers should be able to detect weaknesses in previous practices by identifying the factors leading to a failure or poor outcome. Elimination of such factors might lead to a successful reuse of these unsatisfactory practices.

- Human judgment

  Expert human intelligence will always be an important factor in deciding what practices fit best. An experienced project manager is capable of analyzing

conditions and inputs that lead to a decision on the suitability of applying previous practices. Tools to retrieve such practices support executive decisions; however, such tools never decide on the suitability of reusing them.

In the next chapter, we will discuss some aspects of knowledge management in more detail, and how they help in creating learning software organizations. Further chapters will focus on the ontological approach as a methodology of applying knowledge management practices to support the organizational learning loop.

We talked about the ability to reuse experience from experience repositories. However, there are many methods of retrieval of information from repositories depending on the information type. The goal of the study will finally focus on better ways of retrieving experience, based on the approach used to store it. In the next section, we will talk about types of approaches used in storing information or experiences. Before we discuss them in detail in Chapter 2, we want to explain how these approaches are used in storing experience and retrieving it. We will eventually explain how these approaches relate to each other and which ones will be given a closer look in this study.

1.5 Information Retrieval

1.5.1 Retrieval Fundamentals

While technologies differ, metrics used for measuring accuracy of information retrieval systems are based on similar concepts. To illustrate, let us assume a repository of stored information of a specific domain. A query issued against this repository will aim towards retrieving a portion of the stored information, i.e., relevant information. Ideally, the query should be able to distinguish between relevant and irrelevant information, returning only information relevant to the query issued. However, this is not the case in reality, as different retrieval systems have different capabilities to judge the relevancy of information. We will have a closer look at this issue in the next section.

Figure 1.1 explains on a conceptual level how queries distinguish relevant and irrelevant information from an information repository. When a query is issued against an information repository, it aims ideally towards returning relevant – and only relevant – information back. However, the query might not be able to judge accurately on the relevance of information, therefore returning irrelevant or incomplete results.

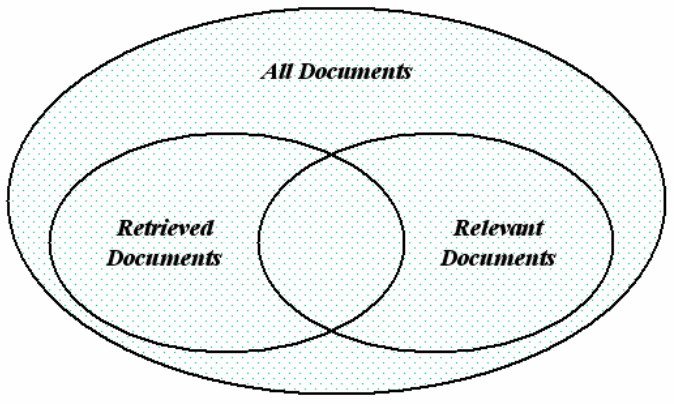

Figure 1.1: Retrieval of relevant and irrelevant information from repositories

Figure 1.1 shows a repository of documents where some of them are relevant for a specific query. The documents in the Retrieved Documents area represent what the query was able to retrieve. It shows that a query can do either of the following:

1. Return irrelevant documents, or
2. Overlook some relevant documents

An optimal query should make the Retrieved Documents area equal to the Relevant Documents area, i.e., retrieving only relevant documents. This case means a perfect *recall* and *precision* of a query. Next chapter, we will discuss recall and precision.

1.5.2 Information Retrieval Systems

Based on the knowledge domain, information exists in different structures and formats. This is because these information structures map to other types of information

structures in that domain. For this reason, various technologies also exist to manage storage and retrieval of information in a fitting manner. For example, databases provide an efficient way to store and retrieve related information. On the other hand, textual storage is an inexpensive approach when relations are of less importance.

Regardless of format and storage type, information is stored so it can be retrieved in the future upon demand. However, retrieval capabilities highly depend on the technology used in the storage of information. This is because each storage technology provides its own tools or languages for accessing, manipulating, and retrieving stored information.

The importance of relations between different pieces of information plays a major factor in the way information is stored and retrieved. In the following sections, we will have a look at four of types of information retrieval systems. Each has its own interpretation and understanding of the information it holds, in addition to specific capabilities for retrieving that information. The nature of each system reflects the way it was intended to be used. This is because some systems were developed to be used in knowledge-based systems, while others were meant for storage and retrieval of information. We will discuss these systems briefly, pointing out how these approaches relate the major approaches discussed in this study.

1.5.2.1 Databases

Databases are among the most common tools used nowadays for efficient storage and retrieval of information. The power of a database comes from its ability to store information while at the same time establishing and maintaining relations between its pieces. Relational databases are among the most popular kinds. A relational database stores information through mapping real-life entities into tables. Columns of a table map attributes of real-life entities, while rows of the same table map different instances.

Databases ensure they keep relations between tables valid through enforcing integrity constraints. Databases will not allow addition, modification or deletion of data if such action contradicts with the rules constraints. By performing this job on the database level, we relive higher-level applications from the need to check for data consistency.

Like all information retrieval systems, databases provide the means to access the data they hold. The Structured Query Language (SQL) has been developed primarily for use with relational databases. SQL is not a complete programming language; however, it has a limited vocabulary allowing access to the data inside databases. SQL provides the capabilities of retrieving (searching), adding, deleting and updating the data inside databases.

1.5.2.2 Textual Retrieval

Textual retrieval is a very basic form of information retrieval. While textual repositories are an inexpensive way of storing textual documents, they do not provide any type of relations between different pieces of information. Usually, textual repositories are made of a compilation of text-based documents.

Therefore, retrieval of information from textual repositories depends mainly on string matching. A query will simply be a string passed to a search engine that will perform a string match against the repository. To improve search quality, search engines can also use advanced techniques such as using synonym lists or searching for regular expressions. Textual retrieval is discussed in more detail in Chapter 2.

1.5.2.3 Case-Based Reasoning

Case-Based Reasoning (CBR) [Aamodt 94] is an approach that aims toward solving problems based on similar ones that occurred in the past. It captures situations

(cases) by storing their description and criteria in order to reuse this knowledge in similar future cases. This creates a solving approach that depends on accumulative knowledge gained through concrete occurrences of previous cases. A single case, in this manner, represents the structure of information that CBR uses in its similarity matching process.

The key to finding similar occurrences is to match the criteria surrounding each case. CBR recognizes a “problem case” as the main entity that has inputs (problem description) and gives an output (solution). A case with more inputs is better described than a case with few inputs. Similarly, a case that has an output (solution) is more useful than a case that does not. This is because future cases will need to take into account previous solutions of similar cases. We will discuss Case-Based Reasoning in more detail in Chapter 2.

1.5.2.4 Ontology-based Retrieval

Ontology is a way of describing a domain by defining the concepts of this domain and the relations between them. An ontology definition can also set attributes and rules of behavior for instances within the domain. In other words, an ontology definition provides an understanding of a domain by defining concepts, relations, attributes, and rules of behavior of instances of that domain. Inheritance is one of the main relations that ontologies support, allowing an IS-A relationship between instances.

Based on relations and attributes defined in the ontology description, users can query for pieces of information. Queries differ in syntax depending on the descriptive language used for laying out the ontology description. However, a query will perform a logical match between the relations and attributes specified in the query and those defined in the ontology. The inference engine will perform a logical search using instance attributes while applying rules wherever necessary in order to evaluate the query. Ontologies are discussed thoroughly in chapter 2.

1.5.3 Retrieval Approaches: A Comparison.

In the previous section, we overviewed some knowledge and information systems. Each has its own methodology of storage and retrieval of information. Their methodologies reflect the goals for which these systems were developed. Databases, for instance, aim towards storage of information while preserving relationships between different pieces. Other systems – like CBR – aim towards knowledge-based systems, storing all the criteria surrounding its problem cases.

While each system has its own way of storage, it also provides its own way of querying repositories. This way reflects the manner in which information was stored as the system makes use of any relations or rules that a repository holds about the information inside it. Figure 1.2 shows the basic features each of the systems poses.

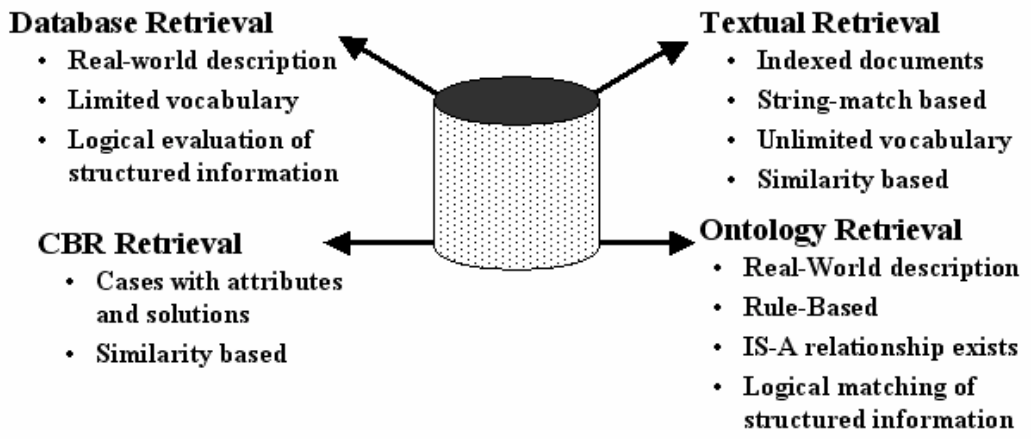

Figure 1.2: Basic features of some knowledge and information retrieval systems

Textual storage of information is considered the simplest of all, because textual documents do not have the capability to relate with each other. Therefore, in order for search engines to be able to search textual repositories, they index the documents. Indexing of documents will identify keywords and the documents they exist in; this improves search performance.

However, search engines use techniques other than exact string match to improve search quality. Search engines can search using regular expressions or use synonym lists of keywords provided by the user. Therefore, it is apparent that textual queries use unlimited vocabulary for searching. This is because users can use arbitrary words in order to find relevant information.

Databases, on the other hand, provide relations between different pieces of information. Queries can make use of these relations defined on the database level to retrieve what they believe to be relevant information. Relevancy of information is more visible to the user since database schemas reflect a real-life situation where relations are more obvious.

Relevancy in the previous context refers to relationships that give the information stored a real-life meaning. For example, relational databases map entities to table and create foreign keys to associate entities to each other. Object oriented databases also reserve relationships by storing data as objects. In fact, Seaman went to the extent of using relational databases as a backend in knowledge-based systems [Seaman 01]. This was done by mapping pieces of knowledge to tables, and mapping knowledge attributes to the columns of these tables.

Although SQL has proven to be a powerful tool on the database level, it has a limited vocabulary for constructing queries. String matching is not the best way to find information in databases; however, it is practical to a certain limit. While text searches depend on string matching directly, databases are more concerned with relations.

Case-Based Reasoning is used more in knowledge-based systems. It provides a retrieval mechanism that depends on similarity between cases. The CBR engine examines all cases and matches them with the problem case in order to find a solution. This approach is different from what we have seen in text or database retrieval. A form of artificial intelligence takes a role in determining similarity rather than exact matching of text, for example.

Similar to CBR is the ontological approach. Both approaches assign attributes to instances (cases in CBR), and these attributes provide means for the query to find an answer. However, ontologies go further in their ability to define rules of behavior for their instances. The inference engine can use this to evaluate the query against instances defined in the ontology. Such methodology allows simpler queries, and a more efficient way of storage.

Ontologies also support inheritance relationship between instances, i.e., an IS-A relationship. Unlike other approaches, this is an advantageous feature as it gives the user more flexibility in asking questions (querying). Such a relationship allows the user to query for instances by querying for their super types. This is useful in cases where the user needs any instance of a certain concept or sub-concept.

1.6 Goal of the Study

We have looked at four different knowledge and information retrieval systems. Each has its own infrastructure, technology, and capability, depending on the purpose it is meant to serve. On the one hand, textual systems are simple and easy to build and use. On the other hand, databases are a higher form of information retrieval as they make use of relationships. Advancements in technology have produced systems like CBR for use in knowledge-based systems. However, ontologies extend the capabilities of CBR with their logical inference capabilities and their ability to create IS-A relationships between knowledge pieces.

All these systems provide different capabilities of information and knowledge retrieval. Therefore, we want to examine their ability to provide quality retrieval of knowledge within the software development domain. The goal of the study is to examine the two ends of the four systems: textual retrieval against ontological retrieval. While we initially hypothesize textual retrieval to be the easiest to establish, we know that the ontology approach requires more effort to build. At the same time, there are some

limitations in the textual retrieval approach, whereas ontologies can handle logical inferences and enforce rules.

Through an exploratory case study, we will explore the capabilities and efficiency of each approach. The study will also explore the effort and resources that accompany the process of building solutions based on each technology. We want to examine the ease and quality that textual retrieval provides against the quality and effort of an ontology-based solution. Comparison results of the exploratory case study will aim towards determining the quality and accuracy of our previous hypothesis regarding the effort and efficiency of each approach.

1.7 Summary

In this chapter, we have touched on knowledge inside software development organizations. We also shed some light on the importance of keeping organizational knowledge for reusing it in the future. We have introduced a number of information and knowledge retrieval systems with various storage and querying capabilities. We have identified textual retrieval and ontological retrieval as the two ends on the complexity scale. In this study, we need to examine the two approaches in order to compare the effort of initiating an ontological solution to the quality of textual retrieval. In the next chapter, we will discuss the methodologies, disciplines, and concepts behind the study.

Chapter 2

Literature Survey

2.1 Introduction

In the previous chapter, we introduced basic ideas and concepts related to experience inside organizations. We also discussed experience from an organizational perspective, focusing mainly on human and process experiences. In addition, we introduced basic ideas regarding learning software organizations and the concept of learning loops inside organizations. Issues related to knowledge acquisition, storage and maintenance were also highlighted.

In this chapter, we will focus on disciplines, methods, and technologies that play roles in organizational knowledge management processes. We will discuss each in relation to this study only, and explain their respective roles within the organizational learning loop. However, the next section will briefly explain what a normal software development process is. It is important to know and distinguish the normal development process from any other processes or process activities inside organizations, e.g., knowledge management process and activities.

2.2 Software Processes

The software process is known to be a set of activities, methods, and practices used in the production and evolution of software [Humphrey 89]. These activities can start from analysis and requirements engineering, go through building and testing and ending with maintenance and support. A software development process is made up of different sub-processes, each of which is carried out with different resources carrying out different duties with different tools. A software process is the ongoing procedure of making software with all its resources and subtasks.

Resources allocated for software processes can be of multiple types: human (employees), tools, equipment, funding, and time availability. These resources take different roles in the development of software over different sub-processes of development. The structure and order of sub-processes taking place within the main development process is called the *model* of that process. Each organization chooses a known or a tailored model and uses it to carry out its projects.

The Capability Maturity Model (CMM) [CMMI 02] is one of the most recognized frameworks defining the maturity of a software development process inside an organization. It does not force certain activities in a specific way, but introduces a framework of a software process, allowing different organizations to apply it in its own way. Commitment of each organization toward the framework proposed decides which CMM level that organization fits in.

Next, we will discuss the experience factory, which is a discipline that can co-exist inside organizations along with normal software processes. With the understanding that software development is the major activity for the organization, experience factory activities run in parallel with the software process activities. The two types of activities make use of each other.

2.3 The Experience Factory

The idea behind the experience factory comes from the need to keep experience within organizations and not lose it upon the loss of experienced resources. Victor Basili has introduced the experience factory concept and defined it as a "logical and physical organization, that its activities are independent from the ones of the development organization" [Basili 94].

Basili introduces the experience factory as an organizational structure responsible for building and maintaining experience repositories. An organization will do the development, giving its input about the process characteristics to the experience factory organization. In return, the experience factory organization will process their input, store it, and give feedback to the development organization with improvements on the process. Suggested improvements will be based on similarities with previous projects carried out, making use of experience gained recently.

The goal of the experience factory is to create a learning loop inside the organization. The loop allows retrieval of experience gained in the past, and use of it to support on-going projects based on their similarities.

Figure 2.1 The learning loop

Fig 2.1 shows a proposed learning loop. It starts with project planning, where all tasks within the process are defined. The process continues with a normal workflow, and experience is gained. To allow future retrieval, this experience has to be packaged and stored. Packaged experience

represents the experience base of the organization. Project planners access the experience base in future to make use of it in current projects.

2.3.1 Packaging Experience

A cornerstone of the experience factory discipline is the packaging of the experience itself. Based on Basili's work, Fig 2.2 shows how the different phases of packaging experience fit together. The diagram present the two sections of an organization:

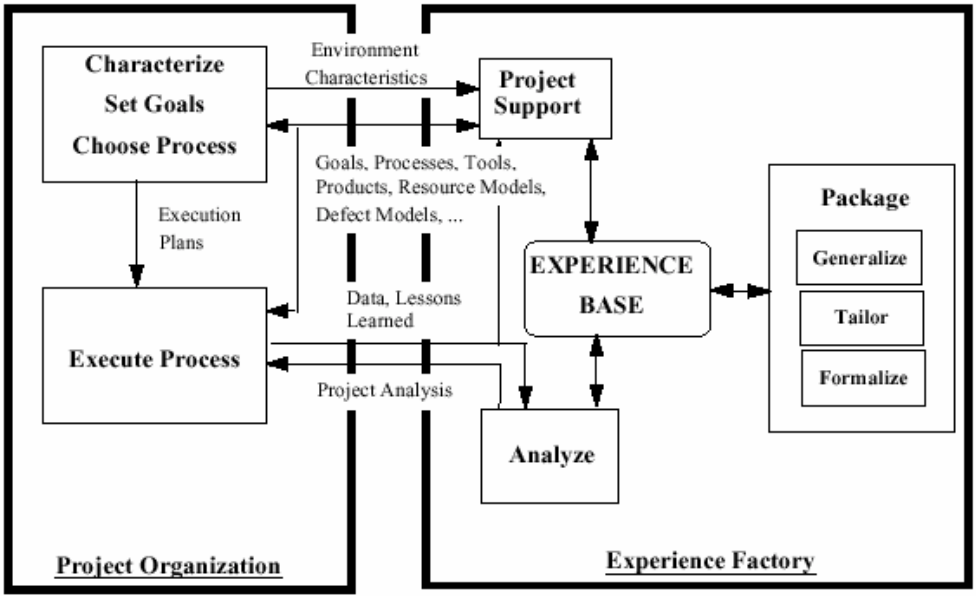

Figure 2.2: Packaging experience [2]

1. Project organization: This is where characterizing, setting goals, and choosing the process of the project take place. At this point, the organization itself - away from the experience factory - understands the process it will execute to carry out against a certain project. The actual execution of the process (building the project) takes place inside this section.

[2] The Experience Factory [Basili 94].

2. Experience Factory: This is where the process of building the experience base happens. On this side resides the experience base itself, accessible by the project support personnel. Conceptually, packaging experience and tailoring it according to needs take place at this side. However, these steps (packaging and tailoring) can take many shapes depending on the methods of packaging and technology used to represent knowledge. We will cover this in more detail when talking about representing knowledge.

The experience factory does not imply technical details for implementing the experience factory process. Packaging depends on the knowledge representation used, and so do other aspects. For example, organizations can have their own planning tools that can access the knowledge base and carry out their project support. Similarly, formalizing experience before making it a part of the knowledge base can also take many forms, as we will see in upcoming sections.

Upon obtaining knowledge, the experience factory aims to use it in supporting projects and projects planning. However, knowledge retrieved might not be concrete steps describing steps of operation, as it only describes previous practices. Therefore, stored experiences might need to be tailored to serve current needs.

The next section (Ontology), along with the experience factory concept will layout a good base for our discussion further on.

2.4 Ontology

2.4.1 What is Ontology?

Linguistically - and as a branch of metaphysics - ontology is the study of existence and the understanding of the nature of being. The prefix *onto* means “being” or “existence”; while *logy* is the suffix that means “the study of” - it comes from the base

logos that means “to reason”. Ontology is the science that deals with the nature of things and tries to understand its principles and relations.

Ontology is also defined as an explicit specification of a conceptualization [Gruber 93-B]. In other words, ontology is a formal method of describing concepts and their relationships within a certain domain. Ontology defines concepts by describing their attributes and relationships to other concepts, and the rules governing their behavior.

Another definition of ontologies is as a vocabulary of terms (names of relations, functions, individuals), defined in a form that is both human and machine-readable [Gruber 92]. This vocabulary is a specification that determines the number of terms, relationships, and assertions contained within a single ontology. The finite amount of vocabulary helps different agents to agree on a unified ontological description of their domain of interest.

Ontological concepts do not necessarily stand for tangible objects, but they represent distinguished concepts. Thus, ontologies can represent their concept in various forms. Concepts used in ontological descriptions describe the domain in an abstract way; however, concepts themselves may not be in their simplest form. Ontologies introduce concepts on a level allowing them to be instantiated and used to reflect domain objects.

Given that ontology is a specification, it establishes a formal way for describing entities of the surrounding world. Looking at ontologies from this perspective takes us one step forward towards thinking of ontological knowledge bases that describe a certain domain. The domain under discussion will consist of specific concepts, relations and rules that allow no understanding of concepts beyond what is defined in the ontology. Ontological statements of a logical theory of this world can be represented in a machine readable form, making other forms of higher reasoning systems capable of understanding represented domains [Gruber 92].

2.4.2 Ontology as a Commitment and Agreement

Ontology describes its domain on an abstract level. However, different ontological descriptions of the same domain might differ. Differences can exist due to using different names of concepts, different number of rules or different relations. This can occur as different perspectives, focuses, and interests exist towards a certain domain. It is important to have a commitment to a specific ontology in order to unify how different applications understand the described domain. Systems operating on a higher level can possibly achieve different results upon using different ontologies. This depends largely on the number and types of relations and rules that semantically tie different concepts together.

A commitment to a specific ontology is important for implementing ontology-based sharable systems, i.e., systems using a common ontology between different agents. For ontology-based systems, a unified ontology of a certain domain is essential, as it is an agreement between all agents on the meaning of concepts and their relations. In other words, while ontologies preserve consistency within them, different higher level systems operate based on a common understanding of domain ontology.

On the other hand, because ontologies are consistent within themselves, they may complement each other if merged into a bigger ontology. However, this is true only in theory, since this is possible only if no two terms will eventually refer to the same concept. Otherwise, this will cause redundancy in concepts and relations. Recognizing these conflicts and correcting them among different ontologies might require human intervention.

The necessity for common ontologies emerges from the need to exchange knowledge. Ontology capabilities of representing concepts and relationships provide a higher level of representation of a certain domain than raw data. Information retrieved based on the ontology semantics will represent pieces of knowledge. Ontological

agreements can eventually participate in establishing shared knowledge bases used in distributed working environments.

A knowledge base is a collection of knowledge gained through previous experience and packaged into one repository. A knowledge base allows the storage of knowledge using a form of representation, and the retrieval of that knowledge through an infinite number of queries. Ontology is one way of representing pieces of knowledge within the knowledge base, with the intention of manipulating this knowledge using a higher level of reasoning systems.

Through the study, we will examine how ontology works and the role it plays in a knowledge base retrieval system. In the next section, we will also see how knowledge representation languages can represent ontological definitions.

2.5 Knowledge Management

In the earlier days of computing, the term *raw data* was used to indicate un-processed data. Similarly, *information* was the term used to indicate processed data carrying meaningful and usable values. With the development of information technologies, a need emerged for the ability to process a higher form of information: knowledge. Knowledge is information combined with experience, context, and reflection [Ruhe 01].

With the rise of the information age, the need to exchange knowledge surfaced. In order to exchange knowledge, different parties need to agree on a common understanding of the shared knowledge, i.e., a shared ontology. This agreement provides guidelines for future storage and interchange of knowledge.

Nevertheless, for a knowledge-based system to operate, knowledge needs to be in a machine-readable form. For example, while ontology defines concepts and relations, it

does not enforce a specific representation language. Therefore, Knowledge based systems need to read knowledge in formats that satisfy the real meaning of the concepts, rules, and relations making this knowledge.

In the next section, we will introduce some of the languages that are used in representing ontologies. These languages allow the exchange of knowledge between different agents as they provide a common syntax for reading ontologies.

2.5.1 Ontology Definition Languages

2.5.1.1 F-Logic

Frame Logic (F-Logic) is an object-oriented, frame-based language. F-Logic is powerful in its capabilities of declaring classes and defining their attributes. F-Logic notations are based on logical notations, and have a complete resolution-based proof theory.

F-Logic has features of object-oriented languages, such as inheritance, complex objects and encapsulation. These features have direct representation in F-Logic [Kifer 95]. Class definitions and IS-A relationships are directly represented in F-Logic.

It is possible in F-Logic to define instances of certain classes on the fly. Instances can have any attributes with assigned values upon declaration. F-Logic is also capable of defining rules that allow setting values or defining relations for all instances referenced by these rules.

The object-oriented features of F-Logic along with its logical implication capabilities make F-Logic a suitable language for representing ontologies. In F-Logic, ontological concepts map to objects and attributes map to fields' values. Relationships between concepts correspond to relationships between objects, e.g., inheritance. F-Logic

inference capabilities and flexibility allow the user to define concepts on the fly and to combine statements into one in order to define attributes and assign values.

2.5.1.2 KIF

The development of the Knowledge Interchange Format (KIF) came through a project sponsored by DARPA. KIF was developed to be a standard format to exchange knowledge between different agents. The syntax of KIF has logic-like expressions, and is of a first-order predicate calculus. KIF has gained popularity due to its solid specification and declarative capabilities [Genesereth 92].

KIF recognizes objects as it considers objects to be instances of defined concepts. From a KIF point of view, the *universe of discourse* is the set of all objects presumed or hypothesized to exist in the world [Fikes 92].

KIF also recognizes *functions* and *relations*. Both functions and relations are defined as sets of lists of objects. Functions take sets of objects as parameters, and associate these objects to another unique object, which represents the returned value of that function. On the other hand, relations combine objects in a list where objects jointly satisfy that specific relation [Fikes 92].

The definitions of KIF are highly detailed and expressive [Fikes 92]. This and other certain features rank KIF high among knowledge exchange formats. These features in KIF are the declarative semantics and comprehensiveness of the universe of discourse. Such capability corresponds to the ideas discussed in the ontological description of domains, where ontology defines concepts and relationships.

2.5.1.3 Ontolingua

Formally, Ontolingua is a mechanism for writing ontologies in a canonical format, such that they can be easily translated into a variety of representation and reasoning systems [Gruber 92]. Ontolingua is available as a tool for writing and editing ontologies. Being a mechanism, Ontolingua supports writing ontologies in a way that allows upper-layer systems to process its representation and use it to produce a more meaningful output.

Ontolingua uses KIF as a basic syntax. However, Ontolingua has developed extensions aiming towards supporting the portability of ontologies. Since it is a translation mechanism, Ontolingua also supports the conventions commonly used in knowledge representation, e.g., inheritance.

Since Ontolingua is based on KIF, it takes advantage of KIF's powerful inference capabilities. Like KIF, Ontolingua recognizes classes, functions and relations. In addition, Ontolingua extensions are dedicated towards writing Ontologies for exchange between higher reasoning systems. This representation scheme offers a standardized information exchange between multiple agents, based on ontological descriptions.

2.5.1.4 OIL

The Ontology Inference Layer (OIL) is a joint work towards a standard for specifying and exchanging ontologies [Harrock 00]. OIL is based on three roots, benefiting from the capabilities and power of each of these roots. OIL is based on description logic, frame-based systems, and web standards. These three roots empower OIL as they provide it with their strengths. At the same time OIL expands their capabilities to provide a better semantic representation of knowledge.

Description logic usually provides formal semantics and efficient reasoning. They are capable of describing knowledge by defining the semantics of the objects and their relationships. OIL inherits from description logic its formal semantics and efficient reasoning support [Harrock 00]. As for frame-based systems, they provide the definition of frames (concepts) within a particular domain. Web standards that OIL makes use of are the Extensible Markup Language (XML) and the Resource Description Framework (RDF). Both have been standardized by the World Wide Web Consortium (W3C). They provide the syntax for OIL that allows the exchange of represented knowledge.

OIL tries to avoid some of the restrictions found in other languages. For example, the authors of OIL point to the high level expressive power of Ontolingua that is provided with no means of controlling it [Harrock 00]. Similarly, OIL adds its own extensions to the standards discussed earlier, like RDF. These extensions allow OIL to better represent its ontology in RDF documents.

Until now, we have discussed languages that are used to represent ontologies in a machine-readable form. However, in the next two sections, we will discuss two technologies that relate to knowledge management and retrieval of experiences. These two approaches are examples of larger systems that aim towards retrieval of previous experience. They extend the concept of representing experience in standard languages in order to be processed later. In fact, they provide a complete system for storage and retrieval of experiences.

We discussed CBR earlier from an information retrieval perspective. However, in the next section we will look at it from a knowledge management perspective. Similarly, we will talk about the BORE system, which is a knowledge management tool that addresses packaging and reusing of experience inside software development organizations.

2.5.2 Case-Based Reasoning

Case-Based Reasoning (CBR) is an approach that has been around for almost a decade. It started as a research project with artificial intelligence (AI) roots. The initial CBR work was based on AI work done by Roger Schank [Schank 82]. In his brief discussion about the history of CBR, Agnar Aamodt [Aamodt 94] also mentions the study of analogical reasoning done by Gentner [Gentner 83] as one base of CBR development. CBR developed later and was more associated with the use and development of expert systems.

Case-Based Reasoning is a problem-solving paradigm [Aamodt 94]. It is an approach towards solving problems based on similarity with previous occurrences of problems. For example, Case-Based Reasoning approaches a solution for a certain situation by remembering a previous situation and the methods used to solve it. However, CBR is a learning system that can make use of previous experiences. If a certain case was approached previously but could not be solved, the reasons behind the failure are identified and remembered so that they can be avoided in the future.

The previous overview showed that CBR is a cyclic process aimed toward solving new problems based on learning from previous ones. “Case-Based Reasoning favours learning from experience, since it is usually easier to learn by retaining a concrete problem solving experience, than to generalize from it.” [Aamodt 94]. In other words, CBR finds a more solid reference when a solution is given based on an actual occurrence than on sets of predefined rules. However, such rules still exist as they govern the structure of experience that is extracted from given cases.

A closer look at CBR allows us to see a certain degree of similarity with ontologies. CBR recognizes cases as instances of previous occurrences of situations. These situations have their own criteria, in other words, inputs, outputs and attributes. In a manner similar to the means used to find attributes in ontological instances, the CBR

engine uses similarity of criteria between different cases in order to find similar cases. Upon retrieving similar cases, CBR can propose a solution to the new case.

2.5.3 The BORE System

According to Henninger, BORE (Building an Organizational Repository of Experiences) "is a prototype tool designed to further explore and refine the requirements for tools supporting experience-based approaches" [Henninger 03]. As the name indicates, the BORE tool aims towards reusing organizational experience through packaging it in experience repositories. The concepts behind BORE extend on the concepts of the experience factory, eventually aiming towards support of software processes. Extensions to the experience factory are done through rule-based process tailoring, support for process modeling and enactment, and case-based organizational learning facilities [Henninger 03].

The basic element inside the BORE tool is a *case*. A case represents the basic element that holds information about an event or activity. Since the system is concerned with software development, cases in this context refer to project tasks. Each case holds all related information and metrics about its task, while the BORE repository represents a compilation of these cases. The BORE tool itself (which is web-enabled and java-based) allows the setting of all information and metrics of tasks.

Three main features exist and relate directly to the information contained in each case: time reports, task dependency, and task history. These three core features provide extra information about cases in the repository. By using information provided by these features, BORE can provide a number of different views of the experience contained in the repository. For example, task dependency information allows the BORE to create Gantt charts[3] for projects in the repository. The features in BORE are designed to

[3] Gantt charts are used in project planning to represent the timing of tasks required to complete a project.

integrate with project management and task enactment to capture the experience necessary to implement the experience factory methodology [Henninger 03].

In the first chapter, we talked about knowledge and storing it in repositories. Now that we have looked at these knowledge systems and knowledge representation languages, it is important to know how effective they are in experience retrieval. This is because retrieval efficiency demonstrates the power of these technologies to provide feedback. We discussed this in the first Chapter; however, we need to go into more detail, as the goal of the study will include a comparison between two approaches. The next section will explain the measurements that we will use in the comparison.

2.6 Information Retrieval

Data, information, and knowledge are three distinct concepts. *Data* is the term used to indicate pieces of raw data that are non-informative by themselves but need to be processed. *Information* is processed data, where a certain degree of meaningful information exists. On the other hand, *knowledge* represents a higher level of analyzed information and understood within a certain context and interpretation [Ruhe 01].

The necessity for information retrieval arose decades ago. It is becoming a more demanding field because of the increasing amount of information made publicly accessible on a daily basis. Therefore, many information retrieval systems based on different technologies exist nowadays. All these systems aim towards facilitating access to the massively growing repositories of information. Web search engines are examples of information retrieval engines that have to provide high efficiency to serve the increasing size of information on the Internet.

Similarly, databases stand as popular, yet efficient information retrieval systems. Although they differ in types and technology, e.g. relational versus object oriented, they still serve the same purpose. The structure of the data to be stored can influence the

decision as to what type of database to use. Whatever the type is, the capabilities of databases allow efficient storage, manipulation and retrieval of data.

Other forms of information retrieval systems depend on simple string search algorithms. These types of information retrieval systems perform text searches over large portions of plain text. This approach is one of the simplest forms of information retrieval to implement. However, this approach does not recognize the existence of relations between pieces of information.

With the rapid expansion of the Internet in the last two decades, search engines introduced document retrieval based on textual searches. The engine performs a search over a large number of indexed documents, returning a list of documents matching the query. This method uses string matching of given keywords against words inside searched documents. Nevertheless, the following techniques, among others, enhance text searches capabilities and performance:

- Use of synonym lists. This technique adds some semantics to the raw text search, allowing possible retrieval of more related documents. The synonym list provides alternative keywords for the ones used in queries and uses them in the search assuming they mean the same thing. For example, if a search engine has “Developer” as a synonym for “Programmer”, then a search using any of the two words should return documents that contain any of the two words.

- Use of certain file types special features. Such features can provide extra information on the content of the document itself. For example, web pages – which are textual HTML files – can contain special tags understood by the search engine, providing metadata about parts of the document. For example:

    - Key words tags in html files, used for the search engine to indicate the type of content of the document
    - Documents titles tags

- HTML alt tags, textually describing images on web pages. This technique allows search engines to retrieve a page if it contains a related image.
- XML or RDF tags, providing information about data structures and content type and format. These tags work as directives for search engines.

2.6.1 Retrieval Measurements

Different techniques on retrieving information result in different types of information retrieved. For example, relational databases allow retrieval of records based on the evaluation of a logical condition between values in columns and given values in the SQL statement. An advantage of this is the ability to retrieve a complete record or a list of records using a limited number of given values. Database designers express semantics through relations with other records using foreign keys on the database level.

Precision and *recall* are two measurements defined to point out the accuracy of searches. They evaluate the success of a query depending on the ratios between the number of retrieved documents, the number of relevant documents, and the number of irrelevant documents.

Fig 2.3 illustrates a repository upon issue of a specific query. It shows a subset of a repository where documents are relevant to that query. However, retrieved documents can contain some irrelevant documents, in addition to some relevant ones. This illustration should help us understand the concepts of precision and recall of information.

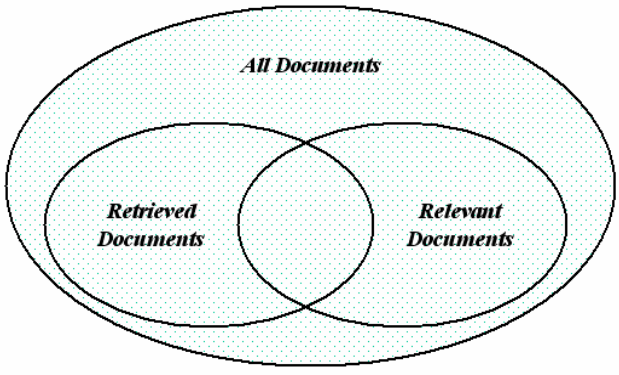

Figure 2.3 Retrieved and relevant sections of a repository upon query issue

Recall is the proportion of relevant documents in a collection that are retrieved in a given search [Harter 86]. In other words, recall is the percent of relevant documents retrieved from a given search in respect to all the relevant documents that were supposed to be retrieved from the search repository.

$$Recall = \frac{relevant\ documents\ retrieved}{all\ relevant\ documents}$$

When the search retrieves all relevant documents then recall has a value of 1, indicating a completely successful query. If there any missing documents in the search result, recall ratio becomes less than one. Recall can be as low as zero; in that case, we say that our query failed to retrieve any of the relevant documents from the search repository.

Precision, on the other hand, is the ratio of relevant documents retrieved in respect to the number of all retrieved documents [Harter 86]. This measurement points out the accuracy of the query in retrieving only relevant documents. It shows the percent of accurate (relevant) hits against all hits made by that query.

$$Precision = \frac{relevant\ documents\ retrieved}{all\ retrieved\ documents}$$

When the number of relevant documents retrieved is equal to the number of retrieved documents, this implies perfect accuracy of the query. Such a case indicates that the query was successful in distinguishing relevant documents, and that what the query retrieved was, in fact, relevant. A complete failure will make precision equal to zero, indicating that none of the retrieved documents were relevant.

To complete the picture, there are two more measurements defined for measuring information retrieval effectiveness: s*pecificity* and *fallout*. Specificity measures if a system is able to reject non-relevant documents [Pao 89]. Fallout, on the other hand, is a

measurement for false drops, i.e., not retrieving relevant documents [Pao 89]. Specificity and fallout are defined as follows:

$$Specificity = \frac{Non\text{-}relevant\ documents\ not\ retrieved}{Non\text{-}relevant\ documents}$$

$$Fallout = \frac{Non\text{-}relevant\ documents\ retrieved}{Non\text{-}relevant\ documents}$$

According to Pao, "both specificity and fallout are fairly insensitive measures when the retrieved set is quite small in comparison with large files"[Pao 89]. However, we will be using specificity in an indirect way to calculate the overall efficiency of a certain query.

Based on previous measurements, the *efficiency* of a system in retrieving relevant documents is defined as follows [Pao 89]:

$$Efficiency = Recall + Specificity - 1$$

Efficiency will take values between –1 and 1, where 1 is the best and –1 is the worst efficiency. In other words, by looking at the formulas we can say that a query achieves best efficiency when it retrieves all relevant documents, and does not retrieve any irrelevant ones.

2.6.2 Efficiency of None-Text-Based Retrieval Systems

We have introduced recall and precision as the two major measurements we are using to calculate the efficiency of our retrieval. However, in text-based searches, queries are usually based on key words, i.e., the search for string matches. Even with the use of synonym lists and other techniques, the search for key words inside textual files is still the base idea of search. Upon finding a hit, the search engine will retrieve the document

containing the keywords. Recall and precision fit within this scenario perfectly as they provide a good indication on the efficiency of the search.

Searches based on logical predefined relationships get, in fact, a recall and precision equal to 1, i.e., retrieving all relevant records in a database. Since pieces of data in the database are related, search engines will be able to retrieve relevant records, if the query is correct. This scenario, however, assumes that the database design and the data itself are correct and reflect a real-life situation.

Similarly, ontological inference engines can achieve an optimal recall value when executing a logical query. This is because ontological pieces of knowledge relate to each other through relationships and rules. If a user were capable of structuring a well-defined query that accurately describes what is needed, recall of the query would be equal to 1. Again, this scenario assumes an ideal world where queries are well-written and the ontology modeling of the domain is complete.

Although intelligent search systems provide better quality results, they also require more effort to build. An ontology-based system, for example, will require building ontologies, modeling knowledge, packaging experience, building retrieval tools…etc. On the contrary, text based search systems do not require lots of effort to build, compared to ontology-based systems.

Finally, the study will compare the retrieval results of a normal textual-repository search against the effort of building an ontological experience base system. We argue that intelligent searches should provide better results due to the relations and rules defined in it. However, domain modeling and tool building to support an ontology-based system require significant effort. Therefore, we need to examine the effort of building such a system, and compare it to the search results we get from text-based systems.

2.7 Summary

In this chapter, we have looked at different ideas, techniques, and methodologies within various disciplines. These methodologies examine related work or take a role directly in the implementation of this study. We have looked at the ideas behind the experience factory. Based on that, we have looked at different systems that use the experience factory ideas and how they implement them. We also discussed the concept of ontology and what it means. Then we moved to how it relates to modeling experience and languages that supports it. We finally talked about information retrieval concepts. We discussed how its measurements would help us determine the efficiency of our experience retrieval.

Chapter 3

Exploratory Setup

3.1 Introduction

By now, we have managed to get an idea of topics relevant to the study. We talked about some of the important aspects of keeping knowledge inside software organizations. We also talked about ideas and methods that support this goal. We discussed different ways of retrieving information and how they relate to each other, in addition to ways of measuring retrieval efficiency.

Throughout the second chapter, we went into some detail about disciplines and technologies related to knowledge management. We talked about methods such as the experience factory, knowledge representation languages, and information retrieval. Many of such technologies and methods like these play roles in establishing a knowledge management process inside the organization. Each is a stand-alone science; however, they serve essential steps within the knowledge management process.

In this chapter, we will develop an exploratory sample project and describe its development process. Based on information from the project case study, we will discuss

the process of implementing an experience base using the ontological approach. We will introduce a set of ontologies and illustrate how these ontologies facilitate knowledge retrieval. Then, we will implement a similar experience base using text-based implementation. After describing the setup of the text-based implementation, we will discuss the results in Chapter 4. Eventually, the case study will help us explore the capabilities of each approach and the quality of results each can offer. From another perspective, it will also demonstrate the effort and resources an organization can expect to face upon building a knowledge retrieval solution based on one of these two approaches.

Before going into the details of the ontology-based approach and how the different steps work together, we will have a look at the ontology definitions we have used for our study.

3.2 Proposed Ontologies

We will be using two ontologies to describe our software development domain: skill[4] and process ontologies. The two ontologies will cover different aspects of activities within software development such as project planning, task assignments and workflow information. However, as we discussed earlier, ontologies are never complete or fully comprehensive. Ontology accuracy of domain description is highly dependant on the number, accuracy and comprehensiveness of concepts, relations and rules set within that ontology.

The overall goal of the study is to evaluate the ontological approach of software engineering experience repositories and measure it against textual retrieval of knowledge. Therefore, the skill and process ontologies focus on areas of the software development process sufficient to allow us to model a real-life example of a development process. The

[4] The original skill ontology was provided by the Norwegian University of Science and Technology [NTNU 01]. We extended and updated the ontology before using it in this study.

richness of the two ontologies might not suit any arbitrary applications, as different applications have different needs. However, they provide a comprehensive description of skills and a process model, enabling us to model real-life software development processes.

3.2.1 Skill Ontology

The skill ontology defines a hierarchal structure of different kinds of skills a person can possess. Skills are obtained from various sources, such as certified training, formal education or previous work experience. Skills have a great effect on the way employees carry out tasks; therefore, the assignment of tasks to proper employees is key to the potential success of a project.

Following is the structure of the skill ontology. The structure shows major concepts and sub-concepts, while a complete F-Logic listing of the ontology is in Appendix A.

- Formal Education. This category is concerned with skills obtained through institutions and certificates training. It contains
  - Certifications: *Certification given by organizations*
  - Degree: *An educational degree*
  - Field of study: *Major in post secondary degree*
  - Formal training: *Training given formally by an organization*
- Formal Experience. This is experience gained from previously filling a job position.
  - Jobs: Previous positions
  - Experience: Work experience in years
- Technology and Methods. This category contains different types of skills related to a certain technology within computer science
  - Artificial Intelligence
  - General software technologies
  - Graphics design and multimedia
  - Object technology: *Technologies based on object orientation*

- Programming languages
- Project Management: *Disciplines in managing projects*
- User interfaces

- Tools and Applications. Skills within this category are related to using one or more of the tools that fit within that sub-category.

  - AI development
  - Database development environments
  - Graphics, multimedia and presentations
  - Office applications
  - Operating systems
  - Programming environments
  - Project management tools
  - Utility programs

- Other Skills. Skills that do not fit in other categories but are believed to have an effect on carrying out certain tasks.

  - Artistic abilities
  - Business type experience: *Field of experience*
  - Foreign culture: *Familiarity with other cultures*
  - Languages: *Spoken human languages*
  - Natural science: *Basic science fields*
  - Outdated: *Technologies not used any more*

3.2.2 Process Ontology

A process model is a guide for development processes of projects. Each project has a model to follow that guides its workflow. While the ultimate goal is to deliver software for customers, the techniques in carrying out development tasks vary among different models. Agile methods, for example, focus more on rapid production of pieces of code with little documentation. On the other hand, traditional models pay more attention to producing a complete solution (documents and code) in logically ordered phases of development.

For this reason, it is hard to create a single ontology that is capable of describing all software development. Some organizations develop software using a pre-defined process model, or a customized one to fit their needs. If an organization is implementing

an ontology-based experience retrieval system, it is normal to make the ontology reflect the model used inside the organization. Developing ontology for a generic process model will weaken the ontology. This is due to the differences between the models in structure of tasks, number of iterations, and priorities.

The strength of ontological descriptions is that they are precise. Ontologies can define a process model in detail, including its main entities, workflow, relations, outputs, etc. Therefore, to create only a single ontology that defines the whole domain of software development, one can end up with one of two cases:

1. Creation of a big and complicated ontology containing not only a large number of concepts, but also a complicated structure of relations and rules. Some relations and rules can end up with complicated structures because they need to handle different contexts where a single concept can exist.

2. Creation of an ontology that is big and complex, but at the same time, limited in the comprehensiveness of describing some or all software development models.

Although it is beneficial to have a generic and comprehensive ontology for describing the software process, it is a hard task to build one. However, it is possible to build distinct ontologies that can separately describe their domains, in our case, a certain software process model. Nevertheless, as we mentioned in the second chapter, handling two or more ontologies can be problematic. This is due to the possible overlap between concepts and their meaning, which also directly affects the rules and relations related to these concepts.

For the purpose of our study, we built our ontological solution based on the *Unified Process* model (UP) [Booch 99]. This ontology will merge with the previously discussed skill ontology, as there is no overlap due to different domain descriptions. The two ontologies will make use of each other, as instances of one will represent values of relations expressed in the other.

The UP ontology described here is a refinement of the widely known unified process model. It will describe basic concepts of the model and their relations to each other. Relations and attributes for each concept are set, enabling us to model a real-life project using this ontology. Following is a brief concept structure of the ontology[5], while the complete ontology with its relations is listed in Appendix B.

- Activity and Activity Set: *An Activity is a tangible workflow piece with well-defined responsibilities, inputs and results.*
- Architectural Pattern: *A pattern that defines a certain structure or behavior, usually for the architectural view of a specific model.*
- Architecture
- Artifact and Artifact Sets: *An artifact is a tangible piece of information that is created, and used by workers. It can be a model, model element or a document. Two subtypes exist:*
  - Engineering Artifacts: e.g., *Pieces of code*
  - Management Artifacts: e.g., *Planning Documents*
- Class: *A type in object oriented terminology*
- Component: *A well defined part of the system*
- Cycle: *The software life cycle. In UP it is the cycle of the four phases in order, i.e. inception, elaboration, construction and transition.*
- Iteration: *A defined set of activities resulting in a release. An iteration goes over the core workflows, i.e. Analysis through test.*
- Milestone: *A point in the process where certain objectives are met, artifacts are built or decisions have to be made. A major milestone is made at the end of each phase. A minor milestone marks the end of each iteration.*
- Phase: *A span of time between two major milestones. Each project life cycle is made of the four following phases (sub-concepts of phase):*
  - Inception
  - Elaboration
  - Construction
  - Transition
- Release: *A set of relatively completed artifacts.*
- Requirements
- Risk
- Software Representation: *A model that describes a certain functionality or structure of a system. It includes the following sub-concepts:*
  - Analysis Model
  - Deployment Model

[5] Definitions of concepts are taken and summarized directly from the book: *The unified software development process*, [Booch 99]

        - Design Model
        - Domain Model
        - Implementation Model
        - Test Model
        - Use Case Model
    - System: *A complete part of the whole project that performs a certain predefined functionality.*
    - Workflow: *Made of different types of work that usually makes the life cycle of a project.*
        - Core Workflow
            - Requirements
            - Analysis
            - Design
            - Implementation
            - Test
        - Iteration Workflow
    - Worker: *A person who is responsible for carrying out a certain task within the development process of the project.*

Briefly, the unified process consists of four phases: inception, elaboration, construction and transition. Each phase consists of a number of iterations until the phase is complete. Each iteration consists of the five workflows (Requirements through Test). Milestones are set at the end of iterations or phases and result in releases. A release will contain a set of artifacts consisting of either code artifacts or managerial documents.

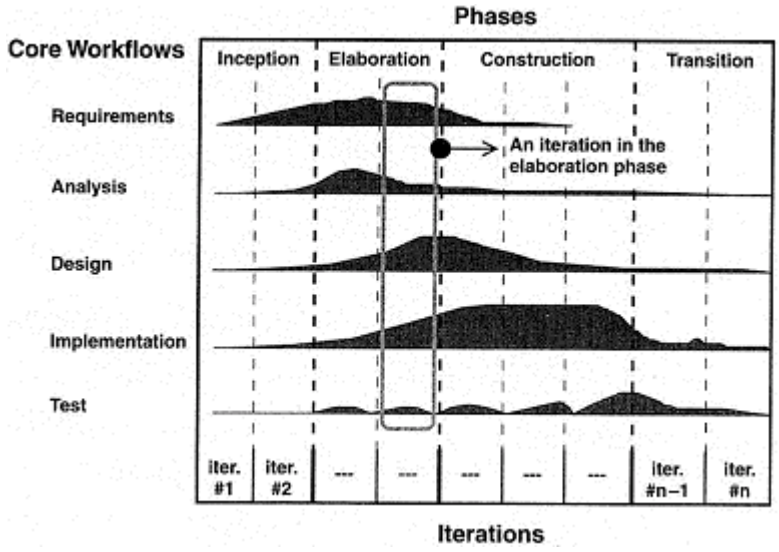

Figure 3.1 Structure of a unified software process [6]

[6] *The unified software development process,* [Booch 99].

The figure explains the order and relations of the four main phases in the project life cycle. It points out the order of the core workflows within every iteration. The graph also shows the amount of work carried out over different phases, iterations and workflows.

The final ontology used in this study is a merge between the skill and the process ontology. In addition, it also contains a number of miscellaneous concepts. Instances of these concepts will represent values for defined relations in the skill or process ontologies. Examples of these concepts are

- Agent: Instances of Agent will take roles in assignments of tasks and will be assigned specific skills.
- Scales: Different instances of *Scales* are used to give values to a number of relations, e.g. a scale of 10 or a set of (High, Medium, Low).
- Models: Sub-concepts of model can be, for example, process models or estimation models. Instances of these are associated with relations that have a range set to one of these sub-models.
- Project: Instances of project will represent different projects modeled and added to the ontological experience base.

In the next two sections, we will discuss our ontological and textual approaches in setting up the experiment for this study.

3.3 Our Ontological Approach

In order to understand how the ontology-based solution works, we will discuss the solution from four different perspectives. Each one will focus on a certain part of the solution explaining its importance and role within the system. Eventually, we should be able to see the big picture of how different parts of the solution work together to establish an experience base retrieval system.

3.3.1 The Ontology Component

In order to build an ontology-based knowledge base system, the methods we discussed earlier in Chapter 2 will have to collaborate. We will discuss how these technologies collaborate by demonstrating the roles they play within the knowledge management process. In order to illustrate the building process of the solution, we will simulate a project development process: a library system developed using the unified software development process [Booch 99].

The steps towards building an ontological knowledge base system start with building the ontology. Since ontology is domain specific, we need to develop the ontology that describes our domain: the software engineering domain. However, software engineering collects a wide range of areas, making it a very hard task to develop an ontology that describes it as a whole. For this reason, it is important to develop a single ontology that describes the world of knowledge that our system will need to be based on.

To limit the domain scope, our study is mainly concerned with retrieving experiences related to software development processes. In other words, the knowledge base system we propose will be concerned with retrieving knowledge of activities related to the development of software products. The knowledge conceived in the knowledge repository will vary due to the variant aspects related to software development.

We will be using the skill and process ontologies discussed in the previous section. Skill ontology describes skills that can exist inside the organization and possessed by its employees. Process ontology describes the development process the organization carries out. The combination of these two ontologies will cover different aspects of the software development process, and provide a base for establishing an experience repository.

3.3.2 Accumulating Knowledge

Not only does ontology define a certain domain; it also stands as the model for future knowledge gained. As the organization gains experience by carrying out more projects, it must store this experience in the organizational knowledge base. In order to be compatible with the already existing experience structure, experience has to be modeled according to the ontology that describes it. In other words, ontologies define the way experience-based systems understand experience.

Based on this, the ontological repository is made of two parts:

1. Ontological definitions: This is – in our case, for example – the ontological description of the software development process.

2. Accumulating knowledge: This is the knowledge an organization gains through successive projects. Modeled on ontological descriptions, this knowledge will be an instantiation of the concepts described in the ontology. For example, the process of developing our experimental project represents an instantiation of the unified process model described in the ontology definition.

Ontology definitions play the role of defining the infrastructure of the experience base. They define the concepts and relationships that instances of real-life projects will be based on in the modeling of these projects. From an object-oriented perspective, this is similar to creating instances of classes (types).

The organization has the responsibility of maintaining the experience base, by either refining it from redundancies or keeping it growing with more experience. The compilation of tasks and activities carried out previously, along with their outcomes, enriches the organization experience base. Nevertheless, the process of modeling existing experience to inject it into the experience repository is laborious. However, this is part of the overall cost of carrying out a knowledge management process inside the organization.

3.3.3 The knowledge-based System

After modeling and packaging experience into the experience base, the system core engine will handle stored knowledge. In a database, for example, the database management system (DBMS) provides the upper layer of the system that can manipulate stored data. Similarly, a knowledge base retrieval engine is a higher reasoning system that can read, analyze and evaluate stored knowledge.

The knowledge base system is capable of reading the representation language used for laying out the ontology. In our case, we are using F-Logic to represent our developed ontology. The core of the knowledge base system is a frame-logic inference engine, capable of parsing F-Logic grammar[7]. Both knowledge and ontology definitions are modeled using F-Logic syntax. The engine logical inference capabilities are capable of analyzing relationships logically in order to find matches to issued queries.

To illustrate the previous discussion, let us consider the following small example written in F-Logic[8].

// Definitions and instantiations

Frank:Employee. *// Frank is an instance of employee*
Philip:Employee. *// Philip is an instance of employee*
MSc:Degree. *// MSc is an instance of Degree*
PhD:Degree. *// PhD is an instance of Degree*

Object Oriented Languages::Programming Language.

// Object Oriented Languages (OOL) is a
// sub-concept (sub-type) of Programming languages

Structured Languages::Programming Language.

// Structured Languages(SL) is a
// sub-concept (sub-type) of Programming languages

[7] The inference engine we used is called OntoBroker, developed by Ontoprise®, a provider of semantic technologies and solutions. (http://www.ontoprise.de)

[8] As in Java and C++, inline comments in F-Logic are preceded by double slashes (//)

Java:Object Oriented Languages. // Java is an instance of OOL

C++:Object Oriented Languages. *// C++ is an instance of OOL*
Fortran:Structured Languages. *// Fortran is an instance of SL*

Frank[aDegree ->> PhD;Skill ->> Java].

// Frank has a PhD degree and is skilled with Java

Philip[aDegree ->> MSc;Skill ->> Java;Exp@(Java)->4].

// Philip has a MSc degree and is skilled with Java, he has four

// years experience with Java

// Queries

FORALL X <- X[aDegree ->> PhD].

// The query should return everybody who has a PhD, i.e., Frank

FORALL X,Y <- X[Skill ->> Y:Programming Language].

// The query should return whoever is skilled with a programming language

// i.e., Frank and Philip

// This example illustrates the ability to search using a parent concept
FORALL X,Y <- X[Exp@(Y:Object Oriented Languages)->_greater(3)].

// The query should return whoever is experienced with any object oriented

// language for more than three years, i.e., Philip

To further explain the F-Logic example, note the following syntax:

1. A double colon denotes inheritance. For example, to indicate that concept A inherits concept B, we use the syntax: *A::B.*
2. A single colon denotes instantiation. For example, to indicate that "a" is an instance of A we use the syntax: *a:A.*
3. Attributes are assigned to instances using "->>". For example, consider the object A: *A[attribute ->> value]*.
4. For assignment of a value to an object based on certain inputs, we use *methods*. For example, to say that Philip has an M.Sc. degree in software engineering we write:

Philip[Degree(Software Engineering)->> MSc].

5. Values can be integers, Boolean, Strings or other instances.
6. Multiple statements are separated using a semicolon.
7. All statements end with a dot.

The first part of the example (Definitions and Instantiations) defines instances of concepts and attributes of these instances. The second part (Queries) gives examples of possible questions (Queries) that can run against our simple ontological experience base. These queries do not exist in the knowledge base. They are issued by a higher-level user interface.

The kernel of a knowledge base system is the inference engine, which is capable of parsing a certain knowledge representation language. Applications built around the inference engine utilize these capabilities in order to provide meaningful information for the user. For example, a project manager is the person usually concerned with querying the knowledge base for employees' skills. Applications built around the knowledge base are supposed to provide the query user interface. The user interface should hide the technical details (F-Logic syntax, for example) of that skill query.

3.3.4 The Learning Loop Link

Like all knowledge systems, ontology-based knowledge systems aim ultimately to provide knowledge back to users. This implies that knowledge based systems should be capable of retrieving only related knowledge from the knowledge repository. This is essential, as the repository is likely to have knowledge related to various aspects of the domain.

After the analysis and modeling of knowledge gained by carrying out new projects, this knowledge is stored in the knowledge repository. This is the first step towards creating the link between gaining knowledge and re-using it. We have already talked about means of representing knowledge in order to store it. Since representation

languages are mathematical notations laid out in textual files, there are many means to store and handle this knowledge.

The real importance of implementing a knowledge-based system appears when the need for a specific knowledge piece arises. Technically, project managers are concerned with stored knowledge as it provides essential input for planning new projects. When new projects get started inside an organization, project managers are responsible for setting up a complete project plan for the new project. The plan includes workflow information such as the following:

- Top level processes and sub-processes (tasks and sub-tasks)
- Time frames (start and end dates) for each task
- Input and output artifacts from and into each task
- Resources assigned to each task
- Any other related information, such as release dates and sets of artifacts

Such information is essential for a project manager upon setting up a new project plan. The knowledge base contains practices, tools and resources used previously. Outcomes of implementing tasks also provide key information in the knowledge base. Previous outcomes provide an indication of possible outcomes upon running the same scenarios in current projects.

It is up to the manager to provide a logical input for the knowledge base system in order to retrieve usable knowledge. Similarity to previous practices is the key for retrieving usable information. However, similarity in practices is affected by more than one factor, since a practice has many inputs, e.g., tools, skills necessary, workflow information, etc. It is important to analyze inputs given into a previous practice in order to understand its outcomes. Understanding these gives the project manager feedback on the possibility of applying the same practice again successfully.

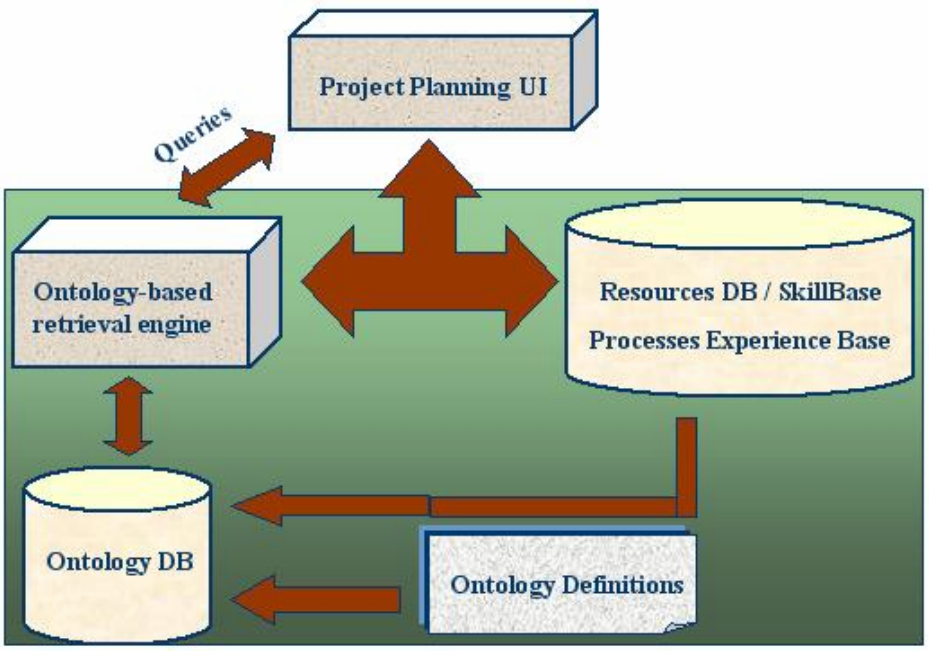

Figure 3.2 Components of an ontology-based system

Figure 3.2 shows – in addition to project planning UI – the different components taking part in the ontology-based system. Ontology definitions and instances participate in making the database. The ontology inference engine is responsible for executing the queries against the database. The whole system interacts with the project manager through planning GUIs capable of issuing queries to the database.

When discussing inputs, we are referring to many aspects that affect different sub-processes. Many variables may affect a sub-process depending on its type. These types include design, build, or test sub-process. Project planners should consider these variables when using previous practices, as they affect potential reuse. The following are examples of such variable inputs:

1. Skills required carrying out the sub-process, e.g., education, certified skill or previous experience.
2. Time frame a sub-process will need to finish: Such input will also depend on availability of other resources.

3. Tools required by the task: These can vary depending on factors such as programming language, design methodology, testing techniques, etc.
4. Process model governing the sub-process: This can affect sub-processes directly regarding the number of times executed, its order within other sub-processes or type of output expected.
5. Testing techniques and quality assurance practices.
6. Number of team members involved in the sub-process.
7. Any other factor that can be expected to affect a sub-process or its quality, either as a process or the quality of its output.

An experienced manager is more capable of analyzing factors affecting previous tasks and their reuse suitability. The experienced manager should be able to compare the situation where a previous practice occurred and reflect on the possibility of re-performing that practice. However, stories of unsuccessful practices do not prevent the manager from reusing a practice after identifying weaknesses.

Skills required or available to carry out a certain task play an important role in the success of executing a task. For this reason, we focus on the skill ontology and how a manager can query for available skills within the organization.

3.4 Our Text-Based Approach

In this section, we will talk about the different aspects of setting up our textual implementation of the study experiment. We will explain how our project development model uses our proposed unified process model ontology. We will also explain how the textual repository will help retrieve information about previous practices.

3.4.1 Documentation of Projects

When an organization carries out a project, documentation of this project will exist in various types and formats. Different phases of development produce different types of documents that describe parts or all of the system functionality. For example, the design phase usually produces database schemas or class diagrams, while a build phase produces source code. Similarly, phases like planning, requirement elicitation, or testing might produce textual documents.

From an experience base perspective, the most important documents are the ones that provide information about workflows, planning, assessment, and task assignments. Project planning documents, workflow descriptions, and inspection documents are all examples of documents that can provide substantial information about the project's life. The compilation of these documents stand as a repository of a project history, one that is easily searchable using the proper tools.

On the other hand, organizations tend to use configuration management systems to keep track of their software builds. These systems contain various project-related documents. Hence, they contain important historical information on projects carried out inside the organization.

When searching a repository of text-based documentation, keyword usage is the way to find relevant documents. Information retrieval based on indexed text searches stands as an ideal way to finding relevant information between large numbers of documents. This approach is commonly used inside organizations for two reasons:

1. Many project documents are either text based or can be stored in textual formats. These documents already represent a formal documentation of projects. Hence, in many cases, not much effort is needed to reproduce them for storage purposes.

2. Text search is an inexpensive method (from an implementation perspective) of finding information. Tools and applications that perform text search are inexpensive and easy to install and use at the organization site.

3.4.2 Documentation of the Study Example

In our example, we have documented the development process of the project. The documentation contains all aspects of the development, including:

1. The process development model including phases, iterations, and core workflow information throughout the project life cycle.
2. Workflow information of processes and sub-processes, including their order and the flow of documents between tasks.
3. Milestones and releases, including all artifacts participating in a release.
4. All subtasks making iterations, their input, output and activities relevant to these tasks.
5. Inspection, testing and quality assurance activities for every sub-system of the final product.
6. Skills required for carrying out every single task through out the whole development cycle. Documentation includes tools and technologies needed to carry out these tasks.
7. Output quality of every task performed through the whole project life cycle. This is essential to exist inside the experience base as it points out the success of performing that task.

3.4.3 Role of the Study Documentation

The example will serve the two following important tasks in this study:

1. To provide textual documentation for the text search engine

   The documentation of this project will be part of the textual repository that the search engine will index[9]. As with previously mentioned textual repositories existing inside organizations, we will add our documentation to a larger number of related and non-related documents. We will examine the ability of a text-search engine to find relevant documents to our queries.

2. To provide the base for an ontological description

   In order to examine the feasibility of building an ontology-based system, we needed to go through the effort of modeling our example based on our ontological definitions. The documentation of the project provides the necessary information to document the development process and workflow information ontologically.

In the following section, we will briefly describe the project and the process it went through. The complete documentation of the project – which is the same documentation indexed by the search engine – can be found in Appendix D.

3.4.4 The Project Model

The project aims to develop a library system for the University of Calgary. The library system will be able to provide book inventory functionality, including checking books in and out. It will also be able to manage library users and keep track of their checked books and calls. There is a sub-system designed for reservation of materials and another sub-system that allows access to the system through the web.

[9] To index text-based documents is to go over all documents identifying keywords and linking them to their documents in order to find documents quickly based on their keywords. This technology is widely used especially in Internet search engines.

Development of the project runs over the four phases defined in the Unified Process (UP) model [Booch 99]. Each phase ends with a named milestone. Each milestone has its release of a defined set of engineering or managerial artifacts[10]. We will emphasize the development tasks taking place within the construction phase. The phases and their sub-tasks are as follows:

1. Inception Phase: This is where the initial work for the project takes place. The work in this phase will be over at the end of the third iteration. The iterations include tasks like configuring hardware requirements, defining functional and non-functional requirements, and producing final design and requirements documents.

2. Elaboration Phase: This is the phase where the overall shape of the system takes place; modules are identified and assigned to workers. At the end of this phase, an overall skeleton exists for the project.

3. Construction Phase: This is the phase where most of the development takes place. It was designed to finish after four iterations, where each iteration will produce a sub-system of the project. The four sub-systems are the insertion, borrowing, reservation and the web-based system.

4. Transition Phase: This is the phase where the system is delivered to the user. In this phase, the network is set up and all subsystems are installed and tested at the customer site.

We have designed the construction phase to run over four iterations. Each iteration will result in the production of a sub-system (it usually takes more then one iteration in a real life project). There is a milestone at the end of each iteration marking a

[10] Engineering artifacts are modules written in a programming language that will eventually be part of either the executable code of the final product or a technical document describing the final product (e.g. design documents). Managerial artifacts are project documents that are related to the development process itself, like project plans, inspection, and assessment documents.

release. The release of a single iteration will be a working sub-system with all its managerial and engineering artifacts. In addition, we have defined workflow of tasks within iterations and the output of each one. For the purpose of the study, we also documented skills required to carry out each task within a certain system.

All systems had their own plan and assessment documents. We documented all artifacts, stating the quality, workers and skill required to produce them. In addition, we assigned a set of workers to work on the system where each one was assigned a specific role. Workers were assigned a predefined set of tasks according to their skills.

It is important to notice that the system itself (the library system) was not developed, i.e., engineering artifacts do not exist. The goal of the development of this exploratory project is to produce the project documentation so that we can use it in the text-based retrieval system. The complete documentation produced and indexed by the search engine is in Appendix D.

3.4.5 Text-Search Implementation

To put our example to work and test the retrieval of the text-search engine, we need to link everything together. We need to take two main steps before running the search engine.

1. Build the textual repository out of all the documentation we need.
   The repository will be a complete directory structure of all textual documents. For realistic results, we needed to include irrelevant data in the repository in order to measure the accuracy of the search engine in retrieving only relevant documents. At the same time, we need to include documents that are similar to our example, but not relevant, i.e., likely to contain similar keywords. We have included the following in the repository:

- All documents from the example project. A printout of the documentation is in Appendix D.
- A softcopy of the book: The Unified Software Process Model. This book stands as a template for the development process governing our sample project.
- A listing of agents – assumed to work in the organization – along with all their skills and expertise. This simulates employees' records in a real organization where all their skills and experiences are documented in their profiles.
- A softcopy of a java book. This is irrelevant information; however, it contains many of the keywords that exist in our sample project. This will allow better evaluation of the search engine precision.
- The Capability Maturity Model (CMM) [CMMI 02].
- A local copy of
  - The CMMI site
  - A business process engineering site
  - J2EE online documentation.

2. Index the textual repository and provide a retrieval interface.

   After compiling all textual data, we placed it in one directory structure. After that, we used a text search engine dedicated to indexing websites[11] and made it crawl[12] our textual repository. The whole repository was analyzed and indexed by the search engine using web standards text-indexing technologies. We set indexing options so that special search features will not affect the results. Those options are the following:

   - No usage of synonym lists.
   - No usage of special tags in html files (e.g. ALT tag) as these tags give extra descriptive information about the content of html files.

[11] Search engine is provided by SiteLevel (http://www.sitelevel.com), a web indexing and searching site.
[12] Crawling is the term used for the process of navigating through web documents in order to index them. Software responsible for performing this procedure is called a *Spider*, hence, the term: *crawling the web*.

Local copies of sites mentioned previously – and taking part in the repository – contain lots of terminology that we use in our project documentation. All this relevant and irrelevant information simulates a real textual software engineering repository inside an organization. This is because organizations tend to have lots of documentation regarding all projects they previously carried out. Not all this documentation is relevant when searching for previous practices. Nevertheless, we need to measure the ability of a normal text retrieval engine to retrieve previous experiences by retrieving only relevant documents.

The complete implementation of the text-based experimental setup of the project is on (http://sern.cpsc.ucalgary.ca/~philip/ThesisSite/index.html). The site contains access to all data inside the repository including project documentation. It also provides an interface that will allow performing searches against the repository. The repository has 538 documents with a total number of 879,902 distinct words indexed.

The next section will explain how we came up with a set of questions for testing the text retrieval implementation.

3.5 Benchmark Questions

Our goal is to test the text search engine and its ability to find previous practices through searching repositories of documentation. For this reason, we have identified a number of questions that reflect what we believe to be the needs of project managers. These question cover many needs that a project manager may have, either when setting a new plan for a certain project or assigning responsibilities.

In section 2.6.2, we discussed the efficiency of non-text-based systems. Therefore, we will test the queries against the text-based system. This is because our ontological modeling of the experimental project is complete within itself. From our perspective, we are familiar with its contents as it represents a complete world. This is an

ideal case for an ontological description of a domain, which is hard to exist in a real world, as we will see in Chapter 5.

In Chapter 4, we will discuss each question separately, pointing out its significance. We will show the corresponding F-Logic syntax used when issuing the same query to the ontology-based experience base. Users of text search systems (e.g. web searches) do not use same keywords all the time when trying to look up a certain topic. For this reason, we have used synonyms of our keywords to examine how that affects our retrieval of relevant documents. Therefore, we have come up with variations of the same query13 using different keywords. We show the results we got after each query and evaluate the query success based on its recall, precession, and efficiency.

We have also assumed degrees of success in finding relevant documents. In other words, we do not consider the search engine to have found a relevant document if it was after the 20th hit. This is based on our assumption that from user perspective, a document is not found if extra effort needs to be made to find it again among a list of retrieved documents. We rank our range of accuracy within 1, 5, 10 and 20 hits. We calculate the recall for each range and come out with a total efficiency for every query.

Questions from 1 – 7 try to focus on the ability to find certain skills, either to match a worker or to inquire about a worker's experience. These questions correspond to the skill ontology, which is supposed to help managers inquire about skills inside the organization. Similarly, questions 12 to 20 focus on the ability to find previous practices and workflow information. These questions correspond to the process ontology we have developed which should be the basis for finding better practices in an ontology-based system. Questions 8 to 11 are special cases; we will discuss them later in Chapter 4. Following is a listing of the benchmark questions14:

[13] Complete results of testing the text search engine with the 20 questions proposed and their variations are found in appendix E

[14] Names of people, tasks, or specific artifacts are used as an example.

1. Give all skills of Philip.
2. Give all people who have an applied skill as a software tester.
3. Give all people who have any type of degree.
4. Give all people who have a skill with SQL Server and their work experience is between 7 and 9 years.
5. Give all people with experience with a certain tool for 7-9 years.
6. Get all people skilled with a tool, which is a programming language IDE, and their work experience is 4-6 years.
7. Give all people skilled with any tool.
8. To which category (concept) does a "UML certified" skill belongs?
9. Give all direct sub-concepts of "Tools and Applications".
10. Give all Concepts under Root (Return All Concepts).
11. Give all tree of concepts that XML belongs to.
12. Who participated in carrying out this task: Develop web-based System Navigation Structure?
13. Who is skilled with Enterprise Java and has good testing skills?
14. What type of skill is needed to do this task: Build Web-based System EJBs?
15. What is the test we did on a java module that produced a high quality artifact?
16. What was the previous task of building the web-based system JSPs?
17. What was the task and its outcome where normal effort was put, but we still got high quality artifact?
18. Which Planning Activity had a very high effort and resulted in a high quality plan?
19. What are the artifacts of the Reservation System release?
20. In which Iteration did we test the Classification Manager?

3.6 Summary

In this chapter, we discussed the setup of the study experiment using both ontological and text-based approaches. We discussed the ontological approach and how its different components work together. We also discussed the two ontologies that we used to describe our development process: skill and process ontologies. Similarly, we also discussed the steps of building the text-based repository of the experimental project. We produced documentation for the project to be part of the textual experience repository reflecting the project development process. Finally, we introduced benchmark questions for examining the text-based retrieval system.

Chapter 4

Exploratory Results

4.1 Introduction

In this chapter, we will discuss in detail the questions we introduced in the previous chapter. During our discussion, we will closely examine the results of these queries and what they mean. After that, we will look at the effort that was put in establishing both experience retrieval approaches, ontology and text-based. We will have a close look at steps within each approach so we can evaluate the retrieval results we obtained against the effort of establishing such systems.

4.2 Dissection of Benchmark Questions

In the following questions, we will discuss the best results we obtained using a single set of keywords, i.e., a variation of the query. A complete listing of all twenty questions and their variations, along with all the results, is in Appendix E. To explain the terms we are using in the dissection of the questions e.g. “best results”, here are a few points:

1. As mentioned in Chapter 3, the type of need defined by the syntax of the question is what we aim for. Names of people, tasks, or artifacts are only examples.
2. From the variants of each question, the one shown here is the one with the most retrieved relevant documents.
3. Recall, Precision, and Efficiency are defined in Chapter 2. However, based on our previously discussed technique in using retrieval measurements in steps of 1, 5, 10 and 20 hits, we will define our own recall, precision, and efficiency. For example, Recall$^{(20)}$ will mean a recall measurement of relevant documents within the first 20 hits. Similarly, Precision$^{(20)}$ will mean a precision measurement based on a total of 20 documents retrieved.
4. "Best Retrieval" means the best results obtained from all question variants. "X out of Y relevant documents within the first Z hits" means that X is the best number of relevant documents we were able to retrieve within the least Z hits. Y is the total number of relevant documents.
5. We will show the actual precision (not Precision$^{(x)}$), as this will give us an indication of the efficiency of the query. Recall is more concerned with how many relevant documents are retrieved from all relevant documents. However, precision is more concerned with the number of irrelevant documents retrieved, i.e., the query accuracy in distinction of relevancy. Therefore, we will also show precision value depending on the real number of documents retrieved.[15]

Following are the dissected benchmark questions:

Question 1: *Give all skills of Philip.*
F-Logic Syntax: *FORALL X,Y <- "Philip"[X ->> Y].*
Best Keywords: *Skill – Philip*
Best Retrieval: *1 out of 1 relevant document within the first 5 hits.*
Recall $^{(5)}$: 100 % **Precision $^{(5)}$:** 20 %
Efficiency$^{(5)}$: 1.0 **Precision:** 50 % **All Retrieved Documents:** 2

[15] Note that Precision$^{(x)}$ has a margin of error when the number of retrieved documents (x) is not exactly 1, 5, 10, or 20. It is shown here to correspond to the Recall$^{(x)}$ measurement which we are more interested in. The normal Precision measurement is a more accurate and favorable one to examine.

Description:

When a project manager needs to assign tasks, this type of question is the most straightforward. A manager will need to become familiar with the skills of a certain employee so that he or she can judge on the suitability of this employee's skills to match the skills required for a certain task.

The search engine was successful in returning the single relevant document from all available documents that described the skills of Philip (the worker); that document was Philip's profile.

Question 2: *Give all people who have an applied skill as a software tester.*

F-Logic Syntax: *FORALL X <- X[AppliedSkill ->> "Software Tester"].*

Best Keywords: *Testing - Experience*

Best Retrieval: 2 *out of 2 relevant documents within the first 10 hits.*

Recall$^{(10)}$: 100 % **Precision $^{(10)}$:** 20 %

Efficiency$^{(10)}$: 0.89 **Precision:** 3 % **All Retrieved Documents:** 62

Description:

This type of question is also straightforward, usually used upon trying to find a fit worker for a certain type of task. The search engine was able to find the two required documents within the first 10 hits. However, the precision was low, indicating that too many irrelevant documents were retrieved.

Question 3: *Give all people who have any type of degree.*

F-Logic Syntax: *FORALL X,Y <- X[Degree ->> Y:"Degree"].*

Best Keywords: *Agent - Degree*

Best Retrieval: *0 out of 9 relevant documents within the first 20 hits.*

Recall$^{(20)}$: 0% **Precision $^{(20)}$:** 0 %

Efficiency$^{(20)}$: - 0.01 **Precision:** 0 % **All Retrieved Documents:** 5

Description:

The intention of this question it to retrieve a listing of all educational skills that employees have. However, although we used four variations of this question, the best recall for this type of question was zero, i.e., a failure. Some documents were retrieved but none was relevant. This is most probably due to the high possibility of these keywords occurring in lots of irrelevant documents. As the question was very generic, the query was not able to retrieve the exact documents needed.

Question 4: *Give all people who have a skill with SQL Server and their work experience is between 7 and 9 years.*

F-Logic Syntax: *FORALL X <- X[ToolSkill ->> "SQL Server"] AND X[WorkExperience ->> "7-9 Years"].*

Best Keywords: *SQL Server – Skill – Year*

Best Retrieval: *1 out of 1 relevant document within the first hit.*

Recall(1): 100 % **Precision (1):** 100 %

Efficiency(1): 1.0 **Precision:** 100 % **All Retrieved Documents:** 1

Description:

The requirement of this query exceeds what we used until now. The manager in this case is querying for very specific skills of any employee. One of the main issues to notice about this kind of query is the two sides of the query, skill and length of experience. As text search lacks semantics and ability to relate pieces of information to each other, the result of this search was surprising.

Another variation of this query was not so successful; however, this one turned out to be a complete success in finding the required document. One reason behind this is the use of a very specific name of technology (SQL Server), which gave good results back. However, it is still not possible for text search technology to evaluate a condition similar to “between 7 and 9”. In fact, numbers in text searches do not hold the semantics of integers or decimals, much less constructing a relation as “between”.

Question 5: *Give all people with experience with a certain tool for 7-9 years.*

F-Logic Syntax: *FORALL X <- X[WorkExperience@(ToolSkill) ->> "7-9 Years"].*

Best Keywords: *Year - Skill*

Best Retrieval: *1 out of 2 relevant documents within the first 20 hits.*

Recall$^{(20)}$: 50 % **Precision $^{(20)}$:** 5 %

Efficiency$^{(20)}$: 0.45 **Precision:** 4 % **All Retrieved Documents:** 28

Description:

This question reveals the weakness points of text search. Unlike the previous question, this one does not search for a specific term. However, the search in here depends mainly on the semantics of a relation “between 7 and 9”. Apparently, the success of this search was not optimal, although it partially retrieved some of the relevant documents. The best efficiency of this search was almost 50%, retrieving half of the needed documents. On the other hand, this result was achieved within 20 hits and with lots of irrelevant documents retrieved, which explains the low precision.

Question 6: *Get all people skilled with a tool, which is a programming language IDE and their work experience is 4-6 years.*

F-Logic Syntax: *FORALL X,Y <- X[ToolSkill ->> Y:"Programming environments"] AND X[WorkExperience ->> "4-6 Years"].*

Best Keywords: *Year – Skill – IDE*

Best Retrieval: 2 *out of 3 relevant documents within the first 5 hits.*

Recall$^{(5)}$: 67 % **Precision $^{(5)}$:** 40 %

Efficiency$^{(5)}$: 0.67 **Precision:** 100 % **All Retrieved Documents:** 2

Description:

This question demonstrates the manager’s need to inquire about workers capable of handling a certain tool efficiently. The keyword “IDE” along with “Year” helped the search engine find the required documents. These two words will most likely exist in the same document if that document is a skill profile. The high precision confirms this point since no irrelevant documents were retrieved. Recall,

on the other, hand shows that only two of the three required documents were actually retrieved.

Question 7: *Give all people skilled with any tool.*

F-Logic Syntax: *FORALL X,Y,Z <- X[ToolSkill ->> Y:Z]*
AND Z::"Tools and Applications".

Best Keywords: *Agent – Expert – Tool*

Best Retrieval: *0 out of 6 relevant documents within the first 20 hits.*

Recall$^{(20)}$: 0 % **Precision $^{(20)}$:** 0 %

Efficiency$^{(20)}$: -0.02 **Precision:** 0 % **All Retrieved Documents:** 8

Description:

This question is supposed to return documentation on any employees skilled with using any kind of tool. It is true that such questions for an ontology-based experience base will result in a nicely ordered list; however, we expected the text search to return at least some of the relevant documents. On the contrary, because we did not use any specific keywords, the best efficiency we got was –0.02. None of the relevant documents had the keywords inside them, so none were returned. The efficiency was taken from the query with the least number of irrelevant documents retrieved.

The next four questions (8 – 11) were not tested for retrieval efficiency using the text search engine. The reason for this is that these questions inquire about the structure of the ontology itself that we – or an organization – are using. The ontology not only defines the manner of reading a domain by computer software; it also reflects the structure of that domain. Our skill ontology, for example, categorizes different kinds of skills under five major categories.

For this reason, we believe that it is important for a manager to inquire about properties of certain concepts or instances that exist in the organization ontologies. In ontology-based systems, these queries return very specific and defined answers. Using

sample names of concept and instances, here are examples of the questions possibly issued by project managers:

Question 8: *To which category (concept) does a "UML certified" skill belong?*
F-Logic Syntax: *FORALL Y,Z <- "UML Certified":Y AND directsub_(Y,Z).*

Question 9: *Give all direct sub-concepts of "Tools and Applications".*
F-Logic Syntax: *FORALL Y <- directsub_(Y,"Tools and Applications").*

Question 10: *Give all Concepts under Root (Return All Concepts).*
F-Logic Syntax: *FORALL Y <- Y::"Root".*

Question 11: *Give the tree of concepts that XML belongs to.*
F-Logic Syntax: *FORALL Y <- "XML":Y.*

In the next few question (12-20) we will introduce different types of queries that focus on the development process. These questions will deal more with the developed example we discussed earlier, aiming to compare finally with the effort of building process ontology. Queries issued here map to structures of questions that a project manager will most likely use to inquire about techniques, outcomes, or skills associated with a certain task.

Question 12: *Who participated in carrying out this task: Develop web-based System Navigation Structure?*
F-Logic Syntax: *FORALL X,Y <- X[Agent->>{Y};ParticipateIn->>{"Develop web-based System Navigation Structure"}].*
Best Keywords: *Agent – Navigation – System – Participate*
Best Retrieval: *1 out of 1 relevant document within the first hit*
Recall(1): 100 % **Precision(1):** 100 %
Efficiency(1): 1.0 **Precision:** 100 % **All Retrieved Documents:** 1

Description:

This query is used to find documentation about workers who previously carried out certain tasks. In this case, we have used the name of the task explicitly (keyword “Navigation”) which resulted in very good results. Other variations of the query gave good results, however, this one returned the only relevant document and did not return any other. Such search resulted in perfect recall returning all relevant documents, and perfect precision in eliminating all irrelevant ones.

Question 13: *Who is skilled with Enterprise Java and has good testing skills?*

F-Logic Syntax: *FORALL X <- X[SkillLevel@("DataBase Systems","Enterprise Java","Java")->>{"High"}] AND X[TechnologySkill->>{"Testing and Testing Techniques"}].*

Best Keywords: *Skill – Java – Test*

Best Retrieval: *1 out of 1 relevant document within the first 5 hits*

Recall$^{(5)}$: 100 % **Precision$^{(5)}$:** 20 %

Efficiency$^{(5)}$: 0.98 **Precision:** 10 % **All Retrieved Documents:** 10

Description:

A manager will issue such query to inquire about best-fit workers to carry out a task with certain skills required. Apparently, being precise in the skills queried for has helped in obtaining good results. Documents giving information about employees with such skills were successfully retrieved. On the other hand, keywords indicating these skills have high occurrences in many other topics. For this reason, the precision of this query was not high.

Question 14: *What type of skill is needed to do this task: Build Web-based System EJBs?*

F-Logic Syntax: *FORALL X <- "Build Web-based System EJBs"[Skill->>X].*

Best Keywords: *knowledge – Develop – EJB – Web*

Best Retrieval: *1 out of 1 relevant document within the first hit*

Recall(1): 100 % **Precision(1):** 100 %

Efficiency(1): 1.0 **Precision:** 100 % **All Retrieved Documents:** 1

Description:

Depending on its outcome, the manager will need to inquire about the skills of the worker who has carried out a certain task. The main goal of this will most probably be to investigate how successful this employee was carrying out this task using his or her skills. Such feedback can give an insight for the manager on potential re-assignment of workers with same skills to a similar task.

Because this query names the tasks very specifically, recall and precision were optimal. This query managed to retrieve the only relevant document and nothing else, which is a complete success.

Question 15: *What is the test we did on a java module that produced a high quality artifact?*

F-Logic Syntax: *FORALL X,Y <- X[TestType ->> Y] AND X[Type ->> "Java Code"] AND X[Quality->>"High"].*

Best Keywords: *Test – Java – Artifact – High*

Best Retrieval: *1 out of 1 relevant document within the first 5 hits*

Recall(5): 100 % **Precision(5):** 20 %

Efficiency(5): 1.0 **Precision:** 50 % **All Retrieved Documents:** 2

Description:

Again, this question is an example of a very useful piece of information that a manager will need to get. It is important for a manager to inquire about what type of activity (in this case, testing) was previously carried out on a module, at a time that module proved to be of high quality.

The search engine was able to retrieve the document describing the test process on that artifact, resulting in a good recall from the first five hits. The precision, however, shows that only one other irrelevant document was retrieved.

Question 16: *What was the previous task of building the web-based system JSPs?*

F-Logic Syntax: *FORALL X <- "Build Web-based System JSPs"*
[PreviousTask ->> X].

Best Keywords: *Task – Previous – Web-Based system JSPs*

Best Retrieval: *1 out of 1 relevant document within the first hit*

Recall(1): 100 % **Precision(1):** 100 %

Efficiency(1): 1.0 **Precision:** 100 % **All Retrieved Documents:** 1

Description:

The importance of this question comes from the need to know about workflow that occurred in a previous project. The goal is to find what took place previously before carrying out a certain task.

Although such a query is asking about some kind of order of tasks executed, one would expect poor results. This is because in text searches it is not possible to express order of tasks or even relate parts of the text to each other. However, using the specific task name allowed the search engine to retrieve the proper document describing the order of tasks relevant this one. It is worth mentioning that a close variation of keywords not containing the task full name did not succeed in retrieving the needed document. This one, though, was a success from the first hit.

Question 17: *What was the task and its outcome where normal effort was put, but we still got high quality artifact.*

F-Logic Syntax: *FORALL X,Y <- X[Effort ->> "Normal Effort"]*
AND X[outputArtifact->> Y[Quality->>"High"]].

Best Keywords: *Normal Effort – High Quality*

Best Retrieval: *1 out of 4 relevant documents within the first 5 hits*

Recall(5): 25 % **Precision(5):** 20 %

Efficiency(5): 0.23 **Precision:** 10 % **All Retrieved Documents:** 10

Description:

This question has a number of relations and attributes that links artifacts with their efforts and outcomes. Although such a question can be helpful for a manager, it is highly expected that a text search will not provide a very accurate result. The best search was able to retrieve only one document out of four. There were ten documents retrieved, making the precision equal to only 10%.

Question 18: *Which Planning Activity had a very high effort and resulted in a high quality plan?*

F-Logic Syntax: *FORALL X,Y <- X[Effort ->> "HighEffort"] AND X[ResultsDocumentation->> Y[Quality->>"High"]].*

Best Keywords: *Plan – Very High Effort – Good Quality*

Best Retrieval: *0 out of 2 relevant documents within the first 20 hits*

Recall$^{(20)}$: 0 % **Precision$^{(20)}$:** 0 %

Efficiency$^{(20)}$: -0.01 **Precision:** 0 % **All Retrieved Documents:** 6

Description:

This question was aimed to find the details about a task that ended with a good quality artifact, and at the same time required a certain amount of work. This is a potential type of question when there is a need to relate effort to quality in carrying out tasks. However, in this case, because a specific task or artifact was not named, we were not able to obtain good results. Although some documents were retrieved, none was relevant.

Question 19: *What are the artifacts of the Reservation System release?*

F-Logic Syntax: *FORALL X,Y <- "Reservation System Release"[MadeOf ->> X] AND X[Artifacts->>Y].*

Best Keywords: *Reservation – System – Release – Artifacts*

Best Retrieval: 2 *out of 2 relevant documents within the first 5 hits*

Recall$^{(5)}$: 100 % **Precision$^{(5)}$:** 40 %

Efficiency: 1.0 **Precision$^{(5)}$:** 100 % **All Retrieved Documents:** 2

Description:

This type of questions is in some respect different from the previous ones. The style of question resembles the queries usually issued against databases to find associated pieces of information. This, however, is also a potential question a manager will need to ask. It might not help directly in retrieving information about previous practices, but it helps to retrieve information about previous products. In addition, such an inquiry might be the first step towards another query about the retrieved information.

The results we got from this search were very good. As the information we were looking for was very precise, keywords were helpful to find exactly the documents needed. In fact, not only did we obtain all relevant documents; we also got a 100% precision, i.e., none irrelevant documents.

Question 20: *In which Iteration did we test the Classification Manager?*

F-Logic Syntax: *FORALL X,Y <- "Reservation System Release"[MadeOf ->> X] AND X[Artifacts->>Y].*

Best Keywords: *Classification Manager – Task – Iteration – Test*

Best Retrieval: *0 out of 3 relevant documents within the first 20 hits*

Recall$^{(20)}$: 0 % **Precision$^{(20)}$:** 0 %

Efficiency$^{(20)}$: -0.01 **Precision:** 0 % **All Retrieved Documents:** 5

Description:

Workflow information of previous projects is important information. It provides historical documentation of the process that took place in developing previous products. In this example query, a manager is inquiring about when a certain task took place within the development cycle, i.e., at which iteration of product development.

One would expect such a query to return good results, however this was not the case. Although it is a straightforward question, such information did not occur in the same document. A closer look at the question will show that more than one step of documentation took place for documenting the phase and its iterations, and

tasks involved in the iteration. For this reason, the search engine was not able to find all keywords in one document. The search engine was also not able to relate the contents of the three documents involved; hence, the overall performance of the search was poor.

Summarizing the results we got from discussing all the questions above, we find that 75% of all searches eventually managed to return relevant documents. On the other hand, this leaves 25% of the questions that did not retrieve any of the relevant documents. The results in more detail are as follows[16]:

- 25% did not retrieve any relevant documents within all the first 20 hits.
- 25% retrieved their relevant documents in the first hit.
- 37.5% retrieved their relevant documents in the first 5 hits.
- 6.25% retrieved their relevant documents between the 5^{th} and the 10^{th} hits.
- 6.25% retrieved their relevant documents between the 10^{th} and the 20^{th} hits.

Notice that the search engine was capable in most times of retrieving relevant documents within only 5 hits. The number of retrieved documents beyond the fifth hit was relatively small. In fact, since the number of relevant documents are sometimes more than one, the query recall could not be measured within $Recall^{(1)}$, but was measured with $Recall^{(5)}$. Therefore, the result we got based on retrieval within the first five hits should be considered jointly with the result of retrieval from the first hit.

In the next two sections, we will discuss a major issue within this study. We will discuss the overall effort needed to implement both solutions: ontology-based and text-based. We will discuss this in steps of implementation and see how each step affects the overall system.

[16] Refer to Appendix E for the more detailed results for all queries. The results sheet shows all recalls, precisions, and efficiency for each query.

4.3 Effort Building a Text-Based Solution

In order to establish a text-based retrieval system, the organization has to perform a few steps. These steps aim towards putting different components together in order to make them act as a single application. The steps an organization needs to perform are as follows:

1. Provide textual documentation of all projects to be involved in the experience base repository.
   It is important that all types of documentation of a certain project be made available in textual format. However, different types of artifacts will surely exist in non-textual formats, e.g., GUI sketches or database schemas coming out of the design phase. Similarly, build phases will produce source code.

   However, engineering artifacts are not necessarily as useful in being part of the textual repository compared to managerial artifacts. Source code, for example, does not provide information about the workflow or about itself, except in case of comments. Therefore, source code does not provide textual metadata about itself that is useable in searches.

   On the other hand, project documentation does not consist only of engineering artifacts. Project plans, workflow information, assessment and inspection documents, etc. are very likely to be text-based (e.g., written in word processors or as web pages). Since these documents are major parts of the whole project documentation, they provide very important information about the project. In fact, from a project manager perspective, documents like assessment and planning are more informative than engineering artifacts.

   In our case of developing our experimental project, we did not develop any engineering artifacts (source code). However, the system was documented for every single engineering and managerial artifact. This included workflow

information, effort, skills required, responsibilities, quality, related artifacts, etc. The overall documentation of a project of this size consumed about a week of workload carried out by a single person. This does not take into consideration the designing and planning of the structure, responsibilities, and workflow information of the project. In real life, this information would be available before the project is under development.

2. Index the experience base repository, using a search utility.
   To make the search engine capable of finding documents, the repository has to be indexed. Search engines are available commercially or free of charge and easily installed on an organization intranet. Like web search engines, the local search engine will need to crawl the repository at the organization site and index all documents available. Such functionality does not place any extra load on the organization, as it is available with the search engine.

   On the other hand, it is necessary to re-index the repository when major changes occur. In other words, it is necessary to re-index new pages after they have been added to the repository. The functionality of re-indexing a textual repository (usually measured in minutes) is not time consuming, nor does it impose extra load on the organization.

   For our project, this step did not take a lot of effort. We used the capabilities of a web search engine that is already operational. We only needed to make our documentation available via a web server so that the search engine could reach it. The search engine needed less than 20 minutes to index our repository, which was 9 megabytes over a T1 connection.

3. Build an interface for issuing queries against the indexed repository.
   Every search engine provides different means of accessing it. One of the most common ways to issue queries and obtain results is through web technology. Search engines usually provide different types of web forms capable of

accessing the indexed repository. Web forms provide an uncomplicated way to interact with the search engine. They are easily modifiable, easy to configure, and can work within any Internet browser.

Similarly, search engines also provide the functionality of constructing a result page, e.g., web page. After obtaining the list of hits, search engines provide means for the user to display this list. In most cases, web pages are the most commonly used, and similar to their forms, they are easily configurable. In general, building an interface for issuing queries and obtaining results is not a hard task.

Again, for our experiment, the search web interface was ready for public use and only needed to be added to a web page. The organization providing the interface had already configured and made the code available. The code mainly issues a pre-defined request to the engine based on the user search keywords. Adding this code to our web page was a trivial task.

4.4 Effort Building an Ontology-Based Solution

Building an ontology-based solution is completely different from building a text-based solution. The technology behind the two solutions does not overlap in any way. Hence, almost no effort spent on building a text-based solution can be re-used in building the ontology-based one. We have already discussed the details involved in many of the aspects of an ontological solution. Therefore, we will briefly summarize major steps towards building an ontology-based solution inside the organization. Within the points, we will emphasize the effort involved in carrying out these steps.

1. Build a well-defined ontology description of the domain. If the organization is not reusing an existing ontology[17], this is the most fundamental step it needs to perform. Ontology definitions – as discussed earlier – serve as the template for any knowledge to be added later to the knowledge base. This is in addition to its role of defining and describing the domain itself by identifying all its entities and their relationships.

   The effort of building ontology for a domain (e.g., a software development process) is not a trivial task. In order to build one, the knowledge management group has to have a complete understanding of the domain. This includes identifying all entities, relationships, and rules governing the domain behavior. A formal documentation of the domain (e.g., a book describing the bases of a process model) can provide substantial help. Such documentation helps identify basic entities and points out relationships. When building an ontology definition, it is important to build one that is as complete as possible.

   For our experiment, we needed to model the unified process model as our domain. As no detailed knowledge of the model existed, we had to go through being acquainted with the model and its concepts before we modeled it. For this, we used the unified process model book [Booch 99]. It took one person about three to four weeks to understand the model and map it down to an ontological definition. This included identifying all concepts and setting relationships for each concept.

2. Model new knowledge based on ontology definitions. As new experience becomes available inside the organization, it needs to be added to the experience repository. Based on what we discussed in Chapter 2, this experience is a new instantiation of concepts defined within the ontology.

[17] Refer to Chapter 2 for a discussion on sharable ontologies.

This step is the most effort-consuming for the knowledge management group. One reason is that this task needs to be re-performed every time new information becomes available. Another reason is that mapping new information to their corresponding ontology concepts requires a deep understanding of the ontology and the modeled information.

Newly introduced information will be instances of concepts defined in the ontology definitions. They will also have rules and relationships governed by the rules defined in the ontology definitions. For this reason, modeling new information according to the ontology definitions is time and effort consuming. This is because the approach has the following requirements:

- Identify every entity within the new information, and map it to a predefined concept in the ontology definition.
- Identify the values of all properties of each instance according to attributes defined in the concept definition. This is the most effort and time consuming step. The effort of modeling new information depends on the size of the information itself. However, considering that a complete project is to be modeled, it is very likely that the amount of information is large. A normal project will have a big number of documents, tasks, artifacts, activities, iterations, workflow information, etc. All these instances will have to have their entire attribute values set for them.

Our experimental project – which is relatively small – had about 132 instances that needed to be assigned to their proper concepts. Each had an average of five attributes for each (depending on the concept) to which we needed to assign a value. For that, we needed to develop a tool to allow setting values in an easy way. The tool will automatically generate F-logic syntax that corresponds to our settings. With one person effort, programming this tool

required about a month of work. It required a few days to set the values for all instances.

Within this step, certain features of ontology facilitate modeling new information. Rules defined in the ontology govern the behavior of all instances of concepts involved in the rule. Therefore, it is not necessary to rewrite new rules for new instances introduced to the knowledge base. Since ontology defines new instances to belong to a certain concept, these instances acquire all attributes and rules set for that concept.

3. Provide the logical inference engine. The inference engine works like a server side that handles ontology definitions and modeled information. From a retrieval perspective, this is similar to the text search engine that retrieves results. However, an inference engine requires all data to be loaded into the memory before it can work on it. This is because the inference engine needs to execute many logical operations on instances to evaluate a possible match with the issued query.

   We used a commercial inference engine for our implementation: OntoBroker©. This engine is capable of reading RDF and F-Logic syntax of the ontology. Setting up this tool and being acquainted with its API's required about two weeks. This included writing wrapper code for the API's and batch files for automating tasks.

4. Build user interfaces that can access the experience base. Logical inference engines provide means of listening to user requests. However, unlike search engines, results are not necessarily made available as a web page. It is up to the organization to build user interfaces to communicate with the server. The interface may be either web-based or normal tools.

   A user interface should provide two main functionalities for the user:

- Ability to add, remove, or modify ontology definitions and the modeled information. The usage of this interface will most probably be used upon inserting new experience in the experience base. User interfaces should be able to add information using the knowledge representation language used within the system, as we discussed in Chapter 3.

- Ability to issue queries against the experience base and obtain results in a user-friendly manner.

As we have mentioned above, our experiment needed such user interfaces to manipulate the ontology. Building the tool for inserting, removing, and modifying ontological entries took about a month with one-person effort. However, issuing queries against the inference engine was easy, and it took only a few days to develop a tool for it. This is because the engine comes with its own APIs, while manipulating F-Logic files based on ontology definitions was our own problem.

It is important to remember that a user interface should reflect the structure of the experience base. In other words, it should present the structure of the experience base to the user in a way that assists him or her in understanding its concepts and relations. This will help the user to issue meaningful queries and understand how retrieved information can be put to use.

By now, we have developed a good idea of the effort needed to implement any of the two solutions inside the organization. In the next section, we will summarize major factors that make a difference between the two approaches. The summary will focus on the advantages and disadvantages that exist in both.

4.5 Ontology-Based Vs. Text-Based Solutions

In a few points, we will discuss practical differences between the ontology-based and text-base solutions. These points represent the major issues to consider when choosing a solution, as they represent major differences in behavior between the two approaches. The approaches have differences not only in degrees of accuracy but also in the manner of understanding and dealing with the stored knowledge. The following points summarize the differences:

1- Semantics do not exist at all in textual descriptions of knowledge. This is one of the most important issues, as entities are not identified in textual representations. Hence, different instances can never relate to each other, have relations or rules, or hold values for attributes.
2- Numeral semantics do not exist in textual description. For this reason it is not possible to issue queries with numeral relationships. For example, it is not possible in textual search to search for experience of more than 7 years (Experience < 7). Benchmark question number 5 illustrates a similar case.
3- Text search is affected by any spelling mistakes. This is true in the case of mistakes in the repository or search keywords. Ontology-based retrieval, on the other hand, is most likely to take place through user interfaces that have preset choices.
4- Text retrieval is affected when using synonym lists. Depending on the domain, the organization can decide on synonym lists they want to use. Question number 3 is a good example, showing how the use of the word “Degree” in the query returned much fewer documents than the word “Education” (Appendix E).
5- Important relationships like inheritance provide basic categorization of entities (e.g., engineering and managerial documents). Such a feature does not exist in textual search. This is one of the most important features available in ontological retrieval. It is possible to retrieve instances based on their inheritance relationship by searching for the base class. For example, it is easier to find employees who

can program in an object-oriented language than searching for them by languages, one by one.

6- The use of abbreviations in documentation might prevent the search engine from finding relevant documents. On the other hand, user interfaces in ontology-based retrieval force the user to use specific names, i.e., controlled vocabulary using menus and lists. However, the abbreviations problem can be solved with the use of synonym lists.

7- Values given to attributes in ontological descriptions can be either numbers (e.g. 90%) or instances of scales (e.g. low, mid, hi). However, in textual descriptions, it is possible for the documenter to use plain English to describe an entity (e.g. “almost complete” instead of “90% complete”). This leaves the door open to the use of many possible keywords in textual searches.

8- Due to lack of semantics in textual search, it is not possible to identify common-sense relationships. For example, the ability to manage large projects surely includes ability to manage smaller ones.

9- Ontological retrieval gives back direct lists of instances or values as an answer to a query. On the other hand, text retrieval returns documents, and the user has to search within the document to find an exact needed piece of information. Benchmark question number 14 is a very good example of this case.

10- In textual descriptions, it is not possible to apply an attribute to an instance. Even though it is most likely that both keywords might exist in the same document, they still do not relate. This creates an issue in case a document is describing more than one similar entity where many attributes are mentioned that relate to many instances. This is a straightforward functionality in ontologies. Question 5 presents a good example of this case.

11- Information needed to answer a specific query can exist in more than one document. Although these relevant documents exist, they might not be retrieved, as the search engine has no means of relating them together. Questions 14 and 16 provide a very good example of such a case.

12- Non-accurate modeling of the domain in ontology-based systems can result in returning wrong results. In fact, this can result in not obtaining any results back, although needed information might actually exist. Considering the amount of information to be modeled, and the effort of modeling a domain, chances of making mistakes can be high. We have discussed this earlier; however, it is important to mention this point again from a retrieval perspective. This is because the power of ontological retrieval is based on logical inferences of an assumed real-world model that might not be complete.
13- Finally, efforts to build, grow, and maintain each system is a major difference. Based on our previous discussion of effort, it is visible how ontology-based systems require more effort than text based. We looked at this issue earlier in this chapter. However, we will also look at this from a different perspective in the last chapter, and at how this effort can affect the organization decision.

4.6 Summary

In this chapter, we have looked at the results of our experimental setup. We have looked at the benchmark questions that we used to test the retrieval efficiency of the text-based system. We also went through the steps of implementing both retrieval solutions: ontology-based and text-based. Through this, we have quantitatively examined the effort needed to carry out each step. We compared this effort to the one we have put into implementing our experimental project. Finally, we have identified a number of points that make a difference in the retrieval capabilities between the two mentioned approaches.

Chapter 5

Conclusions

5.1 Introduction

After what we have discussed in the previous chapters, we should be able by now to judge on different aspects of the two solutions. Although textual retrieval has performed very well in many cases, it is clear that its accuracy cannot measure up to ontology-based retrieval. This is true in many cases of use, although, in other cases, textual retrieval has given great results. Different factors discussed earlier were behind the different results we obtained.

On the other hand, it is obvious that the effort of laying out an ontology-based solution infrastructure is very costly. The cost is not limited to starting the system (although it is higher at that time) but it continues over all the time that the system is functioning. The cost is the continuous effort needed to model new information, in addition to the time and organizational resources needed to carry out this task. Tasks that need to be performed once are also non-trivial and take real effort to perform before the system is able to function for the first time. One-time efforts are made to build basic ontology definitions for the domain, install inference engines, and build user interfaces.

However, all these aspects are also subject to modifications and further enhancements as the organization's requirements develop.

5.2 Ontology-based Vs. Text-based: An Economic Decision

It is an economic decision to choose one of the two approaches. After looking at all the inputs and outcomes of the two approaches, an organization should be able to make a decision. That decision will have to be made carefully, as it requires large allocations of resources to implement a knowledge management solution. This is even more true in the case of an ontology-based system.

The accuracy of an ontology-based system is a good factor that can push towards implementing a knowledge management solution. However, the resources that must be allocated for such a system might not be available, depending on the size of the organization. These resources can include any of the following:

- Personnel to carry out tasks continuously, e.g., to model new information into the experience base
- Time and money to build and maintain the system software, either the inference engine or the user interfaces
- Ability to initiate new experience management process parallel to the normal development process already existing inside the organization.

The size of the company is not the only factor that matters. The organization's maturity directly affects its ability to initiate an experience management process inside the organization. The quality of the workflow and execution of normal development processes gives an indication of the organization ability to introduce a new experience management process. In order to do so, the organization has to be able to carry out this process concurrently with its normal development process.

At the same time, the organization has to create coordination and synchronization mechanisms that will ensure collaboration between the experience management and software development processes. This is especially true in these two cases:

1. The development process feeds the experience management process with newly gained knowledge.
2. The experience management process feeds the development process with input about best practices to be applied in new projects.

5.3 Ontology-based Vs. Text-based: A Final Word

Based on the last section, the organization has to answer two questions before safely introducing the proper type of experience management activities:

1. Is the amount of experience and accuracy obtained from an inexpensive and simple text retrieval system sufficient to meet the organization's needs?
2. Does the organization's maturity and control over its processes allow it to handle a newly introduced experience management process and maintain it in parallel with the normal development process already in place?

Based on the answers, the organization should be able to work out the balance between the results obtained from one system and the effort required to build the other. At the same time, experience management processes have to develop over time to provide the best support possible for the organization development process.

5.4 Summary

In this chapter, we looked at both solutions from an organizational perspective. We have emphasized the importance of balancing between the accuracy required and the effort and cost the organization is willing to invest. We also emphasized the ability of an organization to conduct an experience management process along with its normal development process.

5.5 Future Work

In this study, we introduced an exploratory case study that simulates a real-life development process. Based on the experimental developed project, we built two retrieval systems: ontology-based and text-based. The goal of the study was to compare the ontological approach, with its logical retrieval capabilities, to the textual approach with its easy-to-implement advantage. The two approaches were chosen as the two ends of a scale on which other retrieval systems with other capabilities exist, e.g., CBR and Database systems.

There are many aspects of this study that introduce future work, either to obtain more accurate results or to widen the scope of the study. Following are three areas where more work can take place in the future:

1. Implement a “precision/recall vs. effort” comparison based on a large-scale case study of real-life projects developed by a software development organization. A good addition to this study would be to include the artifacts as part of the experience repository and examine the effect of that on the retrieval results.

2. Examine the effects of optimizing the textual and the ontological repositories. This can be done by improving on the ontologies of the ontology-based system to make them as complete as possible. As for the text-based system, we can improve on it by using advanced techniques of text search. It would be enlightening to examine the effects of using synonym lists, regular expression, and natural language processing.

3. Examine other approaches on the scale we discussed in Chapter 1 (Text, Databases, CBR, Ontology). Since these systems have different capabilities, it would be helpful to know how they compare to one another in their retrieval capabilities and the effort required to build an experience system based on any of them.

BIBLIOGRAPHY

[Aamodt 94] Case Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches
Agnar Aamodt, Enric Plaza
AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-59 (1994).

[Basili 95] The Experience Factory.
Victor Basili, Dieter Rombach and Gianluigi Caldiera.
Encyclopedia of Software Engineering – 2 Volume Set
Page 469 – 476, John Wiley & Sons Inc., 1994.

[Becker 63] Information Storage and Retrieval, Tools, Elements, Theories.
Joseph Becker, Robernt M. Hayes
John Wiley & Sons, 1963

[Booch 99] The Unified Software Development Process.
Grady Booch, Ivar Jacobson and James Rumbaugh
Addison-Wesley, 1999.

[CMMI 02] Capability Maturity Model Integration (CMMISM), Version 1.1
Software Engineering Institute, March 2002.

[Dellen 97] Enriching Software Process Support by Knowledge-based Techniques.
B. Dellen, F. Maurer, J. Muench, M. Verlage
International Journal of Software Engineering and Knowledge Engineering, Vol. 7, No. 2, pp. 185-215, 1997.

[Genesereth 92] Knowledge interchange format
Michael Genesereth, Richard Fikes
Reference Manual - Version 3.0, Stanford University, 1992

[Gentner 83] Structure Mapping – a Theoretical Framework for Analogy.
Dedre Gentner, Cognitive Science, Vol. 7, s.155-170, 1983.

[Gruber 92] Ontolingua: A Mechanism to Support Portable Ontologies.
Thomas R. Gruber. Knowledge System Laboratory, Stanford University, 1993.

[Gruber 93-A] A Translation Approach to Portable Ontology Specifications.
Thomas R. Gruber. Knowledge Acquisition, 5(2): pp 199-220, 1993.

[Gruber 93-B] Toward Principles for the Design of Ontologies Used for knowledge sharing.
Thomas R. Gruber. Knowledge System Laboratory, Stanford University, 1993.

[Harrocks 00] The Ontology Inference Layer
I. Harrocks, D. Fensel et-al, August, 2000

[Harter 86] Online Information Retrieval
Stephen P. Harter, Academic Press, 1986.

[Henninger 03] Tool Support for Experience-Based Software Development Methodologies
Scott R. Henninger
Advances in Computers, Vol 59, pp 29-82, 2003.

[Humphrey 89] Managing the Software Process
Watts Humphrey, SEI Series in Software Engineering
Addison-Wesley, 1989.

[Kifer 95] Logical foundations of object-oriented and frame-based languages.
Michael Kifer, George Lausen and James Wu.
Journal of the ACM, 42(4): pp 741-843, July 1995.

[Kochen 74] Principles of Information Retrieval
Manfred Kochen
Melville Publishing Company, 1974.

[Leake 96] CBR in Context: The Present and Future
David B. Leake
In Leake, D., ed., 1996, Case-Based Reasoning: Experience, Lessons, and Future Directions, Menlo Park: AAAI Press/MIT Press, 1996.

[NTNU 01] Proposal for Center of Excellence: NOCO: NOmadic COmputing for Virtual Organizations – Mobile, Distributed and Multimedia ICT Applications.
By NTNU: Norwegian University of Science and Technology., 2001.

[Pao 89] Concepts of Information Retrieval
Miranda Lee Pao
Libraries Unlimited, 1989.

[Ruhe 01] Learning Software Organizations
Gunther Ruhe
Handbook of Software Engineering and Knowledge Engineering (S.K. Chang, ed.), World Scientific Publishing Co., Inc. 2001

[Schank 82] Dynamic Memory: A Theory of Remembering and Learning in Computers and People.
Roger Schank, Cambridge University Press, 1982.

[Seaman 01] A Prototype Experience Management System for a Software Consulting Organization
C. Seaman, M. Mendonca, V. Basili, and Y. Kim
Proceedings of SEKE 2001 Conference, Buenos Aires, Argentina, June 2001.

[Sutton 95] An Analysis of Process Languages
Stanley M. Sutton, Jr., Peri L. Tarr, Leon J. Osterweil
CMPSCI Technical Report 95-78, 1995

[Sutton 97] The Design of a Next-Generation Process Language
Stanley M. Sutton, Jr., Leon J. Osterweil
Software Engineering – ESEC/FSE 97, LNCS 1301, Springer, 1997, pp. 142-158

Appendix A

Skill Ontology

Following is a printout of the skill ontology (written in F-Logic) used in this study. The ontology is printed in a hiracical view only to show the inheritance realationship between different concepts. Instances declarations are left in a non-bold font to distingush them from concepts.

==========

"Root" // This line is not part of the file as “Root” is always assumed to exist
"Formal Education"::"Root".
"Certifications"::"Formal Education".
"APlus":“Certifications".
"Cisco Certified":“Certifications".
"IBM Certified":“Certifications".
"Java Certified":“Certifications".
"MCDA":“Certifications".
"MCP":“Certifications".
"MCSD":“Certifications".
"MCSE":“Certifications".
"MCT":“Certifications".
"Novell Certified":“Certifications".
"OCP":“Certifications".
"UML Certified":“Certifications".
"Degree"::"Formal Education".
"Bsc":“Degree".
"Diploma":“Degree".
"Msc":“Degree".

"No Degree":“Degree".
"PhD":“Degree".
"Technical Degree":“Degree".
"Field of Study"::"Formal Education".
"Accounting":“Field of Study".
"Business":“Field of Study".
"Computer Science":“Field of Study".
"Computer/Electronic Engineering":“Field of Study".
"Electrical Engineering":“Field of Study".
"Information Systems":“Field of Study".
"MIS/DSS":“Field of Study".
"Management":“Field of Study".
"Secretaria":“Field of Study".
"Software Engineering":“Field of Study".
"Formal Training"::"Formal Education".
“External Training":“Formal Training".
"InHouse Training":“Formal Training".
"Formal Experience"::"Root".
"Jobs":: "Formal Experience".
"Implementer":“Jobs”.
"Project Manager":“Jobs”.
"Administrator":“Jobs”.
"Business Strategy Manager":“Jobs”.
"Design Manager":“Jobs”.
"Development Manager":“Jobs”.
"Educator":“Jobs”.
"Quality Controller:“Jobs”."
"Researcher":“Jobs”.
"Salesman":“Jobs”.
"Software Design Engineer":“Jobs”.
"Software Developer":“Jobs”.
"Software Tester":“Jobs”.
"Support Staff":“Jobs”.
"System Analyst":“Jobs”.
"System Architect":“Jobs”.
"Team Leader":“Jobs”.
"Technical Writer":“Jobs”.
"Testing Manager":“Jobs”.
"WorkExperienceLength"::"Formal Experience".
"1-3 Years":”WorkExperienceLength”.
"10-15 Years":”WorkExperienceLength”.
"4-6 Years":”WorkExperienceLength”.
"7-9 Years":”WorkExperienceLength”.
"Over 15 Years":”WorkExperienceLength”.
"Other Skills"::"Root".
"Artistic Abilities"::"Other Skills".
"Music":"Artistic Abilities".
"Other Artistic Abilities":"Artistic Abilities".
"Painting - Drawing":"Artistic Abilities".
"Business Experience"::"Other Skills".
"Banking":"Business Experience".
"Economics":"Business Experience".
"Finance":"Business Experience".
"Insurance":"Business Experience".
"Local Government Administration":"Business Experience".
"Mairtime Sector":"Business Experience".

"Military":"Business Experience".
"OffShore":"Business Experience".
"Processing Industries":"Business Experience".
"Public Administration":"Business Experience".

"Foreign Cultures"::"Other Skills".

"American (US) Business Culture":"Foreign Cultures".
"Arabic (culture)":"Foreign Cultures".
"Chinees (culture)":"Foreign Cultures".
"French (culture)":"Foreign Cultures".
"German (culture)":"Foreign Cultures".
"Italian (culture)":"Foreign Cultures".
"Japanees (culture)":"Foreign Cultures".
"Other Cultures":"Foreign Cultures".
"Spanish (culture)":"Foreign Cultures".

"Languages"::"Other Skills".

"Arabic (language)":"Languages".
"Chinees (language)":"Languages"
"English (language)":"Languages".
"French (language)":"Languages".
"German (language)":"Languages".
"Italian (language)":"Languages".
"Japanees (language)":"Languages".
"Other Languages":"Languages".
"Russian (language)":"Languages".
"Spanish (language)":"Languages".

"Natural Sciences"::"Other Skills".

"Chemistry":"Natural Sciences".
"Civil Engineering":"Natural Sciences".
"Electronics":"Natural Sciences".
"Mathematics":"Natural Sciences".
"Mechanical Engineering":"Natural Sciences".
"Other Natural Sciences":"Natural Sciences".
"Physics":"Natural Sciences".
"Research Skills":"Natural Sciences".
"Statistics":"Natural Sciences".

"Other Sciences"::"Other Skills".

"Commercial Field":"Other Sciences".
"Humanistic Field":"Other Sciences".
"Medical Science":"Other Sciences".
"Other non-natural Science":"Other Sciences".
"Pedagogics":"Other Sciences".
"Psychology":"Other Sciences".

"OutDated"::"Other Skills".

"IBM AS/400":"OutDated".
"Netscape Server":"OutDated".
"Simulation":"OutDated".

"Technology and Methods":: "Root".

"AI - Artificial Intelligence"::"Technology and Methods".

"CBR - Case Based Reasoning":"AI - Artificial Intelligence".
"Constraint Based Reasoning":"AI - Artificial Intelligence".
"Distributed AI/BlackBoards":"AI - Artificial Intelligence".
"Genetic Algorithms":"AI - Artificial Intelligence".
"Intillegent Agents":"AI - Artificial Intelligence".
"KA- Knowledge Aquisition":"AI - Artificial Intelligence".
"KBE - Knowledge ?Based Engineering":"AI - Artificial Intelligence".
"KR - Knowledge Representation":"AI - Artificial Intelligence".

"Logic Programming/Rule based Reasoning":"AI - Artificial Intelligence".
"Natural Language Understanding":"AI - Artificial Intelligence".
"Neural Networks":"AI - Artificial Intelligence".
"Scheduling/Planning":"AI - Artificial Intelligence".

"Computas Frameworks"::"Technology and Methods".

"BRIX Based Development":" Computas Frameworks".
"Beans Based Development":" Computas Frameworks".
"COMworks Based Development":" Computas Frameworks".
"FrameSolutions Concepts":" Computas Frameworks".
"FrameSolutions Process Modeling":" Computas Frameworks".
"Sara96 Based Development":" Computas Frameworks".
"Sara98 Based Development":" Computas Frameworks".

"General SW technology"::"Technology and Methods".

"Complex System Architecture":"General SW technology".
"Cryptography":"General SW technology".
"DataBase Systems":"General SW technology".
"Network Infrastructure Security":"General SW technology".
"Networking":"General SW technology".
"Product Development":"General SW technology".
"Product Release":"General SW technology".
"Screen Scraping":"General SW technology".
"Support":"General SW technology".
"System Analysis":"General SW technology".
"Testing and Testing Techniques":"General SW technology".
"Web Security":"General SW technology".
"dotNet":"General SW technology".
"system Design":"General SW technology".

"Graphics Design and Multimedia"::"Technology and Methods".

"3D/4D Graphics Programming":" Graphics Design and Multimedia".
"Animation Techniques":" Graphics Design and Multimedia".
"Graphical Design":" Graphics Design and Multimedia".
"Icon Design":" Graphics Design and Multimedia".
"Qt GUI Toolkit":" Graphics Design and Multimedia".

"OT - Object Technology"::"Technology and Methods".

"Architecture (OLE/DNA/COM/DCOM)":"OT - Object Technology".
"COM (OLE/DNA/COM/DCOM/)":"OT - Object Technology".
"CORBA":"OT - Object Technology".
"Enterprise Java":"OT - Object Technology".
"OOD (OMT/UML)" :"OT - Object Technology".
"OODB":"OT - Object Technology".
"Patterns/FrameWorks":"OT - Object Technology".

"Programming languages"::"Technology and Methods".

"ADA":"Programming languages".
"ASP":"Programming languages".
"Assembly":"Programming languages".
"COBOL":"Programming languages".
"CPP":"Programming languages".
"CSharp":"Programming languages".
"FORTRAN":"Programming languages".
"HTML":"Programming languages".
"Java":"Programming languages".
"LISP":"Programming languages".
"Other Programming Languages":"Programming languages".
"Pascal":"Programming languages".
"Perl":"Programming languages".
"Prolog":"Programming languages".

"SQL/PLSQL":"Programming languages".
"SmallTalk":"Programming languages".
"Visual Basic":"Programming languages".
"Visual Cpp":"Programming languages".
"XML":"Programming languages".

"Project Management"::"Technology and Methods".

"DSDM":"Project Management".
"Documentation":"Project Management".
"GDPM":"Project Management".
"GQM":"Project Management".
"Managing Projects":"Project Management".
"Packing and Distribution":"Project Management".
"Project Management for FrameSolutions Projects":"Project Management".
"Project Management for Large Projects":"Project Management".
"SPICE":"Project Management".
"Software Process Improvment":"Project Management".
"Training":"Project Management".

"UI - User interface"::"Technology and Methods".

"CX user interaction principles":"UI - User interface".
"MS Windows Design Guidelines":"UI - User interface".
"MacOS Design Guidelines":"UI - User interface".
"Motif Design Guidelines":"UI - User interface".
"UI Architecture":"UI - User interface".
"UI Design":"UI - User interface".
"UI Evaluation":"UI - User interface".
"UI Knowledge Aquisition":"UI - User interface".
"Web Design Guidelines":"UI - User interface".

"Other"::"Technology and Methods".

"CAD":"Other".
"Mobile/PDA Technology":"Other".
"Writting Offers":"Other".

"Tools and Applications"::"Root".

"AI Development"::"Tools and Applications".

"CBR Express":"AI Development".
"CPLEX":"AI Development".
"ILOG Schedule":"AI Development".
"ILOG Solver":"AI Development".
"KADS":"AI Development".
"ND SmartElements":"AI Development".

"Database Development Environments"::"Tools and Applications".

"GemStone":"Database Development Environments".
"Ingress":"Database Development Environments".
"MS Access":"Database Development Environments".
"ObjectStore":"Database Development Environments".
"Oracle":"Database Development Environments".
"SQL Server":"Database Development Environments".
"Sybase":"Database Development Environments".

"Graphics, Multimedia and Presentation"::"Tools and Applications".

"Adobe PhotoShop":"Graphics, Multimedia and Presentation".
"Animation ShopPro":"Graphics, Multimedia and Presentation".
"CorelDraw":"Graphics, Multimedia and Presentation".
"MS Powerpoint":"Graphics, Multimedia and Presentation".
"Other Graphics Packages":"Graphics, Multimedia and Presentation".
"Paint Shop Pro":"Graphics, Multimedia and Presentation".
"VISIO":"Graphics, Multimedia and Presentation".

"Office Applications"::"Tools and Applications".
"Acrobat Writer":"Office Applications".
"MS Excel":"Office Applications".
"MS Outlook":"Office Applications".
"MS Word":"Office Applications".
"Star Office":"Office Applications".
"Operating Systems"::"Tools and Applications".
"DOS":"Operating Systems".
"Linux":"Operating Systems".
"MacOS":"Operating Systems".
"Novell":"Operating Systems".
"Unix":"Operating Systems".
"Windows":"Operating Systems".
"Programming environments"::"Tools and Applications".
"CSharp.Net":"Programming environments".
"Cpp Builder":"Programming environments".
"Delphi":"Programming environments".
"GenSym G2":"Programming environments".
"IBM VisualAge for Java":"Programming environments".
"IBM VisualAge for Smalltalk":"Programming environments".
"IBM WebSphere Application Developer":"Programming environments".
"JBuilder":"Programming environments".
"Lotus Notes":"Programming environments".
"MS FrontPage":"Programming environments".
"MS Visual Basic":"Programming environments".
"MS Visual Cpp":"Programming environments".
"MS Visual Interdev":"Programming environments".
"Symantec Visual Cafe 2.1":"Programming environments".
"Visual Basic.Net":"Programming environments".
"Visual Cpp.Net":"Programming environments".
"dotNetFrameWork":"Programming environments".
"Project Management Tools"::"Tools and Applications".
"Computas GDPM":"Project Management Tools".
"MS Project":"Project Management Tools".
"Utility Programs, Tools and others"::"Tools and Applications".
"Apache":"Utility Programs, Tools and others".
"CD-ROM Production":"Utility Programs, Tools and others".
"CVS":"Utility Programs, Tools and others".
"Crystal Reports":"Utility Programs, Tools and others".
"Install Shield":"Utility Programs, Tools and others".
"Install Shield-Java Edition":"Utility Programs, Tools and others".
"Matlab":"Utility Programs, Tools and others".
"Other Utility Program":"Utility Programs, Tools and others".
"Rational Clear Case":"Utility Programs, Tools and others".
"RoboHelp":"Utility Programs, Tools and others".
"Visual SourceSafe":"Utility Programs, Tools and others".

Appendix B

Process Ontology

Following is a printout of the *Unified Process Model* ontology. The printout was indented –which should not have any effect – in order to show hierarchical structure of the ontology. The relations' part of the ontology shows the range of relations (attributes) that instances of each concept can have. Opposite to setting attributes – which uses a single arrow (->>) – specifying a relation range uses double arrows (=>).

For Example, a relation like: *"MiloStone"[Artifacts=>>"Artifact Set"].*

Reads like: An instance of the concept Milestone, i.e. any milestone, will have an attribute called Artifacts that takes a value of any instance of the "Artifact Set" Concept. When setting this attribute for a milestone instance, we would use:

"Reservation System MiloStone"[Artifacts ->>"Borrowing System Artifacts"].

Where

"Borrowing System Artifacts"[Artifacts->>{"Borrowing System Assesment Document", "Borrowing System Interface","Borrowing System Plan Document", "Borrowing System Schema","Borrowing System Test Document"}].

==========

// Concepts

"Root" // This line is not part of the file as “Root” is always assumed to exist

```
"Activity"::"The Unified Process".
        "Assesment"::"Activity".
        "Planning"::"Activity".
"Activity Set"::"The Unified Process".
"Architectural Pattern"::"The Unified Process".
        "Layer"::"Architectural Pattern".
                "Application General Layer"::"Layer".
                "Application Specific Layer"::"Layer".
                "Middleware Layer"::"Layer".
                "System Software Layer"::"Layer".
        "Black Board"::"Architectural Pattern".
        "Brother"::"Architectural Pattern".
        "Filters"::"Architectural Pattern".
        "Horizantal-Vertical Metadata"::"Architectural Pattern".
        "MVC"::"Architectural Pattern".
        "Pipes"::"Architectural Pattern".
"Architecture"::"The Unified Process".
"Artifact"::"The Unified Process".
        "Engineering Artifact"::"Artifact".
        "Management Artifact"::"Artifact".
"Artifact Set"::"The Unified Process".
"Class"::"The Unified Process".
"Component"::"The Unified Process".
"Cycle"::"The Unified Process".
"Iteration"::"The Unified Process".
"MiloStone"::"The Unified Process".
"Patterns"::"The Unified Process".
        "Analysis Pattern"::"Patterns".
        "Architectural Patterns"::"Patterns".
        "Hardware Patern"::"Patterns".
                "Client-Server"::"Hardware Patern".
                "Peer-To-Peer"::"Hardware Patern".
                "Three Tier"::"Hardware Patern".
        "Design Pattern"::"Patterns".
        "Organizational Pattern"::"Patterns".
"Phase"::"The Unified Process".
        "Construction"::"Phase".
        "Elaboration"::"Phase".
        "Inception"::"Phase".
        "Transition"::"Phase".
"Release"::"The Unified Process".
"Requirement"::"The Unified Process".
        "Functional Requirement"::"Requirement".
        "Non-Functional Requirement"::"Requirement".
"Risk"::"The Unified Process".
        "Architecture Risk"::"Risk".
        "New Technology Risk"::"Risk".
        "Performance Risk"::"Risk".
        "Right System Risk"::"Risk".
"Role"::"The Unified Process".
"Software Representation"::"The Unified Process".
        "Analysis Model"::"Software Representation".
```

```
        "Deployment Model"::"Software Representation".
        "Design Model"::"Software Representation".
        "Domain Model"::"Software Representation".
        "Implementation Model"::"Software Representation".
        "Test Model"::"Software Representation".
        "UseCase Model"::"Software Representation".
"StakeHolder"::"The Unified Process".
        "Agency"::"StakeHolder".
        "Customer"::"StakeHolder".
        "Funding Authorities"::"StakeHolder".
        "Manager"::"StakeHolder".
        "Sales Person"::"StakeHolder".
"System"::"The Unified Process".
"Type"::"The Unified Process".
"WorkFlow"::"The Unified Process".
        "Core WorkFlow"::"WorkFlow"
                "Analysis"::"Core WorkFlow".
                "Design"::"Core WorkFlow".
                "Implementation"::"Core WorkFlow".
                "Requirements"::"Core WorkFlow".
                "Test"::"Core WorkFlow".
        "Iteration WorkFlow"::"WorkFlow"
"Worker"::"The Unified Process".
        "Analyst"::"Worker".
        "Architect"::"Worker".
        "Designer"::"Worker".
        "Implementer"::"Worker".
        "Project Manager"::"Worker".
        "Tester"::"Worker".
        "User"::"Worker".
```

// Relations

```
"Activity"[ Effort=>>"Effort"; Skill=>>"Skill"; Tool=>>"Tools and Applications"].

"Activity Set"[Activity=>>"Activity"].

"Agent"[AppliedSkill=>>"Formal Experience"; Degree=>>"Degree";
        Documentation=>>STRING; FieldofStudy=>>"Field of Study";
        OtherSkill=>>"Other Skills";PossibleJob=>>"Jobs"; PreviousJob=>>"Jobs";
        SkillLevel=>>"LevelScale"; TechnologySkill=>>"Technology and Methods";
        ToolSkill=>>"Tools and Applications";WorkExperience=>>"Work ExperienceLength"].

"Architecture"[Component=>>"Component";Pattern=>>"Architectural Pattern";
                Requirement=>>"Requirements";System=>>"System"].

"Artifact"[Availability=>>INTEGER;Quality=>>"LevelScale";Type=>>"Type"].

"Artifact Set"[Artifacts=>>"Artifact"].

"Assesment"[OutcomeResults=>>"LevelScale";ResultsDocumentation=>>"Management
Artifact"].

"Component"[Class=>>"Class"].
```

"Core WorkFlow"[Effort=>>"Effort";NextTask=>>"Core WorkFlow";PreviousTask=>>"Core WorkFlow"; Skill=>>"Skill";iteration=>>"Iteration";outputArtifact=>>"Artifact"].

"Cycle"[ConcludesWith=>>"Release";ConsistsOf=>>"Phase"].

"Elaboration"[BaseLine=>>"Artifact Set"].

"Engineering Artifact"[BugsPerKLOC=>>INTEGER;TestType=>>"Testing Type"].

"Iteration"[Activity=>>"Activity";AnalysisWF=>>"Analysis";Assesment=>>"Assesment";
DesignWF=>>"Design";ImplementationWF=>>"Implementation";Order=>>INTEGER;
Phase=>>"Phase";Planning=>>"Planning";RequirementWF=>>"Requirements";TestWF=>>"Test"].

"Layer"[System=>>"System"].

"MiloStone"[Artifacts=>>"Artifact Set"].

"Phase"[Iterations=>>"Iteration";ResultsIn=>>"MiloStone"].

"Planning"[ResultsDocumentation=>>"Management Artifact"].

"Project"[Documentation=>>STRING;ProcessModel=>>"Process Model"].

"Release"[MadeOf=>>"Artifact Set"].

"Requirements"[Activity=>>"Activity Set"].

"Role"[Skill=>>"Skill"].

"Skill"[Comment=>>STRING].

"The Unified Process"[Documentation=>>STRING;Project=>>"Project"].
"Type"[Comment=>>STRING].

"WorkFlow"[Assesment=>>"Assesment";Planning=>>"Planning"].

"Worker"[Agent=>>"Agent";ParticipateIn=>>"WorkFlow";ResponsibleFor=>>"Activity Set";Role=>>"Role"].

Appendix C

Project Ontology

Following is a printout of the ontology files describing the study-developed project. The description is based on the process and skill ontologies discussed earlier. The ontology is mainly divided into two parts: i) projects instances ii) instances attributes. The instances are listed inside the file in alphabetical order.

==========

// Instances

"Allow Checking material in and out":"Functional Requirement".
"Allow Limitless Data":"Non-Functional Requirement".
"Allow Search":"Functional Requirement".
"Allow insertion of new material":"Functional Requirement".
"Allow web-based access":"Functional Requirement".
"Assign Modules to employees ":"Iteration".
"Basic ER schemas":"Engineering Artifact".
"Basic GUI sketches":"Engineering Artifact".
"Book Manager Test Document":"Engineering Artifact".
"Books Manager":"Engineering Artifact".
"Borrowing System":"System".
"Borrowing System Artifacts":"Artifact Set".
"Borrowing System DB Schemas":"Artifact Set".
"Borrowing System Development Activities":"Activity Set".
"Borrowing System Development Assesment":"Assesment".
"Borrowing System Interface":"Artifact".
"Borrowing System Interfaces":"Artifact Set".
"Borrowing System Plan Document":"Management Artifact".

"Borrowing System Release":"Release".
"Borrowing System Schema":"Engineering Artifact".
"Borrowing System Test Document":"Engineering Artifact".
"Borrowing System Test Documents":"Artifact Set".
"Borrowing System Use Case Model":"UseCase Model".
"Build Books Manager":"Implementation".
"Build Classification Manager":"Implementation".
"Build User Insertion Interfaces":"Implementation".
"Build Web-based System EJBs":"Implementation".
"Build Web-based System JSPs":"Implementation".
"Classification Manager":"Engineering Artifact".
"Classification Manager Test Document":"Engineering Artifact".
"Classification System DB Schema":"Engineering Artifact".
"Configuration for HW Requirements":"Iteration".
"Decide on needed modules ":"Iteration".
"Decide on non functional requirements ":"Analysis".
"Define Network Requirements":"Analysis".
"Define Network Usability":"Analysis".
"Define System Requirements":"Iteration".
"Define none functional requirements":"Iteration".
"Design Strategy for non functional requirements":"Design".
"Develop Borrowing Interfaces":"Implementation".
"Develop Borrowing Schema":"Design".
"Develop Borrowing System":"Iteration".
"Develop Classificatiojn System Schema":"Design".
"Develop Insertion System":"Iteration".
"Develop Reservation System":"Iteration".
"Develop Reservations Interfaces":"Implementation".
"Develop Reservations Schema":"Design".
"Develop User DB Schema":"Design".
"Develop Web-based System":"Iteration".
"Develop Web-based System GUI sketches":"Design".
"Develop web-based System Navigation Structure":"Design".
"Direct DB Effect":"Non-Functional Requirement".
"Elaboration Phase Artifacts":"Artifact Set".
"final requirements document ":"Engineering Artifact".
"GUI Sketches":"Type".
"GUI Standards Confirmation Test":"Test".
"GUI Standards Confirmation Test Document":"Engineering Artifact".
"Insertin System Test Documents":"Artifact Set".
"Insertion System":"System".
"Insertion System Artifacts":"Artifact Set".
"Insertion System Assesment Document":"Management Artifact".
"Insertion System DB Schemas":"Artifact Set".
"Insertion System Development Activities":"Activity Set".
"Insertion System Development Assesment":"Assesment".
"Insertion System Development Plan":"Planning".
"Insertion System Interfaces":"Artifact Set".
"Insertion System Plan Document":"Management Artifact".
"Insertion System Release":"Release".
"Insertion System Use Case Model":"UseCase Model".
"Install Borrowing system and test it":"Iteration".
"Install Insertion system and test it ":"Iteration".
"Install Network infrastructure ":"Iteration".
"Install Reservation system and test it":"Iteration".
"Install Web-Based system and test it":"Iteration".

"LS-Analyst":"Analyst".
"LS-Architect":"Architect".
"LS-Concstruction":"Construction".
"LS-Cycle":"Cycle".
"LS-Designer":"Designer".
"LS-Implementer":"Implementer".
"LS-Manager":"Manager".
"LS-Tester":"Tester".
"LS-Text-based System":"Non-Functional Requirement".
"LS-User":"User".
"Overall Design document produced":"Iteration".
"Produce the requirements document":"Design".
"Reservatin System Test Documents":"Artifact Set".
"Reservation System":"System".
"Reservation System Artifacts":"Artifact Set".
"Reservation System Assesment Document":"Management Artifact".
"Reservation System DB Schemas":"Artifact Set".
"Reservation System Development Activities":"Activity Set".
"Reservation System Development Assesment":"Assesment".
"Reservation System Development Plan":"Planning".
"Reservation System Interfaces":"Artifact Set".
"Reservation System Plan Document":"Management Artifact".
"Reservation System Release":"Release".
"Reservation System Schema":"Engineering Artifact".
"Reservation System Test Document":"Engineering Artifact".
"Reservation System Use Case Model":"UseCase Model".
"Test Book Manager":"Test".
"Test Borrowing Interfaces":"Test".
"Test Classification Manager":"Test".
"Test Reservations Interfaces":"Test".
"Test Web-based System":"Test".
"use case diagrams":"Engineering Artifact".
"University of Calgary":"Customer".
"Up-Time of 95%":"Non-Functional Requirement".
"User Insertion Interface":"Engineering Artifact".
"User Insertion Interface":"Artifact".
"Users DB-Schema":"Engineering Artifact".
"Web-based System":"System".
"Web-based System Artifacts":"Artifact Set".
"Web-based System Assesment Document":"Management Artifact".
"Web-based System Development Activities":"Activity Set".
"Web-based System Development Assesment":"Assesment".
"Web-based System Development Plan":"Planning".
"Web-based System EJBs":"Engineering Artifact".
"Web-based System GUI Sketches":"Engineering Artifact".
"Web-based System Interfaces":"Artifact Set".
"Web-based System Navigation Structure":"Engineering Artifact".
"Web-based System Plan Document":"Management Artifact".
"Web-based System Release":"Release".
"Web-based System Test Document":"Engineering Artifact".
"Web-based System Test Documents":"Artifact Set".
"Web-based System Use Case Model":"UseCase Model".
"Web-based System pages":"Engineering Artifact".

// Setting instances attributes

"Allow Checking material in and out"[Project->>{"Library System"}].

"Allow Limitless Data"[Project->>{"Library System"}].

"Allow Search"[Project->>{"Library System"}].

"Allow insertion of new material"[Project->>{"Library System"}].

"Allow web-based access"[Project->>{"Library System"}].

"Book Manager Test Document"[Availability->>{100};Quality->>{"Meduim"};Type->>{"Word Document"};BugsPerKLOC->>{7};Project->>{"Library System"}].

"Borrowing System"[Project->>{"Library System"}].

"Books Manager"[BugsPerKLOC->>{2};TestType->>{"Integration Test","System Test"};Project->>{"Library System"};Type->>{"PL/SQL Code"};Quality->>{"High"};Availability->>{90}].

"Borrowing System DB Schemas"[Project->>{"Library System"};Artifacts->>{"Borrowing System Schema"}].

"Borrowing System Development Activities"[Project->>{"Library System"};Activity->>{"Borrowing System Development Assesment","Borrowing System Development Plan"}].

"Borrowing System Development Assesment"[Effort->>{"Normal Effort"};OutcomeResults->>{"Meduim"};ResultsDocumentation->>{"Borrowing System Assesment Document"};Project->>{"Library System"};Skill->>{"Managing Projects","Product Development","Project Management for Large Projects"};Tool->>{"MS Project"}].

"Borrowing System Development Plan"[Effort->>{"Normal Effort"};Skill->>{"Project Management for Large Projects"};Tool->>{"MS Project"};ResultsDocumentation->>{"Borrowing System Plan Document"};Project->>{"Library System"}].

"Borrowing System Artifacts"[Project->>{"Library System"};Artifacts->>{"Borrowing System Assesment Document","Borrowing System Interface","Borrowing System Plan Document","Borrowing System Schema","Borrowing System Test Document"}].

"Borrowing System Interface"[Type->>{"PL/SQL Code"};Project->>{"Library System"};Quality->>{"Meduim"};Availability->>{100}].

"Borrowing System Interfaces"[Project->>{"Library System"};Artifacts->>{"Borrowing System Interface"}].

"Borrowing System Plan Document"[Type->>{"MS Project Document"};Quality->>{"Meduim"};Project->>{"Library System"};Availability->>{75}].

"Borrowing System Release"[Project->>{"Library System"};MadeOf->>{"Borrowing System Artifacts"}].

"Borrowing System Schema"[Availability->>{75};Quality->>{"Meduim"};Type->>{"DB/ER Diagram"};TestType->>{"Integration Test","System Test"};Project->>{"Library System"}].

"Borrowing System Test Document"[Availability->>{80};Quality->>{"Meduim"};Type->>{"Text Document"};Project->>{"Library System"}].

"Borrowing System Test Documents"[Artifacts->>{"Borrowing System Test Document"};Project->>{"Library System"}].

"Borrowing System Use Case Model"[Project->>{"Library System"}].

"Build Books Manager"[Effort->>{"VeryHighEffort"};Skill->>{"DataBase Systems"};Skill@("Oracle","Product Development","SQL/PLSQL")->>{"4-6 Years"};NextTask->>{"Develop Borrowing Interfaces"};PreviousTask->>{"Build Classification Manager"};iteration->>{"Develop Insertion System"};outputArtifact->>{"Books Manager"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Build Classification Manager"[Effort->>{"Normal Effort"};Skill@("Oracle","SQL/PLSQL")->>{"1-3 Years"};NextTask->>{"Develop Borrowing Interfaces"};PreviousTask->>{"Build User Insertion Interfaces"};iteration->>{"Develop Insertion System"};outputArtifact->>{"Classification Manager"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Build User Insertion Interfaces"[Effort->>{"LowEffort"};Skill@("Oracle","SQL/PLSQL")->>{"1-3 Years"};NextTask->>{"Build Classification Manager"};PreviousTask->>{"Develop User DB Schema"};iteration->>{"Develop Insertion System"};outputArtifact->>{"User Insertion Interface"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Build Web-based System EJBs"[Effort->>{"VeryHighEffort"};Skill@("Enterprise Java","Java")->>{"10-15 Years"};Skill@("IBM VisualAge for Java","Oracle")->>{"1-3 Years"};Skill->>{"Java Certified"};NextTask->>{"Build Web-based System JSPs"};PreviousTask->>{"Develop Web-based System GUI sketches"};iteration->>{"Develop Web-based System"};Assesment->>{"Web-based System Development Assesment"};Planning->>{"Web-based System Development Plan"};Project->>{"Library System"}].

"Build Web-based System EJBs"[outputArtifact->>{"Web-based System EJBs"};Skill@("IBM WebSphere Application Developer")->>{"1-3 Years"}].

"Build Web-based System JSPs"[Effort->>{"VeryHighEffort"};Skill@("Enterprise Java","HTML","IBM WebSphere Application Developer","Java")->>{"4-6 Years"};PreviousTask->>{"Build Web-based System EJBs"};iteration->>{"Develop Web-based System"};outputArtifact->>{"Web-based System pages"};Assesment->>{"Web-based System Development Assesment"};Planning->>{"Web-based System Development Plan"};Project->>{"Library System"}].

"Classification Manager"[Availability->>{95};Quality->>{"High"};Type->>{"PL/SQL Code"};BugsPerKLOC->>{1};TestType->>{"Acceptance Test","Unit Test"};Project->>{"Library System"}].

"Classification Manager Test Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Classification System DB Schema"[Availability->>{100};Quality->>{"High"};Type->>{"DB/ER Diagram"};TestType->>{"Integration Test"};Project->>{"Library System"}].

"Develop Borrowing Interfaces"[Effort->>{"Normal Effort"};Skill@("Oracle","SQL/PLSQL")->>{"1-3 Years"};NextTask->>{"Test Borrowing Interfaces"};PreviousTask->>{"Develop Borrowing

Schema"};iteration->>{"Develop Borrowing System"};outputArtifact->>{"Borrowing System Interface"};Assesment->>{"Borrowing System Development Assesment"};Planning->>{"Borrowing System Development Plan"};Project->>{"Library System"}].

"Develop Borrowing Schema"[Effort->>{"Normal Effort"};Skill@("Oracle")->>{"4-6 Years"};PreviousTask->>{"Develop User DB Schema"};NextTask->>{"Develop Borrowing Interfaces"};iteration->>{"Develop Borrowing System"};outputArtifact->>{"Borrowing System Schema"};Assesment->>{"Borrowing System Development Assesment"};Planning->>{"Borrowing System Development Plan"};Project->>{"Library System"}].

"Develop Borrowing System"[Activity->>{"Borrowing System Development Assesment","Borrowing System Development Plan"};Assesment->>{"Borrowing System Development Assesment"};DesignWF->>{"Develop Borrowing Schema"};ImplementationWF->>{"Develop Borrowing Interfaces"};Order->>{2};Planning->>{"Borrowing System Development Plan"};TestWF->>{"GUI Standards Confirmation Test","Test Borrowing Interfaces"};Project->>{"Library System"}].

"Develop Classificatiojn System Schema"[Effort->>{"Normal Effort"};Skill@("Oracle")->>{"4-6 Years"};NextTask->>{"Build Classification Manager"};iteration->>{"Develop Insertion System"};outputArtifact->>{"Classification System DB Schema"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Develop Insertion System"[Activity->>{"Insertion System Development Assesment","Insertion System Development Plan"};Assesment->>{"Insertion System Development Assesment"};DesignWF->>{"Develop Classificatiojn System Schema","Develop User DB Schema"};ImplementationWF->>{"Build Classification Manager","Build User Insertion Interfaces"};Order->>{1};Planning->>{"Insertion System Development Plan"};TestWF->>{"GUI Standards Confirmation Test","Test Book Manager","Test Classification Manager"};Project->>{"Library System"}].

"Develop Reservation System"[Activity->>{"Reservation System Development Assesment","Reservation System Development Plan"};Assesment->>{"Reservation System Development Assesment"};DesignWF->>{"Develop Reservations Schema"};ImplementationWF->>{"Develop Reservations Interfaces"};Order->>{3};Planning->>{"Reservation System Development Plan"};TestWF->>{"GUI Standards Confirmation Test","Test Reservations Interfaces"};Project->>{"Library System"}].

"Develop Reservations Interfaces"[Effort->>{"VeryHighEffort"};Skill->>{"Bsc","Communication Skill","Computer Science","DataBase Systems","IBM WebSphere Application Developer","Java","Java Certified","Oracle","Software Developer"};NextTask->>{"Test Reservations Interfaces"};PreviousTask->>{"Develop Reservations Schema"};iteration->>{"Develop Reservation System"};outputArtifact->>{"Reservation System Interface"};Assesment->>{"Reservation System Development Assesment"};Planning->>{"Reservation System Development Plan"};Project->>{"Library System"}].

"Develop Reservations Schema"[Effort->>{"VeryHighEffort"};Skill@("Oracle","system Design")->>{"10-15 Years"};Skill->>{"IBM WebSphere Application Developer","Java","Java Certified","Msc","SQL/PLSQL"};NextTask->>{"Develop Reservations Interfaces"};iteration->>{"Develop Reservation System"};outputArtifact->>{"Reservation System Schema"};Assesment->>{"Reservation System Development Assesment"};Planning->>{"Reservation System Development Plan"};Project->>{"Library System"}].

"Develop User DB Schema"[Effort->>{"Normal Effort"};Skill@("Oracle","system Design")->>{"4-6 Years"};NextTask->>{"Build User Insertion Interfaces"};iteration->>{"Develop Insertion

System"};outputArtifact->>{"Users DB-Schema"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Develop Web-based System"[Activity->>{"Web-based System Development Assesment","Web-based System Development Plan"};Assesment->>{"Web-based System Development Assesment"};DesignWF->>{"Develop Web-based System GUI sketches","Develop web-based System Navigation Structure"};ImplementationWF->>{"Build Web-based System EJBs","Build Web-based System JSPs"};Order->>{4};Planning->>{"Web-based System Development Plan"};TestWF->>{"GUI Standards Confirmation Test","Test Web-based System"};Project->>{"Library System"}].

"Develop Web-based System GUI sketches"[Effort->>{"HighEffort"};Skill->>{"Graphical Design","HTML","MS FrontPage","system Design"};Skill@("Graphical Design","system Design")->>{"7-9 Years"};NextTask->>{"Build Web-based System EJBs","Build Web-based System JSPs"};PreviousTask->>{"Develop web-based System Navigation Structure"};iteration->>{"Develop Web-based System"};outputArtifact->>{"Web-based System GUI Sketches"};Planning->>{"Web-based System Development Plan"};Assesment->>{"Web-based System Development Assesment"};Project->>{"Library System"}].

"Develop web-based System Navigation Structure"[Effort->>{"HighEffort"};Skill->>{"MS FrontPage","system Design"};Skill@("system Design")->>{"10-15 Years"};NextTask->>{"Develop Web-based System GUI sketches"};iteration->>{"Develop Web-based System"};outputArtifact->>{"Web-based System Navigation Structure"};Assesment->>{"Web-based System Development Assesment"};Planning->>{"Web-based System Development Plan"};Project->>{"Library System"}].

"Develop Web-based System GUI sketches"[Skill->>{"Graphical Design"}].
"Direct DB Effect"[Project->>{"Library System"}].
"GUI Sketches"[Project->>{"Library System"}].
"Insertion System"[Project->>{"Library System"}].

"GUI Standards Confirmation Test"[Effort->>{"HighEffort"};Skill->>{"Testing and Testing Techniques"};iteration->>{"Develop Borrowing System","Develop Insertion System","Develop Reservation System","Develop Web-based System"};outputArtifact->>{"GUI Standards Confirmation Test Document"};Project->>{"Library System"}].

"GUI Standards Confirmation Test Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Insertin System Test Documents"[Artifacts->>{"Book Manager Test Document","Classification Manager Test Document","GUI Standards Confirmation Test Document"};Project->>{"Library System"}].

"Insertion System Artifacts"[Artifacts->>{"Book Manager Test Document","Books Manager","Classification Manager","Classification Manager Test Document","Classification System DB Schema","Insertion System Assesment Document","Insertion System Plan Document","User Insertion Interface","Users DB-Schema"};Project->>{"Library System"}].

"Insertion System Assesment Document"[Availability->>{99};Quality->>{"Low"};Type->>{"Text Document"};Project->>{"Library System"}].

"Insertion System DB Schemas"[Artifacts->>{"Classification System DB Schema","Users DB-Schema"};Project->>{"Library System"}].

"Insertion System Development Activities"[Activity->>{"Insertion System Development Assesment","Insertion System Development Plan"};Project->>{"Library System"}].

"Insertion System Development Assesment"[Effort->>{"LowEffort"};Skill@("Managing Projects")->>{"7-9 Years"};OutcomeResults->>{"Meduim"};ResultsDocumentation->>{"Insertion System Assesment Document","Insertion System Plan Document"};Project->>{"Library System"}].

"Insertion System Development Plan"[Effort->>{"Normal Effort"};Skill@("Project Manager")->>{"4-6 Years"};Tool->>{"MS Project"};ResultsDocumentation->>{"Insertion System Plan Document"};Project->>{"Library System"}].

"Insertion System Interfaces"[Artifacts->>{"Books Manager","Classification Manager"};Project->>{"Library System"}].

"Insertion System Plan Document"[Availability->>{100};Quality->>{"Meduim"};Type->>{"MS Project Document"};Project->>{"Library System"}].

"Insertion System Release"[MadeOf->>{"Insertion System Artifacts"};Project->>{"Library System"}].

"Insertion System Use Case Model"[Project->>{"Library System"}].

"LS-Analyst"[Project->>{"Library System"};Agent->>{"Alicia","Angelina"};ParticipateIn->>{"Develop Borrowing Schema","Develop Reservations Schema","Develop User DB Schema","Develop web-based System Navigation Structure"};ResponsibleFor->>{"Borrowing System Development Activities","Insertion System Development Activities","Reservation System Development Activities","Web-based System Development Activities"}].

"LS-Architect"[Project->>{"Library System"};Agent->>{"Lina"};ParticipateIn->>{"Develop Web-based System GUI sketches","Develop web-based System Navigation Structure"};ResponsibleFor->>{"Web-based System Development Activities"}].

"LS-Concstruction"[Project->>{"Library System"};ResultsIn->>{"LS-Construction MileStone"};Iterations->>{"Develop Borrowing System","Develop Insertion System","Develop Reservation System","Develop Web-based System"}].

"LS-Cycle"[ConsistsOf->>{"LS-Concstruction","LS-Elabotration","LS-Inception","LS-Transition"};Project->>{"Library System"};ConcludesWith->>{"Borrowing System Release","Insertion System Release","Reservation System Release","Web-based System Release"}].

"LS-Designer"[Project->>{"Library System"};Agent->>{"Jeniffer","Mariella"};ParticipateIn->>{"Develop Borrowing Schema","Develop Classificatiojn System Schema","Develop Reservations Schema"};ResponsibleFor->>{"Borrowing System Development Activities","Insertion System Development Activities","Reservation System Development Activities"}].

"LS-Implementer"[Agent->>{"Dima","Sandy"};Project->>{"Library System"};ParticipateIn->>{"Build Web-based System EJBs","Build Web-based System JSPs"};ResponsibleFor->>{"Web-based System Development Activities"}].

"LS-Manager"[Project->>{"Library System"}].

"LS-Project Manager"[Agent->>{"Philip"};ParticipateIn->>{"Build Books Manager","Build Classification Manager","Build User Insertion Interfaces","Build Web-based System EJBs","Build Web-based System JSPs","Develop Borrowing Interfaces","Develop Borrowing Schema","Develop Classificatiojn System Schema","Develop Reservations Interfaces","Develop Reservations Schema","Develop User DB Schema","Develop Web-based System GUI sketches","Develop web-based System Navigation Structure","GUI Standards Confirmation

Test","Test Book Manager","Test Borrowing Interfaces","Test Classification Manager","Test Reservations Interfaces","Test Web-based System"};ResponsibleFor->>{"Borrowing System Development Activities","Insertion System Development Activities","Reservation System Development Activities","Web-based System Development Activities"}].

"LS-Tester"[Project->>{"Library System"};Agent->>{"Arda"};ParticipateIn->>{"Test Book Manager","Test Borrowing Interfaces","Test Classification Manager","Test Reservations Interfaces","Test Web-based System"};ResponsibleFor->>{"Borrowing System Development Activities","Insertion System Development Activities","Reservation System Development Activities","Web-based System Development Activities"}].

"LS-Text-based System"[Project->>{"Library System"}].
"LS-User"[Agent->>{"Carmen","Rita"};Project->>{"Library System"}].

"Reservatin System Test Documents"[Artifacts->>{"Reservation System Test Document"};Project->>{"Library System"}].

"Reservation System"[Project->>{"Library System"}].

"Reservation System Artifacts"[Artifacts->>{"Reservation System Assesment Document","Reservation System Interface","Reservation System Plan Document","Reservation System Schema","Reservation System Test Document"};Project->>{"Library System"}].

"Reservation System Assesment Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Reservation System DB Schemas"[Artifacts->>{"Reservation System Schema"};Project->>{"Library System"}].

"Reservation System Development Activities"[Activity->>{"Reservation System Development Assesment","Reservation System Development Plan"};Project->>{"Library System"}].

"Reservation System Development Assesment"[Effort->>{"Normal Effort"};Skill->>{"Project Management for Large Projects"};OutcomeResults->>{"High"};ResultsDocumentation->>{"Reservation System Assesment Document"};Project->>{"Library System"}].

"Reservation System Development Plan"[Effort->>{"HighEffort"};Skill->>{"MS Project","Project Management for Large Projects"};Skill@("Project Manager")->>{"10-15 Years"};Tool->>{"MS Project"};ResultsDocumentation->>{"Reservation System Plan Document"};Project->>{"Library System"}].

"Reservation System Interface"[Availability->>{50};Quality->>{"High"};Type->>{"PL/SQL Code"};Project->>{"Library System"}].

"Reservation System Interfaces"[Artifacts->>{"Reservation System Assesment Document","Reservation System Interface","Reservation System Plan Document","Reservation System Schema","Reservation System Test Document"};Project->>{"Library System"}].

"Reservation System Plan Document"[Availability->>{40};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Reservation System Release"[MadeOf->>{"Reservation System Artifacts"};Project->>{"Library System"}].

"Reservation System Schema"[Quality->>{"Meduim"};Type->>{"DB/ER Diagram"};TestType->>{"Integration Test","Unit Test"};Project->>{"Library System"}].

"Reservation System Test Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Reservation System Use Case Model"[Project->>{"Library System"}].

"Test Book Manager"[Effort->>{"Normal Effort"};Skill->>{"Testing and Testing Techniques"};Skill@("Testing and Testing Techniques")->>{"1-3 Years"};PreviousTask->>{"Build Books Manager"};iteration->>{"Develop Insertion System"};outputArtifact->>{"Book Manager Test Document"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Test Borrowing Interfaces"[Effort->>{"Normal Effort"};Skill->>{"Testing and Testing Techniques"};PreviousTask->>{"Develop Borrowing Interfaces"};iteration->>{"Develop Borrowing System"};outputArtifact->>{"Borrowing System Test Document"};Assesment->>{"Borrowing System Development Assesment"};Planning->>{"Borrowing System Development Plan"};Project->>{"Library System"}].

"Test Classification Manager"[Effort->>{"LowEffort"};Skill->>{"Testing and Testing Techniques"};Skill@("System Architect")->>{"1-3 Years"};PreviousTask->>{"Build Classification Manager"};iteration->>{"Develop Insertion System"};outputArtifact->>{"Classification Manager Test Document"};Assesment->>{"Insertion System Development Assesment"};Planning->>{"Insertion System Development Plan"};Project->>{"Library System"}].

"Test Reservations Interfaces"[Effort->>{"VeryHighEffort"};Skill@("Software Tester","System Analyst","Testing and Testing Techniques")->>{"10-15 Years"};PreviousTask->>{"Develop Reservations Interfaces"};iteration->>{"Develop Reservation System"};outputArtifact->>{"Reservation System Test Document"};Assesment->>{"Reservation System Development Assesment"};Planning->>{"Reservation System Development Plan"};Project->>{"Library System"}].

"Test Web-based System"[Effort->>{"Normal Effort"};Skill->>{"Testing and Testing Techniques"};PreviousTask->>{"Build Web-based System EJBs","Build Web-based System JSPs"};iteration->>{"Develop Web-based System"};outputArtifact->>{"Web-based System Test Document"};Assesment->>{"Web-based System Development Assesment"};Planning->>{"Web-based System Development Plan"};Project->>{"Library System"}].

"The Unified Software Process"[Project->>{"Library System"}].
"University of Calgary"[Project->>{"Library System"}].
"Up-Time of 95%"[Project->>{"Library System"}].
"Web-based System"[Project->>{"Library System"}].

"User Insertion Interface"[Availability->>{99};Type->>{"Java Code","PL/SQL Code"};BugsPerKLOC->>{10};Quality->>{"Meduim"};TestType->>{"Installation Test","Unit Test"};Project->>{"Library System"}].

"Users DB-Schema"[Availability->>{100};Quality->>{"High"};Type->>{"DB/ER Diagram"};Project->>{"Library System"}].

"Web-based System Artifacts"[Artifacts->>{"Web-based System Assesment Document","Web-based System EJBs","Web-based System GUI Sketches","Web-based System Navigation Structure","Web-based System Plan Document","Web-based System Test Document","Web-based System pages"};Project->>{"Library System"}].

"Web-based System Assesment Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].

"Web-based System Development Activities"[Activity->>{"Web-based System Development Assesment","Web-based System Development Plan"};Project->>{"Library System"}].

"Web-based System Development Assesment"[Effort->>{"VeryHighEffort"};Skill->>{"Product Development","Project Management for FrameSolutions Projects"};Skill@("Software Engineering")->>{"10-15 Years"};OutcomeResults->>{"High"};ResultsDocumentation->>{"Web-based System Assesment Document"};Project->>{"Library System"}].

"Web-based System Development Plan"[ResultsDocumentation->>{"Web-based System Plan Document"};Project->>{"Library System"};Skill->>{"Project Management for Large Projects"};Skill@("Managing Projects")->>{"4-6 Years"};Effort->>{"VeryHighEffort"}].

"Web-based System EJBs"[Quality->>{"High"};Type->>{"Java Code"};BugsPerKLOC->>{10};TestType->>{"Integration Test","System Test","Unit Test"};Project->>{"Library System"}].

"Web-based System GUI Sketches"[Availability->>{100};Quality->>{"High"};Type->>{"Image"};Project->>{"Library System"}].

"Web-based System Interfaces"[Artifacts->>{"Web-based System EJBs","Web-based System pages"};Project->>{"Library System"}].

"Web-based System Navigation Structure"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};TestType->>{"External Function Test","System Test"};Project->>{"Library System"}].

"Web-based System Plan Document"[Availability->>{100};Quality->>{"High"};Type->>{"MS Project Document"};Project->>{"Library System"}].

"Web-based System Release"[MadeOf->>{"Web-based System Artifacts"};Project->>{"Library System"}].

"Web-based System Test Document"[Availability->>{100};Quality->>{"High"};Type->>{"Text Document"};Project->>{"Library System"}].
"Web-based System Test Documents"[Artifacts->>{"Web-based System Test Document"};Project->>{"Library System"}].

"Web-based System Use Case Model"[Project->>{"Library System"}].

"Web-based System pages"[Availability->>{99};Quality->>{"High"};Type->>{"Java Code"};BugsPerKLOC->>{15};TestType->>{"Installation Test","System Test","Unit Test"};Project->>{"Library System"}].

"Dima"[AppliedSkill ->> "Software Tester";Degree ->> "Bsc";FieldofStudy ->> "Computer Science";PossibleJob ->> "Software Tester";SkillLevel ->> "Meduim";TechnologySkill ->> "CPP";TechnologySkill ->> "CSharp";TechnologySkill ->> "Visual Basic";WorkExperience ->> "4-6 Years"].

"Lina"[Degree ->> "Bsc";FieldofStudy ->> "Management";OtherSkill ->> "Communication Skill";PossibleJob ->> "Development Manager";PreviousJob ->> "Development Manager";PreviousJob ->> "Team Leader";SkillLevel ->> "High";TechnologySkill ->> "CPP";TechnologySkill ->> "Managing Projects";ToolSkill ->> "SQL Server";WorkExperience ->> "7-9 Years"].

"Samer"[Degree ->> "Bsc";FieldofStudy ->> "Computer Science";PossibleJob ->> "Development Manager";SkillLevel ->> "Low";TechnologySkill ->> "Web Security";TechnologySkill ->> "Network Infrastructure Security"].

"Alicia"[AppliedSkill@("System Analyst")->>{"4-6 Years"};Degree@("Computer Science")->>{"Bsc"};FieldofStudy->>{"Computer Science","Software Engineering"};OtherSkill->>{"French (language)"};PossibleJob->>{"Software Developer","System Analyst"};PreviousJob->>{"System Analyst"};SkillLevel@("System Analyst")->>{"Meduim"};TechnologySkill->>{"CPP","System Analysis","Visual Cpp","system Design"};ToolSkill->>{"MS Visual Cpp"};WorkExperience->>{"7-9 Years"}].

"Angelina"[AppliedSkill->>{"Software Design Engineer","Software Developer"};Degree@("Computer Science")->>{"Bsc"};FieldofStudy->>{"Information Systems"};PossibleJob->>{"Researcher","Software Design Engineer","Software Developer"};SkillLevel@("Software Engineering")->>{"High"};TechnologySkill->>{"Product Development","Software Process Improvment"};WorkExperience@("Software Design Engineer")->>{"1-3 Years"}].

"Arda"[AppliedSkill->>{"Testing Manager"};Degree@("Computer Science")->>{"Bsc"};Degree@("Software Engineering")->>{"Msc"};FieldofStudy->>{"Computer Science","Computer/Electronic Engineering","Software Engineering"};OtherSkill->>{"French (language)","Italian (culture)","Spanish (language)"};PossibleJob->>{"Development Manager","Educator","Software Tester"};PreviousJob->>{"Educator"};SkillLevel@("DataBase Systems","Enterprise Java","Java")->>{"High"};TechnologySkill->>{"Enterprise Java","Testing and Testing Techniques"};ToolSkill->>{"IBM WebSphere Application Developer"};WorkExperience@("Educator")->>{"Over 15 Years"}].

"Carmen"[AppliedSkill->>{"Researcher","Software Developer"};Degree@("MIS/DSS")->>{"Bsc"};Degree@("Software Engineering")->>{"Msc"};FieldofStudy->>{"MIS/DSS","Software Engineering"};PossibleJob->>{"Implementer","Researcher","Software Developer"};SkillLevel@("Italian (language)")->>{"Meduim"};SkillLevel@("Java")->>{"High"};TechnologySkill->>{"DataBase Systems","Enterprise Java","Java","OOD (OMT/UML)","SQL/PLSQL"};ToolSkill->>{"IBM VisualAge for Java","IBM WebSphere Application Developer","MS FrontPage"};WorkExperience->>{"1-3 Years"}].

"Violin"[AppliedSkill->>{"Development Manager","System Architect"};AppliedSkill@("Project Manager")->>{"1-3 Years"};Degree@("Computer Science")->>{"PhD"};FieldofStudy->>{"Computer Science","MIS/DSS"};OtherSkill->>{"Music"};OtherSkill->>{"Arabic (culture)","Arabic (language)"};PossibleJob->>{"Development Manager","Software Design Engineer"};PreviousJob->>{"Development Manager"};SkillLevel@("Software Design Engineer","System Analyst","System Architect","Complex System Architecture")->>{"High"};TechnologySkill->>{"Architecture (OLE/DNA/COM/DCOM/...)","COM (OLE/DNA/COM/DCOM/)","CPP","Complex System Architecture","DataBase Systems","Managing Projects","OOD (OMT/UML)","Project Management for Large Projects","SQL/PLSQL","UI Architecture","UI Design","system Design"};ToolSkill->>{"CSharp.Net","MS Project","Oracle","Rational Clear Case","SQL Server","Visual Cpp.Net"};WorkExperience->>{"7-9 Years"}].

"Philip"[AppliedSkill@("Project Manager")->>{"1-3 Years"};AppliedSkill->>{"Educator","Software Developer"};Degree@("Computer Science")->>{"Bsc"};Degree@("Software Engineering")->>{"Msc"};FieldofStudy->>{"Computer Science","Software Engineering"};OtherSkill->>{"Arabic (culture)","Arabic (language)","Music"};PossibleJob->>{"Project Manager","Development Manager","Educator","Software Design Engineer","Software Developer"};PreviousJob->>{"Educator","Software Developer"};SkillLevel->>{"High"};TechnologySkill->>{"COM (OLE/DNA/COM/DCOM/)","DataBase Systems","Enterprise Java","HTML","KBE - Knowledge ?Based Engineering","OOD (OMT/UML)","OODB","Other Programming Languages","SQL/PLSQL","system Design"};ToolSkill->>{"CVS","GemStone","IBM VisualAge for Java","IBM WebSphere Application Developer","MS Project","Oracle"};WorkExperience->>{"4-6 Years"}].

Appendix D

Developed Project

Following is the textual documentation of the developed project example used in this study. The documentation should explain all the aspects of development that took part in the study. It is listed in here for two purposes:

1- Only as a reference documentation for the project study example. ***It is not intended to be read as part of the study*** except as a reference or to have an overview of the structure of the project.

2- This documentation is the exact textual documentation that was used in the textual retrieval testing. All documents were indexed by the search engine

A softcopy of the above mentioned documentation can be found on the web at: http://sern.cpsc.ucalgary.ca/~philip/ThesisSite/index.html, other relevant and irrelevant documentation can also be accessed from the same site. In addition, there is a web form that can be used to search inside all documentation found on the site[18].

[18] For better viewing of the site, Internet Explorer and a minimum of 1024 X 768 resolution is recommended.

1. The Library System

The Library System discussed in here is a project - owner assumed to be UofC - which aims finally to build and install a complete library system in the institution. There is no prior system for this one, hence, this is system is to be analyzed, designed, built and implemented from scratch. The system is to be installed in a distributed environment where multiple users are expected to use the system. it should be able to manage different users accessing the system including anonymous users in the case of the web-based part.

1.1 System Structure

The system is made out of four main sub-systems, each of which is designed to go over one single iteration within the construction phase, in the order they are listed:

- The Insertion system: Which is probably the biggest. This system is responsible for the insertion of all users and books in tot he system. This means that other systems will rely heavily on it as it contains all basic data used in them. It will take care of users and books configuration matters.
- The Borrowing system: Which is the system that is responsible for checking books in and out for users.
- The Reservation system: Which is responsible for reserving material and calling books that are checked out already.
- The web-based system: Which is a web-enabled interface that uses the back end of the original system to allow user web-access to library contents.

The complete project will be developed using one central database (Oracle) that is located on a central server. All other client machines will not have any local information on them except for the client software, all processing is done on the server. Windows 2000 based machines will be used on both sides. Oracle CASE tools will be used to develop the client side application, while Enterprise Java beans and JSP's will be the technology used for building the web-enabled system.

The system in general should allow all common sense functionality of a library system plus other features that the customer would like to have. The insertion system should be flexible and able to handle different kinds of books and will only be concerned with only the classification system proposed. It should also be able to handle all types of transactions (add, remove, modify) on both the users and books database. The web-enabled system should allow search capabilities as it is considered one of the basic requirements of the interface and should use the same back end server in order to make changes on-the-fly.

1.2 Development Process Model

In order to build the complete application, we will be using the unified process model as our process model that will control the life cycle of the library system. The inception phase will handle all the analysis and requirements issues in an incremental approach (as this is one of the UP approaches) while iterating through two iterations. the Elaboration phase will be responsible for defining the modules and artifacts that need to be developed and assigning them to their respective developers. Basic design and documentation documents on the overall look and feel, technology and tools used are to be created.

The construction phase is the phase where the skeleton will start to cover up and the systems of the application are going to be developed. The system is going to go through 4 iterations, one iteration for the development of each sub-system mentioned above. This phase will make sure that all artifacts are built including design and build artifacts plus all management, testing and assessment documents. The Transition phase will be the phase where the system is to be implemented at the customer site; this includes tasks of installing networks, software and carrying out configuration tasks.

The system activities, tasks, iterations, artifacts, agents, technology and skills required to carry out each task are all documented in their corresponding documents. Please find your way to the document discussing the part of the system that interests you, or use the search form provided.

Note: As this is an example project where the experiment is intended to measure the accuracy of the text retrieval; more attention was given to the process in the construction phase, and only a general description was given to what happens in other phases. This has been done because all questions and queries that will be used in the experiment will address information happening in the construction phase only, as the construction phase contains enough tasks, artifacts and activities to work on.

2. Phases

2.1 Inception Phase

The Inception phase is the phase where the initial work for the library system development takes place. The following iterations in this phase take place:

- Iteration 1: Configuration for HW Requirements
  This includes tasks like:
  - Define Network Usability
  - Define Network Requirements

- Iteration 2: Define none functional requirements
  This includes tasks like:
  - Decide on non functional requirements
  - Design Strategy for non functional requirements
- Iteration 3: Define System Requirements
  This includes tasks like:
  - Produce the requirements document

This Phase ends with a milestone called the LS-Inception Milestone, which is made of all the documents and artifacts produced throughout the phase and previous phases, such as the final requirements document for the system.

2.2 Elaboration Phase

The Elaboration phase is the phase where an overlook of the system is possible and it starts to have a skeleton and modules needed are more obvious. The following iterations in this phase takes place:

- Iteration 1: Decide on needed modules
- Iteration 2: Assign Modules to employees
- Iteration 3: Overall Design document produced
  This is where the theme, technologies, tools ... etc to be used in the development are specified.

This Phase ends with a milestone called the LS-Elaboration Milestone, which is made of all the documents and artifacts produced throughout the phase and previous phases, such as use case diagrams, basic ER schemas and basic GUI sketches.

2.3 Construction Phase

The Construction phase is one of the most important phase and comes third in the phases of the unified model. In this project, most of the work is being done in this phase. The construction phase is made of four iterations, each sub-system of the four sub-systems will take one single iteration to complete. The iterations of this phase are:

1. Develop Insertion System
2. Develop Borrowing System
3. Develop Reservation System
4. Develop Web-based System

At the end of this phase we will have a milestone in the development of the library system project, mile stone is called the LS-Construction Milestone, which will be

made out of all milestones artifacts of previous milestones plus the artifacts that where developed during the construction phase.

2.4 Transition Phase

The Transition phase is the phase where the system actually is put to work and is being used by its users. The following iterations in this phase takes place:

- Iteration 1: Install Network infrastructure
- Iteration 2: Install Insertion system and test it
- Iteration 3: Install Borrowing system and test it
- Iteration 4: Install Reservation system and test it
- Iteration 5: Install Web-Based system and test it

This Phase ends with a milestone called the LS-Transition Milestone, which is made of all the documents and artifacts produced throughout the phase and previous phases. This milestone represents the end of the project as it is being installed at the user side, support is now provided.

3. The Insertion System

The insertion system is one of the main four sub-systems in the whole project as its existence is essential for other subsystems to work. Artifacts from this system will interact with artifacts and other data structures from other subsystems. It will be developed in the first iteration of the construction phase and is planned to stay under development for only one iteration. The insertion system is responsible for the following

- User Management: This includes the addition, modification and deletion of users who are allowed to use the system. Different users with different privileges will exist. The system will be built so that different ranks of users will exist, such as: system administrators, managers and counter employees. This will not need to include the web-user as they do not need to login.
- Book Management: This includes all functionality necessary to check-in new books into the library system, modify any item entry and delete any books that no longer exist. The same functionality will include setting the properties of the books like its location, classification number, author information, publisher... etc.
- Classification System: This part is built in at design time and will have no interface to modify t as it is not an applicable idea. The classification system in use is the congress library system. All data structures and artifacts are built in a way that it will always use this system of classification. The data structure for this component is to be built first as other functionalities will depend on it.

Assessment and planning activities were carried out for this sub-system. Please follow these links for more information about the Tasks, Artifacts, and Activities included within the development of this system.

3.1 Tasks

The insertion system was developed by carrying out the following tasks:

- Develop the user database schema
    - Output Artifact: User Database schema
    - Description: This is the task where all the design of the database that will make the table infrastructure for the users, users types and users privileges takes place.
    - Skills: This task required a normal effort as it was straight forward, however, a designer with experience of 4-6 years is systems design on a relational database was needed to ensure good quality. This system will be the base for developing the user interfaces needed to manage the users ad their privileges.
- Develop Classification System Schema
    - Output Artifact: Classification System Schema
    - Description: This task is about developing a the database structure and tables that will hold the information about the classification system being used in classifying books
    - Skills: This task required normal effort from a system design employee assuming an experience of 4-6 years in system design in databases. This is the next tasks after developing the user database schema and the base for starting to develop the classification system manager interfaces
- Build user insertion interfaces
    - Output Artifact: User Insertion interfaces
    - Description: The goal of this task is to develop an interface that will allow the insertion of different users with the ability to manipulate their access privileges.
    - Skills: This task was of a low effort as it was a straight forward data entry interface. It required knowledge of Oracle and PL/SQL with about 1-3 years of experience.

- Build the classification Manager

    - Output Artifact: The Classification Manager
    - Description: The goal of this task is to develop the interfaces for the classification Manager.
    - Skills: Similar to the user insertion interfaces, this task requires an experience of 1-3 years in SQL, PL/SQL and Oracle. It required normal

weight of effort for developing this interface and it depended mainly on the classification schema developed earlier.

- Build the Book Manager

    - Output Artifact: Book Manager Interfaces
    - Description: Complementary for the classification system as it uses it to insert actual books into the system
    - Skills: As this is one of the main sub-systems, it needed to be of high quality, an employee of at least 4-6 years experience was to develop this interface. We require good experience with Oracle and prior knowledge of developing applications, with good command of PL/SQL. With providing this, it still needed lots of work and effort to ensure high quality of this interface and that the least of bugs exist

- Test Classification Manager

    - Output Artifact: Classification Manager test document
    - Description: To test the interfaces for the classification system
    - Skills: This task was very easy to do and required basic knowledge and experience in testing techniques as these interfaces deal with a very hard solid classification system where not much manipulation is required.

- Test Book Manager

    - Output Artifact: Book Manager test document
    - Description: The goal of this task is to test the interfaces handling insertion, and modification of books
    - Skills: There is nothing that makes this interface hard, so a normal effort was spent on this task; a person with 1-3 years experience was to carry out this task.

3.2 Artifacts

The insertion system is made up from the following artifacts

- Insertion system Assessment Document: Not yet finally complete and needs revision. This document is not of a good quality as it does not cover all areas and provide a good over view of the insertion system
- Classification system DB schema: A very good design document. The design is basically an Entity relationship and table design document. it was tested using an integration test and is released to be used.
- Classification Manager test document: Very good document of high quality, covers the test process and the outcomes very well.
- Classification Manager: Still needs to fix some bugs so it can be said to be 95% complete, however, it is a very good artifact and of high quality. It was written

using PL/SQL and the average of bugs per one kilo of code is only one bug. Acceptance test and unit test was applied when testing this artifact.

- Book Manager test document: Completely done and can be considered a good document, does not cover everything but it gives an good overview
- User Insertion Interfaces: Done except for a very minor bug fixes. These artifacts were written using PL/SQL and some java code. They are of a medium or normal quality and were found to have an average of 10 bugs in each 1K of lines of code. They were tested using unit and installation test techniques.
- User DB Schema: Very good schema and does not need to be modified, completely done. Developed as an ER/DB diagram.
- The book Manager: A very good interface with an average of 2 bugs per 1K o code. This interface has been tested (using system and integration test techniques). It still needs a bit of work before it can be released (almost 90%). Written with PL/SQL.
- Insertion system Plan Document: Was of a regular quality, covered all needed stuff. Document is released and used. Document was created using MS-Project.
- Insertion System Use Case Model

3.3 Activities

The insertion system had to go through some extra activities and configurations in order to complete it, these are:

- Insertion System Development Assessment: The assessment was a normal and easy one as the system was a straightforward system, however, since the system will be the base for other systems we required a 7-9 years experience in project management for the person conducting the assessment, the results were of a normal quality. The outcome of the assessment was documented in the insertion system assessment document.
- Insertion System Development Plan: As for the assessment, it was of a normal effort to plan the system. The person to lead the planning was to have a 4-6 years of experience in project management in order to plan how workflow of the system. Planning was done using MS-Project and the plan was documented as the insertion system development plan.

The insertion system was designed to go through one iteration within the construction phase, the first iteration. Upon the finish of this system/iteration a release will be made consisting of all the artifacts of this system as mentioned in the artifacts document.

4. The Borrowing System

The Borrowing system is the system that is responsible for all activities regarding the checking out and checking in the books from and into the library system. The system

will interact with other systems for user and book information and will keep track of all the material checked out and the periods of keeping them out.

The system will go through one iteration in the construction phase where it will be designed (i.e. build database structures) built, and tested. Assessment and planning activities were carried out for this sub-system. Please follow these links for more information about the Tasks, Artifacts, and Activities included within the development of this system.

4.1 Tasks

The Borrowing system was developed by carrying out the following tasks

- Develop Borrowing Schema
  - Output Artifact: Borrowing system schema
  - Description: This design task is required to build the database schema for the borrowing system. This is the base for the system that will handle checking books in and out.
  - Skills: Not very much effort was put into this one, however, for accuracy, we needed a 4-6 years experience with Oracle.
- Develop Borrowing system Interfaces
  - Output Artifact: Borrowing system interfaces
  - Description: This task finished by building the interfaces for the borrowing system which will allow checking books in and out.
  - Skills: Normal effort was put into this task. The agent carrying it out was required to have 1-3 years of experience with Oracle and PL/SQL. The task is based on the completion of the previous task mentioned which is to build the database schema for the borrowing system.
- Test The Borrowing System interfaces
  - Output Artifact: Borrowing system test document
  - Description: The goal of this task is to test the borrowing system for bugs
  - Skills: To accomplish this task we required an experienced agent to carry it out, someone knowledgeable with testing and testing techniques. It required normal effort to carry out this task.

4.2 Artifacts

The Borrowing system is made up from the following artifacts

- Borrowing system Test Document: not yet completely done as it is missing some parts. Considered a normal test document
- Borrowing System Schema: Was made as an ER/DB Diagram. A normal design document, however, missing some major parts (about 75%) complete. Was tested using integration test with other systems and a system test.

- Borrowing system Plan document: An MS-Project document, Medium quality as it needs some modifications
- Borrowing system Interfaces: The interfaces were written using PL/SQL. Interfaces are done and released. Good quality
- Borrowing System Use Case Model

4.3Activities

The Borrowing system had to go through some extra activities and configurations in order to complete it, these are:

- Borrowing System Development Assessment: The assessment of the borrowing system was also normal and required a regular effort of r such a project. The outcome of the assessment was satisfactory and covered many areas of the development o the system. The assessment document is identified as the Borrowing system Assessment document and was developed as an MS-Project plan document. The leader of the assessment process was needed to be experienced in large project managements and development of products.
- Borrowing System Development Plan: The plan needed to be put by an experienced personnel specially in managing large projects. The plan was made as an MS-Project plan like other systems and was considered of a normal task load for a plan for a project of this size. The result document is called the Borrowing system development plan

The borrowing system was designed to go through one iteration within the construction phase, the second iteration. Upon the finish of this system/iteration a release will be made consisting of all the artifacts of this system as mentioned in the artifacts document.

5. The Reservation System

The Reservation system is the sub-system responsible for reserving books and material for people booking them ahead of time and for calling material that has been checked out already. There will be a database structure for only this system but will of course talk to the other data structures from other system. The system has gone through a design stage and a build stage like the other systems plus a testing stage for the interfaces developed.

Assessment and planning activities were carried out for this sub-system. Please follow these links for more information about the Tasks, Artifacts and Activities included within the development of this system.

5.1 Tasks

The insertion system was developed by carrying out the following tasks

- Develop the user database schema
    - Output Artifact: User Database schema
    - Description: This is the task where all the design of the database that will make the table infrastructure for the users, users types and users privileges take place.
    - Skills: This task required a normal effort as it was straight forward, however, a designer with experience of 4-6 years is systems design on a relational database was needed to ensure good quality. This system will be the base for developing the user interfaces needed to manage the users ad their privileges.
- Develop Classification System Schema
    - Output Artifact: Classification System Schema
    - Description: This task is about developing a the database structure and tables that will hold the information about the classification system being used in classifying books
    - Skills: This task required normal effort from a system design employee assuming an experience of 4-6 years in system design in databases. This is the next tasks after developing the user database schema and the base for starting to develop the classification system manager interfaces
- Build user insertion interfaces
    - Output Artifact: User Insertion interfaces
    - Description: The goal of this task is to develop an interface that will allow the insertion of different users with the ability to manipulate their access privileges.
    - Skills: This task was of a low effort as it was a straight forward data entry interface. It required knowledge of Oracle and PL/SQL with about 1-3 years of experience.
- Build the classification Manager
    - Output Artifact: The Classification Manager
    - Description: The goal of this task is to develop the interfaces for the classification Manager.
    - Skills: Similar to the user insertion interfaces, this task requires an experience of 1-3 years in SQL, PL/SQL and Oracle. It required normal weight of effort for developing this interface and it depended mainly on the classification schema developed earlier.
- Build the Book Manager
    - Output Artifact: Book Manager Interfaces
    - Description: Complementary for the classification system as it uses it to insert actual books into the system
    - Skills: As this is one of the main sub-systems, it needed to be of high quality, an employee of at least 4-6 years experience was to develop this interface. We require good experience with Oracle and prior knowledge of

developing applications, with good command of PL/SQL. With providing this, it still needed lots of work and effort to ensure high quality of this interface and that the least of bugs exist

- Test Classification Manager
  - Output Artifact: Classification Manager test document
  - Description: To test the interfaces for the classification system
  - Skills: This task was very easy to do and required basic knowledge and experience in testing techniques as these interfaces deal with a very hard solid classification system where not much manipulation is required.
- Test Book Manager
  - Output Artifact: Book Manager test document
  - Description: The goal of this task is to test the interfaces handling insertion, and modification of books
  - Skills: There is nothing that makes this interface hard, so a normal effort was spent on this task, a person with 1-3 years experience was to carry out this task.

5.2 Artifacts

The Reservation system is made up from the following artifacts

- Reservation System Assessment document: A very good document assessing this sub-system. Document is final and released.
- Reservation system Interface: This interface is half done till now, however, what we have is of a high quality, as other subsystems, this was developed using PL/SQL.
- Reservation System Plan document: Not even half way done, but the plan seems to be solid and good. This is not an Ms-Project document but normal text (MS-Word)
- Reservation system Schema: Normal Schema, like other systems it is an ER/DB Document and was tested using system test and unit test.
- Reservation System Test Document: The Test document is a very good one and shows lots of what is missing in the reservation system, document is complete.
- Reservation System Use Case Model

5.3 **Activities**

The Reservation system had to go through some extra activities and configurations in order to complete it, these are:

- Reservation System Development Assessment: A large project management experienced person was to carry out this task. Normal effort was spent on it; however, it turned out that we did get a high quality assessment document that covered a lot about the reservation system. The output document is identified as the reservation system assessment document.

- Reservation System Development Plan: Planning the work of this system took very high effort as it had to do with other systems and making sure they are complete enough to work with. A highly skilled person was to work on this assuming experience of 10-15 years in project management. Plan has been made using MS-Project and is identified as the reservation system plan document.

The reservation system was designed to go through one iteration within the construction phase, the third iteration. Upon the finish of this system/iteration a release will be made consisting of all the artifacts of this system as mentioned in the artifacts document.

6. The Web-Based System

The Web-Based system is the final system to be developed within the library system project. This system is aimed towards providing access functionality to users over the web to the books database of the library. Users will use this system to search and check the books available in the library. This system will have to deal with all other systems as it uses same data, except that the interface is different in this case. The system will be built based on Enterprise Java technologies and the web pages will be built using Java server pages.

The system will go through some design phases to make sure that the navigation structure and the look and feel of the system are user friendly and consistent. The second phase will be to build the enterprise java beans so that they use the same data structures developed earlier for the use of other systems. JSP's will be built later depending on the functionality provided by the EJB's. The development of this system is considered of high effort as it is not a small system plus the tackling of data structures developed initially for the use in other systems.

Assessment and planning activities were carried out for this sub-system. Please follow these links for more information about the Tasks, Artifacts, and Activities included within the development of this system.

6.1 Tasks

The Web-Based system was developed by carrying out the following tasks

- Develop Navigation Structure
  - Output Artifact: Web-Based Navigation system structure
  - Description: The goal of this task is to develop the structure of how the system will look like and how different pages are going to link to each other in a user-friendly fashion.

    - Skills: Since this is very related to design, and lots of issues are to be considered as this system will use all other systems; a well skilled agent is to carry out this task. he/she should be 10-15 years experienced with system design and architecture with very good knowledge using Microsoft FrontPage. This navigation structure will be the base for developing the GUI sketches.
- Develop Web-Based System GUI Sketches
    - Output Artifact: Web-Based System GUI Sketches
    - Description: The goal of this task is to develop the actual look of the pages for the whole web system based on the architecture developed from the first task. These pages are to be programmed later using JSP's.
    - Skills: The effort for carrying out this task was not tremendous but it was more than what we can call a normal effort. Graphical and graphics design skills were to exist in the person carrying out the task in addition to knowledge of HTML and MS FrontPage plus system design experience for about 7 years.
- Build Web-based systems EJB's
    - Output Artifact: Web-based systems EJB's
    - Description: One of the most important tasks in the whole system. This is where the back end of the of the whole system is being developed where entity beans are to communicate with existing system structures and tables. These beans are to provide the functionality required from the JSP's on the client side by communicating with the system tables.
    - Skills: A very sensitive task as it uses already existing tables plus providing computing functionalities to the JSP's on the client side. Hence, a very high effort was put on to the implementation of this task awhile in the same time a very well experienced agent was put to it. Very good knowledge in Enterprise Java is required with an experience of 10-15 years, plus being a java certified. Agent is also to have 1-3 years experience using Oracle, IBM VisualAge for Java and IBM WebSphere application developer.
- Build Web-Based System JSP's
    - Output Artifact: Web-Based System JSP's
    - Description: The goal of this task is to build the JSP's that will talk to the EJB's developed in the previous task and at the same time will confirm to the navigation structure and the GUI sketches developed earlier in the design stage.
    - Skills: Again, a very high effort was spent on building these pages. Like building the EJB's, building these JSP's requires Enterprise Java knowledge, plus knowledge in HTML and the use of IBM WebSphere Application developer with at least an experience of 4-6 years
- Test GUI Standards Confirmation
    - Output Artifact: GUI Standards confirmation test
    - Description: The goal of this task is to test how the GUI used in the pages are consistent with each others and with the conventions used

    - Skills: Although not expected, this task was hard to carry out, probably due to graphical type of objects and measures a tester has to deal with. Good testing techniques are required to carry out this task.
- Test Web-Based System
    - Output Artifact: Web-Based System Test Document
    - Description: The goal of this task is to test the complete web-based system and ensure that all EJB's and JSP's work fine with a good speed performance. The test should make sure no bugs are available.
    - Skills: Although building the backend of this system was hard, however, testing it was of a normal effort as it is only using the system with no programming effort, it was not easy but it was not hard. The agent carrying out the task was required to have proven skills in testing systems and applying different testing methodologies.

6.2 Artifacts

The Web-Based system is made up from the following artifacts

- Web-based system Assessment document: Very good document, ready and complete, covers lots of aspects of the project
- Web-based system EJB's: Developed using Java Enterprise technology. Output was of a very high quality specially that they communicate with tables of other systems, bugs were found (10/1K LOC) but they still functioned in a very good way. EJB's were tested using system, unit and integration tests techniques.
- Web-based system GUI Sketches: Very good Sketches, provide good guidelines for the designer and programmer. They are completely developed. The sketches are in an image format
- Web-based System Plan Document: Document is ready and complete and is considered very good. Plan for this sub-system is in MS-Project format
- Web-based System Navigation Structure: this is a text document describing the structure of the web-based system plus the navigation path through it. The document is complete and is very comprehensive and good. It was tested using external function test and system test.
- Web-based System Test Document: The test document is done and complete, a very good one of high quality. Document is of textual type
- Web-based System Pages: these are the actual pages the user see on the browser, developed based on JSP technology. These pages turned out to be of very high quality and the function pretty well. However, bugs were found in them upon testing and they average of 15 bugs per 1K of lines of code. Unit testing, System test, and installation tests were applied to these pages.
- GUI Standards confirmation test document: This is a very good document describing well the standards and errors of the web pages developed for the web-based system. The document is complete and covers all pages. Written in normal text as an MS-Word document and identified as the GUI Standards confirmation test document.
- Web-based System Use Case Model

6.3 Activities

The Web-Based system had to go through some extra activities and configurations in order to complete it, these are:

- Web-Based System Development Assessment: This assessment was a very hard one to do as the web-based system is big and complicated so it was hard to assess it. We required a high experience (at least 10 years) manager to lead the team and to possess the experience in product development and project management plus similar experience in software engineering. This paid of as the assessment of the web-based system was a very good one and the document was comprehensive. The document can be identified as the Web-based system assessment document.
- Web-Based System Development Plan: As this project is big and has to do with all other projects, planning it had to be carefully done. It required much time and effort to develop the plan for the workflow and deadlines of this sub-system and required experience in managing large projects (we required 4-6 years).

The web-based system was designed to go through one iteration within the construction phase, the fourth and final iteration. Upon the finish of this system/iteration a release will be made consisting of all the artifacts of this system as mentioned in the artifacts document.

7. Basic Requirements

In addition to the normal requirements figured out by the normal behavior expected from the system and what has been described in the other pages, the system will have the following requirements.

- The system should allow limitless data to be stored and handled
- The system Shall allow checking material in and out
- The system shall allow search capabilities
- The system shall allow insertion of new material all the time
- The system shall allow web access to the system
- The system shall have direct affect on the database upon update
- The system shall have up time up to 95%
- The system will have good performance on the web
- The GUI's shall be user friendly
- The system shall be text based, except for the web part
- The system shall allow reservation and calling of books
- The system shall allow classification of books

8. Personnel

Throughout the development of this system, many of the agents working in the organization were assigned to different tasks within the system. Each one might have

been assigned to more than one task or has played a different role in different sub-systems, or might have been responsible for more than one task or artifacts. Following are the people who participated in building the system[19]

- **Philip**
  The project manager, he has also participated in the following tasks

  - Build Books Manager
  - Build Classification Manager
  - Build User Insertion Interfaces
  - Build Web-based System EJBs
  - Build Web-based System JSPs
  - Develop Borrowing Interfaces
  - Develop Borrowing Schema
  - Develop Classification System Schema
  - Develop Reservations Interfaces
  - Develop Reservations Schema
  - Develop User DB Schema
  - Develop Web-based System GUI sketches
  - Develop web-based System Navigation Structure
  - GUI Standards Confirmation Test
  - Test Book Manager
  - Test Borrowing Interfaces
  - Test Classification Manager
  - Test Reservations Interfaces
  - Test Web-based System

  Philip also was responsible for the following

  - Borrowing System Development Activities
  - Insertion System Development Activities
  - Reservation System Development Activities
  - Web-based System Development Activities

- **Arda**

  Her main responsibility was testing. She has participated in the following tasks

  - Test Book Manager
  - Test Borrowing Interfaces
  - Test Classification Manager
  - Test Reservations Interfaces
  - Test Web-based System

[19] Names used in the example are made up and do not refer to any particular individuals.

Arda also was responsible for the following

- Borrowing System Development Activities
- Insertion System Development Activities
- Reservation System Development Activities
- Web-based System Development Activities

- **Dima, Sandy**

  They have participated in the following tasks

  - Build Web-based System EJBs
  - Build Web-based System JSPs

  They also were responsible for the following

  - Web-based System Development Activities

- **Alicia, Angelina**

  They have participated in the following tasks

  - Develop Borrowing Schema
  - Develop Reservations Schema
  - Develop User DB Schema
  - Develop web-based System Navigation Structure

  They also were responsible for the following

  - Borrowing System Development Activities
  - Insertion System Development Activities
  - Reservation System Development Activities
  - Web-based System Development Activities

- **Lina**

  She has participated in the following tasks

  - Develop Web-based System GUI sketches
  - Develop web-based System Navigation Structure

  She also was responsible for the following

  - Web-based System Development Activities

- **Jennifer, Mariella**

  They have participated in the following tasks

  - Develop Borrowing Schema
  - Develop Classification System Schema
  - Develop Reservations Schema

  They also were responsible for the following

  - Borrowing System Development Activities
  - Insertion System Development Activities
  - Reservation System Development Activities

- **Carmen, Rita**

  These two agents were asked to play the role of a normal user and provide feedbacks.

Appendix E

Textual Retrieval Results

Following are the results of the textual retrieval test described in chapter 4. The 20 questions that were discussed, their variations, and complete set of results are on the large comprehensive sheet at the end of this appendix. The following legend will explain the symbols and formulas used in making the sheet.

A	Number of relevant retrieved documents	
B	Whole Number of retrieved documents	
C	Whole Number of relevant documents	
D	Number of non-relevant documents not retrieved	
E	Number of non-relevant documents retrieved	
	==========	
P	Precession	(A / B)
R	Recall	(A / C)
S	Specifity	D / (E + D)
F	Fallout	E / (E + D)

Efficiency: Overall efficiency of the search R + S –1

The following is a list of all the questions discussed in chapter 4 and found on the comprehensive sheet at the end of this appendix. The questions are listed in normal English and their corresponding F-Logic syntax.

1. **Give all skills of Philip**

   FORALL X,Y <- "Philip"[X ->> Y].

2. **Give all people who have an applied skill as a software tester**

   FORALL X <- X[AppliedSkill ->> "Software Tester"].

3. **Give all people who have any type of degree**

   FORALL X,Y <- X[Degree ->> Y:"Degree"].

4. **Return all people who have a skill with SQL Server and their work experience is between 7 and 9 years**

   FORALL X <- X[ToolSkill ->> "SQL Server"]
   AND X[WorkExperience ->> "7-9 Years"].

5. **Give all people with experience with a certain tool for 7-9 years**

   FORALL X <- X[WorkExperience@(ToolSkill) ->> "7-9 Years"].

6. **Get all people skilled with a tool which is a programming language IDE and their work experience is 4-6 years**

   FORALL X,Y <- X[ToolSkill ->> Y:"Programming environments"]
   AND X[WorkExperience ->> "4-6 Years"].

7. **Give all people skilled with any tool**

   FORALL X,Y,Z <- X[ToolSkill ->> Y:Z] AND Z::"Tools and Applications".

8. **To which category (concept) does a "UML certified" skill belong?**

   FORALL Y,Z <- "UML Certified":Y AND directsub_(Y,Z).

9. **Give all direct sub-concepts of "Tools and Applications".**

   FORALL Y <- directsub_(Y,"Tools and Applications").

10. **Give all Concepts under Root (All Concepts)**

    FORALL Y <- Y::"Root".

11. **Give all tree of concepts that XML belongs to**

    FORALL Y <- "XML":Y.

12. **Who participated in carrying out this task: Develop web-based System Navigation Structure?**

    FORALL X,Y <- X[Agent->>{Y};ParticipateIn->>{"Develop web-based System Navigation Structure"}].

13. **Who is skilled with Enterprise Java and has good testing skills (to test a java module)?**

    FORALL X <- X[SkillLevel@("DataBase Systems","Enterprise Java","Java")->>{"High"}] AND X[TechnologySkill->>{"Testing and Testing Techniques"}].

14. **What type of skill is needed to do this task: Build Web-based System EJBs?**

    FORALL X <- "Build Web-based System EJBs"[Skill->>X].

15. What is the test we did on a java module that produced a high quality artifact?

FORALL X,Y <- X[TestType ->> Y] AND X[Type ->> "Java Code"]
AND X[Quality->>"High"].

16. What was the previous task of building the web-based system JSPs?

FORALL X <- "Build Web-based System JSPs"[PreviousTask ->> X].

17. What was the task and its outcome where normal effort was put, but we still got high quality artifact?

FORALL X,Y <- X[Effort ->> "Normal Effort"] AND
X[outputArtifact->> Y[Quality->>"High"]].

18. Which Planning Activity had a very high effort and did result in a high quality plan?

FORALL X,Y <- X[Effort ->> "HighEffort"]
AND X[ResultsDocumentation->> Y[Quality->>"High"]].

19. What are the artifacts of the Reservation System release?

FORALL X,Y <- "Reservation System Release"[MadeOf ->> X]
AND X[Artifacts->>Y].

20. In which Iteration did we test the Classification Manager?

FORALL X <- "Test Classification Manager"[iteration->>X].

Next, is the questions comprehensive results sheet.

	Question	KeyWords	A1 (First Hit)	A2 (First 5 Hits)	A3 (First 10 Hits)	A4 (First 20 Hits)	B	C	D	E	Precision 1	Precision 2	Precision 3	Precision 4	Recall 1	Recall 2	Recall 3	Recall 4	Specifity	Fallout	Efficiency 1	Efficiency 2	Efficiency 3	Efficiency 4
1	Give me all skills of Philip	Skill / Philip	0	1	1	1	2	1	536	1	0.00	0.50	0.50	0.50	0.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00
		Expert / Philip	0	0	0	0	0	1	537	0	"0"	"0"	"0"	"0"	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
2	Give me all people who have an applied skill as a software tester	Software Tester / Agent / Skill	0	0	0	0	6	2	530	6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Testing / Experience	0	1	2	2	62	2	476	60	0.00	0.02	0.03	0.03	0.00	0.50	1.00	1.00	0.89	0.11	-0.11	0.39	0.89	0.89
3	Give me all people who have any type of degree	Agent / Degree	0	0	0	0	5	9	524	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Employee / Degree	0	0	0	0	8	9	521	8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.02	-0.02	-0.02	-0.02	-0.02
		Degree	0	0	0	0	45	9	484	45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.09	-0.09	-0.09	-0.09	-0.09
		Education	0	0	0	0	85	9	444	85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.84	0.16	-0.16	-0.16	-0.16	-0.16
4	Return all people who have a skill with SQL Server and their work experience is between 7 and 9 years	SQL Server / Experience / Employee / Year	0	0	0	0	0	1	537	0	"0"	"0"	"0"	"0"	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
		SQL Server / Skill / Year	1	1	1	1	1	1	537	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
5	Give all people with experience with a certain tool for 7-9 years	Tool / Experience / Agent / Year	0	0	0	0	7	2	529	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Skill / Year	0	0	0	1	28	2	510	26	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.50	0.95	0.05	-0.05	-0.05	-0.05	0.45
		Tool / Experience	0	0	0	0	80	2	456	80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.85	0.15	-0.15	-0.15	-0.15	-0.15
6	Get all people skilled with a tool which is a programming language IDE and their work experience is 4-6 years	IDE / Skill / Year	1	2	2	2	2	3	535	0	0.50	1.00	1.00	1.00	0.33	0.67	0.67	0.67	1.00	0.00	0.33	0.67	0.67	0.67
		Programming / Expert / Employee	0	0	0	0	3	3	532	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Proficient / IDE	1	1	1	1	1	3	535	0	1.00	1.00	1.00	1.00	0.33	0.33	0.33	0.33	1.00	0.00	0.33	0.33	0.33	0.33
7	Give me all people skilled with any tool	Agent / Expert / Tool	0	0	0	0	8	6	524	8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.02	-0.02	-0.02	-0.02	-0.02
		Employee / Skill / Tool	0	0	0	0	13	6	519	13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.02	-0.02	-0.02	-0.02	-0.02
		Skill	0	0	0	0	72	6	466	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.12	-0.12	-0.12	-0.12	-0.12
8	To which category (concept) does a "UML certified" skill belong?	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
9	Give me all direct sub-cocepts of "Tools and Applications".	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	Give me all Concepts under Root (All Concepts)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	Give me all tree of concepts that XML belongs to	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
12	Who participated in carrying out this task: Develop web-based System Navigation Structure	Task / Develop Web System / Navigation Structure	0	0	1	1	8	1	530	7	0.00	0.00	0.13	0.13	0.00	0.00	1.00	1.00	0.99	0.01	-0.01	-0.01	0.99	0.99
		Agent / Navigation / System / participate	1	1	1	1	1	1	537	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
13	Who is skilled with Enterprise Java and has good testing skills (to test a java module)	Enterprise Java / Testing / Agent	0	0	0	0	2	2	534	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
		Skill / Java / Test	0	1	1	1	10	1	528	9	0.00	0.10	0.10	0.10	0.00	1.00	1.00	1.00	0.98	0.02	-0.02	0.98	0.98	0.98
14	What type of skill is needed to do this task: Build Web-based System EJBs	Skill / Build / EJB / Task / Web	1	1	1	1	1	1	537	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
		Knowledge / Develop / EJB / Web	1	1	1	1	1	1	537	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
15	What is the test we did on a java module that produced a high quality artifact	Test / java / Artifact / High	0	1	1	1	2	1	536	1	0.00	0.50	0.50	0.50	0.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00
		Testing / Java / Module / High / Quality	0	0	0	0	2	1	535	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
		Good / Quality / Test / Java	0	1	1	1	7	1	531	6	0.00	0.14	0.14	0.14	0.00	1.00	1.00	1.00	0.99	0.01	-0.01	0.99	0.99	0.99
16	What was the previous task of building the web-based system JSPs	Task / Previous / Web-Based System JSPs	1	1	1	1	1	1	537	0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
		Build Web-Based System / Activity / previous	0	0	0	0	1	1	536	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
17	What was the task and its outcome where normal effort was put, but we still got high quality artifact.	Task / Normal Effort / High Quality Artifact	0	0	0	0	1	4	533	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
		Task / Medium Effort / High Quality	0	0	0	0	5	4	529	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Normal Effort / High Quality	0	1	1	1	10	4	525	9	0.00	0.10	0.10	0.10	0.00	0.25	0.25	0.25	0.98	0.02	-0.02	0.23	0.23	0.23
18	Which Planning Activity had a very high effort and did result in a high quality plan.	Planning / High Effort / High Quality	0	0	0	0	36	2	501	35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.07	-0.07	-0.07	-0.07	-0.07
		Plan / Very High Effort / Good Quality	0	0	0	0	6	2	530	6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
19	What are the artifacts of the Reservation System release	Reservation / System / Release / Artifacts	1	2	2	2	2	2	536	0	0.50	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00	0.00	0.50	1.00	1.00	1.00
		Reservation / Artifacts / System	0	2	2	2	8	2	530	6	0.00	0.25	0.25	0.25	0.00	1.00	1.00	1.00	0.99	0.01	-0.01	0.99	0.99	0.99
20	In which Iteration did we test the Classification Manager	Classification Manager / Iteration / Test	0	0	0	0	9	3	526	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.02	-0.02	-0.02	-0.02	-0.02
		Classification Manager / Task / Iteration / Test	0	0	0	0	5	3	530	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.01	-0.01	-0.01	-0.01	-0.01
		Average	0.2	0.5	0.5	0.5	14.0	2.9	521.8	13.3	0.19	0.26	0.26	0.26	0.16	0.33	0.37	0.38	0.98	0.02	0.13	0.30	0.34	0.35

All Documents:	538

*** The symbol "0" denotes a result of a division over zero*