USER MODEL DRIVEN ARCHITECTURE FOR INFORMATION RETRIEVAL IN CONSTRUCTION PROJECT MANAGEMENT

By

HAIYAN XIE

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2005

UMI Number: 3178054

UMI Microform 3178054

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This dissertation is dedicated to my family.

ACKNOWLEDGMENTS

First of all, I would like to thank Dr. R. Raymond Issa, who has been a great supervisor and mentor throughout my Ph.D. study at the University of Florida. His inspiring enthusiasm and energy have been contagious; and his advice, encouraging, and support have been extremely helpful. I also want to thank Dr. William O'Brien, whose knowledge and direction were crucial to my research. Dr. Robert Cox and Dr. Ian Flood added to the content and theoretical significance of this dissertation. Finally I thank Dr. Joachim Hammer, who offered me the valuable opportunity to broaden my knowledge and experience beyond this project. Each of them has positively impacted my current and future career.

The process of writing this dissertation was an interesting and exciting adventure for me. I had the opportunity to study in the field of construction information technology in depth, and to gain insights and experience in real industry work. The biggest privilege that I enjoyed was the work experience with many wonderful people who I sincerely thank for their advice, their support, and their encouragement. I also want to thank the staff of Taylor Contractors of Florida, Inc., a construction company from which I gained a lot of useful construction experience.

I also would like to thank my undergraduate assistants and my colleagues and co-workers for their willingness to help me with my daily teaching, research, and technical difficulties while completing this work: Dr. Mary Good, Mike Tramel, Sandra Bates, Dr.

Jim Blacklock, Jim Carr, Chris Ray, and all members of the Department of Construction Management at University of Arkansas at Little Rock.

I thank my friends and family for their support during this long and sometimes convoluted process. Specifically, I thank my parents, who are always prepared to support me and give me their deepest love. Additionally, I thank my parents-in-law. Without their help, I would not have been able to finish this dissertation.

My deepest gratitude goes to my husband, Wei Shi. He perpetually encouraged me to continue writing this dissertation. He is always cheering me up. Finally, I want to thank my son, Owen Shi, for his love.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

USER MODEL DRIVEN ARCHITECTURE FOR INFORMATION RETRIEVAL IN CONSTRUCTION PROJECT MANAGEMENT

By

Haiyan Xie

May 2005

Chair: R. Raymond Issa
Cochair: William O'Brien
Major Department: M.E. Rinker, Sr. School of Building Construction

Numerous software systems are designed to facilitate information retrieval and communication among people in the construction industry. It is arguable that user experience and preferences are given enough consideration in a lot of them. To design a system serving the information retrieval needs of the construction industry, designers must determine the information that construction personnel need to retrieve from various sources, and in what kinds of formats it has to be displayed. The next problem is how to formalize the user model for a given class of users (such as vice president, project manager, or estimator) to direct information retrieval.

Detailed case studies were used to specify and formalize user model(s) for construction personnel. The goal of this research is to improve the level of satisfactory results obtained from a conventional or advanced information retrieval (IR) system by modeling the user's information retrieval approach and leveraging the user model into structured knowledge. The users' needs will be analyzed and modeled by the use-case

modeling method, and then the relationships in the use case will be used to address the IR system with a more relevant request. The user model proposed in this research is based on the implementation of user research in software development. The Unified Modeling Language (UML) is implemented to build up a formal user model to generate a fast and smooth transition from the user needs to the software products. Furthermore, the Java object-oriented language is used to build up a configurable navigator pattern to retrieve information. Coding and system programming for information retrieval can be achieved based on these specifications. Five validation cases were designed to test the UMDA system from the aspects of user involvement and satisfaction, usability, sensitivity, search scope, and information retrieval performance. The validation process demonstrated that the UMDA system provides, in all the considered cases, the needed representations that are usable and that it supports effective data access and data collection of project management information. It also demonstrated that the interaction style of the UMDA system allows in many cases for efficient user interaction.

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

Because of the emergence of less expensive personal computers and the advent of software packages with friendly interfaces, computers are no longer restricted to highly specialized engineers or consultants. Computers are widely used in the construction industry as well. For example, plumbers use computers to design domestic water/wastewater pipes and calculate hydraulic formulas. Structural subcontractors use computer systems to design structural steel and to place purchase orders. Flooring subcontractors use computers to help customers select materials, colors, and patterns. Many superintendents have mobile phones with built-in cameras to send construction pictures to remote computers. Some of them type their daily field reports and send them through email to remote headquarters. According to a recent survey by the Construction Financial Management Association (CFMA), 89% of general contractor project managers responding to the survey use personal computers (PCs). Among project managers working for specialty contractors, use of PCs also showed a rise to 77%. Seventy nine percent of GC firms surveyed said their satellite offices and jobsites were connected to headquarters office networks (Cassidy 2004; CFMA 2004). All of these reflect that the usage of computers by contractors and engineers at jobsites has a widespread and profound influence on construction management.

The life cycle of a building can be described as a long-term decision-making process. From the decision to initiate planning, to the final decision of building

decommissioning, the diverse participants collaborate in a continuous effort to define and achieve the ever-changing set of multiple objectives of a building project. The flow of information required among all participants is enormous, diverse, and detailed, yet dynamic. Usability has always been important; but in reality, the threat of instant abandonment seldom hangs over every moment someone uses the product. This is because the user (or the company) paid for the software and was planning to use it for extended periods of time; and also because of the availability of other systems and the degree of their improvement. Things are changing now. With more and more options and the rapid production of updates in the software market, users are increasingly asking for systems to fit their needs.

But this occurs in an environment in which disconnected islands of information are documented and communicated in the traditional forms of voluminous paper-based specifications and graphical drawings with text annotations (Vorster et al. 1998). Much detailed and important information (such as project objectives and design rationale) is lost due to inadequate documentation and insufficient sharing of information between life-cycle participants (Mays and Novitski 1997). Participants who finish their tasks early and leave the project could also lead to information loss. For example, some subcontractors leave the project earlier than others; and if they did not keep good records of their scope of work, information may disappear. Thus, in most construction projects, project managers or superintendents will ask subcontractors to maintain as-built drawings on the jobsite, so that changes occurring on the jobsite can be recorded in a timely manner. Information loss or chaos may also occur when there are lots of changes during the design/construction process of a project. The vast amount of data and information

documentation generated during the construction process makes it difficult to manage the project effectively. The knowledge embedded in the design of complex engineered equipment and systems essential for project performance (as well as the knowledge needed for its successful integration, operation, and maintenance) is frequently lost or underutilized. For instance, in the design/build process, some commercial projects need to go through all the permit applications after drawings have been done. Normally to get all the permits and the Notice of Commencement, contractors must wait several months or even 1 or 2 years to make all the corrections and meet all the code requirements. If no well-kept change logs track all the revisions, contractors may lose money. The resulting inefficiencies impact time, cost, and quality of construction; as well as the lifecycle performance of facilities.

1.2 Research Problem

People often must go to great lengths to obtain useful information to make decisions (Figure 1-1). Even though some developers claim that their systems are

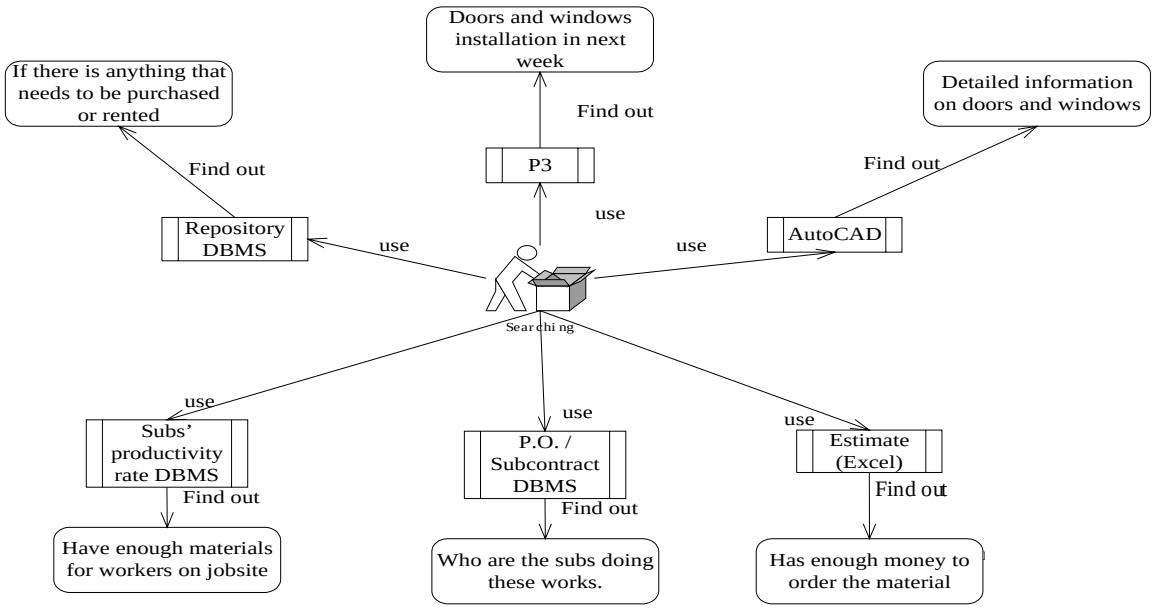


Figure 1-1 Efforts of a project manager in order to find out the answer to a resource question

user-friendly, some of the construction software products are not so easy to use. We agree that there should be some adequate user training to let users get used to new systems. But the clumsy searching process and excessive requirements of users' ability with database query languages will affect the usability of systems. Based on the analysis of the construction project management user model, the author aimed to build an efficient and practical "request-break-down and information-retrieval" model. By using this model, the author hopes to help project participants develop a more suitable tool for retrieving the intended information. The author also analyzed what the project-management software should provide to users through detailed case studies, thus showing the significance of understanding users and their needs. The research questions (RQ) include the following (RQ1 relates to conceptual design. RQ2 relates to implementing system architecture.):

- **RQ1**: What are the constituent components of user models for construction project management?
    - **RQ1a**: What user model can we build based on the detailed case studies and literature reviews?
    - **RQ1b**: How do we group users into different classes and what are the features of each of the user classes?
    - **RQ1c**: How does the system judge the user preferences based on the user interaction with system?
    - **RQ1d**: How do we present the search results to the users?
- **RQ2**: How does user modeling direct query processing and result fusion?
    - **RQ2a**: How do we utilize the existing functionalities from AutoCAD, Primavera Project Planner (P3)/Microsoft Project, Timberline/ Excel, and other existing software systems that are popular in the construction industry?
    - **RQ2b**: How do we utilize the design ideas from current user modeling projects illustrated in this research?

- **RQ2c**: Using the user model as a direction; can we build up the functionalities of the system?

1.3 Significance

This research discusses the current trends in user studies and user modeling. User Modeling is a technology that addresses the need to personalize information services in a computer system. A fundamental objective of human-computer interaction research is to provide users with experiences that fit their specific background knowledge and objectives. Different user research projects were compared and it was found that many times, system designers resort to a certain scale of investigation or survey and design the system based on the analysis of the results of the survey. These surveys can only help the system designers develop a ballpark idea of what users want. For example, the IBM User-Centered Design (UCD) approach began as a small pilot project and was developed into a company-wide fundamental element in the design process for all IBM product offerings. The UCD approach has a user-driven feature, which means that users are involved in system design and development. The user experience is designed first, and the product or system architecture is created to support this design (UCD 2004). The Object View Interaction Design (OVID) implemented the UCD approach. The Lumiere project at Microsoft Research was targeted at developing methods and an architecture for reasoning about the goals and needs of software users as they work with software. At the heart of Lumiere is Bayesian models that capture the uncertain relationships between the goals and needs of a user and observations about program state, sequences of actions over time, and words in a user's query (when such a query has been made) (Horvitz et al. 1998). For a lot of computer systems, including the above two example systems, user expectation,

user experience, human-computer interaction, and user process control are considered in the system design, when the user first logs into the system, or as an isolated module. Even though users are involved in the system design process, most designers still only consider user involvement at the beginning or the end of the design process.

User modeling aspect of implementation is the key component for customizing user experience for the user. However, the existing methodologies and systems have not yet been stressed or scaled to more complex levels of tasks. Thus the contribution of this research is to create a system that would allow people to combine different aspects or the elements of an overall user model in a scalable way. For this specific work, the author has taken the profile of a user class and combined it with the task profile of a task class. After determining the common elements of the user profile and the task profile, the author created a system to combine them. This is much more scalable than enumerating every possible use case. The intent of this research is to build a domain-specific system which will help people in the construction industry retrieve the information they need. The purpose of Information Retrieval (IR) is to find the information according to requestors' needs and user modeling servers to find out the right ways to response to these needs. Different from current commercial user modeling systems, the design and development process of the proposed system shows how the ideas are generated and how the process progresses based on surveys, use cases, formal use cases, to information retrieved from user profiles and task profiles and then how to configure the entire system. This research proposesan adaptable system, which is changed and controlled by user.

The research analyzed 3 case studies in detail and generated system requirements for construction management from the study of the construction project data and users

activities. In addition, some design ideas regarding the user interfaces which reflect the function and system requirements were presented. The primary operation of the proposed User Model Driven Architecture (UMDA) system is to search for the information needed by the users and to present the information in a way that the users will get to the requested information first without wasting time reading through all redundant data. The proposed UMDA system will act as a middleware in between the user interface and the underlying data management system. In order to find a suitable connection between the proposed UMDA and the data management system two options were compared: the Visitor Design Pattern and the ElementAmbassadors with Navigational Models. Both of these choices are appropriate for the proposed UMDA and they have different advantages as well as shortcomings. The ElementAmbassadors and Navigational Models were selected for implementation because it requires relatively simple programming. An ***EX***tensible ***T***ask ***E***lement ***T***ree (EXTET) for the user profiles and task profiles was built. The EXTET is a collection of lightweight utilities for generating system commands and processing. This data structure is a hierarchy of objects, each of which represents an element in the formal use case. It has the ability to extend to more complicated use cases or system rules.

By providing answers to the research questions, the proposed model will help the underlying computer system to process user requests into performable pieces of system commands. Standard information retrieval precision and recall techniques were used to test and validate the system. Another significant contribution of this model is that it provides a DBMS design schema for small- or mid-sized construction companies in management.

By doing research on user models for information retrieval, solutions on practical problems using a project DBMS can be tested. Although the proposed system will only test a couple of use cases, given sufficient case studies involving similar situations, an appreciation of the congruence between theory and reality can be achieved. In addition, by using synthetic cases to represent of the real world, the testing process will lead to modifications in the theories used in building up the system, which will let us form hypotheses which require further empirical, quantitative testing.

By studying information retrieval process in construction management, we can improve the productivity, quality, and efficiency of the construction industry. Results of this study can be evaluated on the basis of multi-user adaptability, user satisfaction, saving in time needed to make decision by contractors, and promoting communication among different participants in a project. Other values of this study are as follows:

- Allow project contractors to achieve improved time control.
- Help contractors and engineers to adapt to changes quickly.
- Lower the risk of information loss or data insufficiency.

After using the proposed system, both experienced and inexperienced participants can benefit from the model. Inexperienced participants benefit even more, especially in gaining valid data, because of the configurability of the system. Although some major barriers exist to implementing this technology (such as resistance to change, cost, and possible training requirements) the advance of new technologies in computer software and hardware is the motivation to explore more suitable techniques and to improve construction management (Mays and Novitski 1997).

1.4 Readers' Guide

User model literature, construction research literature, and information management literature compose the major part of the literature review of this thesis. One way of reading this study, especially for those without too much experience in user modeling or construction information technology, is to read through it entirely. For those who are in advanced stages of building user models but who are not yet thoroughly familiar with the construction industry, it is recommended that they skim through sections 2.1 and 2.2, and start reading from section 2.3 to Chapter 4, concentrating on Chapter 5 for case studies and Chapter 6 for system implementation and user interfaces. Readers interested primarily in implementing user models and information retrieval should review Chapter 5 and Chapter 6. Chapter 7 discusses the conceptual design of the proposed system and introduced some possible applications. Chapter 8 covers the design and programming details involved in implementing the proposed system. These two chapters do not require readers to have computer or information science background. Reading Chapter 6 first before reading Chapter 7 and 8 would help readers understand the design ideas of the programming code. For readers with interest to explore further into user model driven architecture design, Chapter 9 and 10 discuss the validation of the system and some suggestions for future research.

CHAPTER 2
LITERATURE REVIEW

This chapter discusses the current trends in user research and user modeling. Different studies regarding user research are discussed and compared in this chapter. With a focus on user research in the construction industry, some of the most widely used construction software is discussed from the aspect of their intended users, and considers how they achieve user satisfaction. Possible ways of implementing user research into construction software development are also discussed in this chapter.

In the past 10 years, the software industry has experienced the roller-coaster-like market cycle. Back in the good old 1990s of the NASDAQ bubble, everyone was dazzled by tech hype and the high stock price of the Information Technology (IT) industry. But since 2000, many IT companies have fallen from highly profitable and stock market-favorite status into bankruptcy. Surviving IT companies are trying to figure out what users want, and to design their products to compete with others in the market. Since 1994, the Standish Group has published annual reports on project failure and success factors, by analyzing tens of thousands of projects. In the failure column, over 30% of the pain factors were strongly related to requirements issues. In the 2000 National Software Quality Experiment (run by U.S. Department of Defense) identified that 41% of defects were related to lack of definition and traceability regarding requirements. In the list of critical success factors identified by the studies, high user involvement and clear basic requirements were two of the key ingredients (Adolph and Bramble 2003).No One-Size-

Fits-All solution exists in the software industry. Software developers need to study what users want.

2.1 User Research and User Model

User research covers a wide range of potential areas of study, from the question of users' choices of hamburgers in a fast food restaurant, to reactions to on-line surveys, to the in-depth analysis of the user needs that result in information seeking. Kuniavsky (2003) defined user research as the process of understanding the impact of design on an audience. Furthermore, he stated that surveys, focus groups, and other forms of user research conducted before the design phase could make the difference between a Web site (or any designed product) that is useful, usable, and successful; and one that is an unprofitable exercise in frustration for everyone involved. Users have different needs, different vocabularies, and different constraints; and are often operating on different schedules. Knowing users, finding out their perspectives and abilities, and scoping their exact needs are not simple tasks. For example, in the information-seeking process, users seek the information needed according to their individual habits. It involves a lot of work for a search engine designer to find the right system for a certain group of users.

2.1.1 User Definition

"User" is a term in many communication/information contexts. "User studies" need to distinguish among these contexts before research is planned. The definition process may also help the researcher draw related concepts and theories from these alternative contexts. According to Wilson (2000), user can be seen as communicator, drawing upon personal or organizational information resources in communicating with organizational colleagues or fellows in society. In seeking the information needed, user can be identified by separate tasks; and the seeking process involves not only inter-personal

communication, but also the use of formal information systems, defined widely, as any device, product or system intended for information representation, storage, conservation, retrieval, or re-packaging. For example, the on-line library database is an information system that provides all the above listed functions.

One important characteristic of users is that they and their needs may change over time. Everyone is unique and each person has his or her own view of the world. Personalities, experiences, and training process shape the individual natures. As a result, each individual sees things differently. The typical user of a system does not exist. There are many different kinds of users, and the requirements of an individual user usually change with experience and time (Mackay 1991). Simple classification schemes based on stereotypes (Rich 1989), such as novice, intermediate, or expert users, are inadequate for complex knowledge-based systems because these attributes become dependent on a specific context rather than applying to users globally. Clark (1972) found that the changing needs of users over time may require changes in the way service tasks are performed. The growing number of users to be served (e.g., as a result of urban expansion) may lead to new organizational structures (e.g., branch library services). Clearly, the changes must also carry with them the promise of reduced costs or improved performance, or both. Organizations, however, are generally resistant to change. There is a definite gap between intention to change (as a result of research findings) and the actual accomplishment. Resistance to change is, in fact, one of the principal causes of organizational failure (Wilson 2000).

2.1.2 Understanding Users

In social psychology literature (as shown in Figure 2-1, Maslow's Hierarchy of Needs (Kotler et al. 1998)) divides human (i.e. user) needs into several classes, which

have been depicted as forming a pyramid from bottom to top according to the importance of the needs to human life. The very bottom level must be satisfied for people to sustain their physical life. The need for achievement, for self-expression, and self-actualization are realized providing the physiological needs are satisfied. With more and more software systems being developed to reflect the complexity and various classifications of the real world, the development should satisfy the needs of users. For software development user needs will be expressed using the concepts inherent in the software development approaches. For example, in object-oriented design, people use the terms such as objects, messages, events, and inheritance. These concepts belong to the software world instead of the user world, but the design itself has to reflect the user's needs.

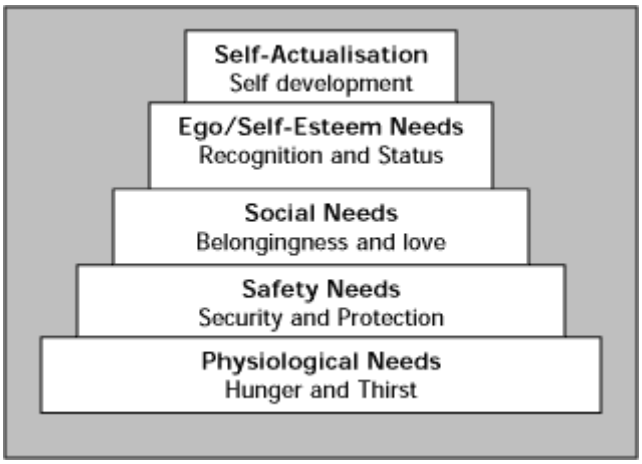


Figure 2-1 Maslow's Hierarchy of Needs Theory (Kotler et al. 1998)

Information needs of users are aroused by the lack of information at the time. The debate on the nature of information (Belkin 1978; Debons 1980; Farradane 1976) revealed the vague character of information science. The research objects of information science vary from the most basic (electrons, hardware, and the associated formulae, i.e. efficient storage in computer memories) to people behaving in particular ways to obtain needed information. The search for information may be rooted in the desire by some

people to pursue power. Possessing information helps them to establish dominance over others by reminding them that the person is better informed and, therefore, in some sense superior.

The particular pattern of needs and the resulting form of information-seeking behavior will be affected by a lot of factors, such as the organizational level at which a role is performed. The way a cashier collects information is different from that of the CEO of a supermarket. Choice of modes to use in searching will be guided not only by cognitive needs (e.g., when it is known that an appropriate data-base is on-line) but also by affective needs (e.g., the greater visual variety offered by TV) and by other environmental or economic factors (Wilson 2000).

In order to uncover these facts from everyday life, we need to uncover the needs that push the individual towards information-seeking behavior. After we understand the users, we will be able to design more effective information services, and to create useful theory of information-seeking behavior and information use.

2.1.3 User Research

User research as used herein refers to research on the generic social meaning to people who have, are or will use a certain product. Research on users has relationships with many research fields, such as sociology, psychology (McQuail 1975), and communication studies (Weinshall 1979). User studies in the past often focused on the interaction of people with systems (Burns and Hasty 1973) or inquiries into the use of systems in general (Bernal 1948) rather than the study of underlying user needs. From then on, researchers have been paying more and more attention to the study of users' needs. Kuniavsky (2003) listed several research methods of doing user studies in his book about observing the user experience. He recommended doing a usability test for the

product a company wants to develop. “The usability test will tell you whether your audience can use what you have made.” There are four major steps in the process of conducting a usability test, shown as follows:

- Define the audience and their goals
- Create tasks that address those goals
- Get the right people
- Watch them try to perform the tasks

Kuniavsky (2003) further stated that a product (or a portion of a product) is functional if it does something considered useful by the people who are supposed to be using it. According to (Kuniavsky 2003):

> Much user experience research is geared toward understanding how people could experience your site or estimating how they would want to. But it is difficult to understand how they are experiencing it. Satisfaction survey can get at what people feel is working and not working, but people’s predictions and preferences are not good predictors of their behavior. Contextual inquiry can reveal issues in the way people use your site, but only one person at a time. Usability testing can uncover many likely problems, but suffers from the problem of projecting people’s behavior in a laboratory environment versus real life. None of these actually tell you how people are currently using the site, what problems they’re really having with it in the wild… There are two sources of information you probably already have that reveal your users’ current experiences: customer support comments and Web server log files. (Kuniavsky 2003, p. 102-156)

In the above quoted paragraph, Kuniavsky (2003) analyzed the current user experience research methods and merged customer support comments and server log files with the methods. Other methods such as competitive research (understanding the competitors) and published information and consultants can also help to deepen the understanding of the users. In user research and profiling, the users' needs and expectations are defined as functional requirements to focus design and to meet the users’ needs. Functional requirements are the guide for software system design and

development. These requirements serve as a basis for the entire project team so they can work together with a common understanding of functionality, organization and goals.

2.1.4 User Modeling

The meaning of User Modeling in this research refers to the model built up to describe the user experience with a certain product, mainly software system. In the rest of this study, Formal User Model will be used to refer to the user model that itself is part of a software system or that can be converted into software systems without loosing its integrity and consistency. With the maturation of the software industry, it has started paying increasing attention to usability. Developers want to know if users can find what they are looking for with satisfying experiences. Usability of software has a strong impact on user satisfaction. More specifically, developers are asking themselves if their system can provide easier information access for users than their competitors' software.

User modeling techniques can be used to describe an individual human experience. As Kobsa (2001) pointed out, the research on user modeling is usually traced back to the works of Allen (1979), Cohen and Perrault (1979), and Rich (1979a; 1979b). Ten years after this seminal research, numerous application systems for user modeling were developed. These models collected different types of information about their users, and exhibited different kinds of adaptation to, their current users. Many of these early user-adaptive applications have been reviewed by Kobsa and Wahlster (1989), and McTear(1993). Kobsa has divided user-modeling research into three stages. The earliest stage occurred before the mid-1980s, and user modeling during that stage was performed by application systems. No clear distinction could be made between the system components that served user modeling purposes and those that performed other tasks. From the mid-1980s to mid-1990s, such a separation was increasingly made (Sleeman

1985), but no efforts have been reported on rendering the user modeling component reusable for the development of future user-adaptive systems. Past work in the area lacks a robustly defined and generally applicable system design that will enable end-users to access personalized information. From the mid-1990s till now, to address the issues of reusable components and robustly defined system design, current user modeling research has focused on user-adaptive systems. User modeling, software reuse, and organizational memories and organizational learning (e.g., creating socio-technical environments that help to transcend the limitation of the individual human mind) are some of the approaches that combine computer techniques with natural and/or social issues.

User Modeling is a technology that addresses the need to personalize information services in a computer system. A fundamental objective of human-computer interaction research is to provide users with experiences fitting their specific background knowledge and objectives. It is called customized software design. The challenge in an information-rich world is not only to make information available to people at any time, at any place, and in any form, but specifically to say the right thing at the right time in the right way. Nowadays, user-modeling research attempts to address these issues, and this is the reason that current user modeling research is focused on user-adaptive systems.

In summary, the definitions and explanations of the frequently used terms have been discussed in this section. The history of the definition of user research and the origination of doing research on users' needs make it possible to build up a user model based on detailed research. Given that there are many ways to research user models, such as psychological studies, this study will only discuss the user research issue from the computer science aspect, with the intended users being construction management staff.

2.2 User-Centered Software Development

For decades, professionals have struggled to understand what makes users decide to use a product or not. Researchers have investigated this question using such techniques as task analysis, usability testing, and semiotic system analysis. Still, the question remains whether the designers conceptualize a product with the user in mind according to cognitive task analyses and requirements gathered. But even for one class of users, such as construction management staff, there are several different groups (i.e. user groups with different experiences or backgrounds).

Users and designers rarely speak the same language. Consequently, a method must be developed to capture the dissimilarity between users and designers in conception. The goal of the User Centered Design (UCD) technique is to allow people to talk about what they think or have experienced, and enables them to speak about their interpretations of others' actions. It is known that product design benefits from an iterative process, where the production team proposes, reviews, and refines ideas. If the final users can talk or work with system designers to build up the system, then the final users can tell the designers exactly what they want and the designed system will fit the users' needs with a higher probability. But this is a Utopian scenario. Many times, system designers resort to a certain scale of investigation or survey and design the system based on the analysis result of the survey. These surveys can only help the system designers develop a ball-park idea of what users want.

2.2.1 User Assumption in Software Development

When people sit down and start to analyze the intended final users, they will find out that the final users can be divided into several groups according to different criteria. The experience level is one of the dividing lines to describe different users. But this

subjective criterion is hard to leverage in practice. How could the designer know a specific user is an expert user or an ordinary user? Should the designer judge from the time s/he has been working in the specific area, or from the number of choices s/he can find out when facing some problems, or the education level?

No matter how the designer wants to scope final users, they need to make some assumptions about the users. User modeling usually includes three possible kinds of facts:

- User observations (i.e., certain information observed about the user or facts present in all the possible models)
- Assumptions (i.e., default information inherited from the stereotypes)
- Derived data (i.e., information inferred from observations and assumptions through inference procedure directly managed by the application)

Lampson (1983) talked about user assumptions that each programmer needs to make in order to demonstrate the correctness of program. He concluded that " defining interfaces is the most important part of system design." The interface is the mediator between the user and the system. Usually interface design is also the most difficult part in system design," since the interface design must satisfy three conflicting requirements: an interface should be simple, it should be complete, and it should admit a sufficiently small and fast implementation."

According to Lampson (1983), it is easier to program and modify a system if its parts make fewer assumptions about each other. On the other hand, it may not be easier to design a good interface for the system. Making good assumptions can also improve performance. The additional assumptions often allow less work to be done, sometimes a lot less. For instance, if a set of size N is known to be sorted, a membership test takes time interval log N rather than N to complete. This technique is very important in the

design of algorithms and the tuning of small modules. In a large system the ability to improve each part separately is usually more important.

2.2.2 User Training

A successful training program is one which focuses on user needs to ensure that learning is achieved. Similar to documentation, training is often one of the last things that a product designer thinks about. Invariably, the assumption is that a product is "intuitive" and needs no training. Usually, it is only after new users provide feedback to the contrary, that the need for training becomes evident. Good training increases the understanding and subsequent comfort level with a product, which translates into product acceptance and use.

Almost every computer system provides user training or user education. Besides the User Guide, Manual, Instruction, or documentations, there are multiple other ways, such as tape, physical model, and multi-media. An empirical study was carried out by Wiedenbeck et al. (1995) comparing three kinds of hands-on practice in training users of a software package: exercises, guided-exploration, and a combination of exercises and guided- exploration. The assumed users of their system were moderate to very experienced computer users. Subjects who were trained with exercises or the combined approach did significantly better in terms of both time and errors than those trained using guided-exploration. There were no significant differences between the exercise and the combined approach groups. Thus, it appears that the better performance of these groups can be attributed to the exercise component of their practice. The result matches with everyday experience: through hands on use of computer software or even computer games, people can learn faster and better. During the user training process or software system testing process, designers could receive some feedback from the users. All this

feedback is valuable to the improvement or next stage of design of the software system. The following is an example of how user's response helps software development:

> Microsoft fought battles about old and new security holes with IE patch since long time ago. In February 2002, Microsoft issued a security patch that fixes six holes in its Internet Explorer Web browser. A nasty Internet worm that threatens to overwrite certain PC files continued its assault on users. The user response and software development process in the Patch design done by Microsoft shows that user responses help in the development and improvement of software system design. In addition, the new released cumulative patch includes a handful of previously released fixes for other holes found in the browser, which will in turn help users to strengthen their systems. One of those fixes is designed to block a worm known as "Klez.e," which has threatened some Internet Explorer users since it first circulated via e-mail. Klez.e affects Internet Explorer Versions 5.01 and 5.5, and strikes at a vulnerability in the Web browser. Once Symantec had received about 1,000 notifications of Klez.e from home and corporate users during 6 months, when the harmful variant surfaced. Symantec still regards the worm as a "moderate" threat (Berger 2004).

2.2.3 Differences between User-Centered and Market-Centered Software Development

The introduction of new technologies has led to a range of customized products with complex and embedded functionality. Increasingly products are being designed to handle a number of complex tasks. Most such products incorporate the functionalities with which user can only interact indirectly through an interface. Computer manuals reflect the inner workings of the software industry (i.e. the heated politics, the short product lifecycle with its eager push-to-market mentality, and the market-driven emphasis on features rather than benefits). In addition, users' poor previous experiences with documentation shape their expectations for their next purchase. Users do not know what their computers can do, and their manuals will not tell them everything. Ultimately there is a fundamental disconnect in the computer industry. Software developers and users do not understand each other. As a consequence, users often struggle with the

complexity of the interface instead of interacting with the content. But the content is the part that the user is interested in.

The goal of designing user-centered products is to establish ways to ease the communication between user and product. The design concepts and prototypes should demonstrate the overall user experience and allow for quality interaction. User-product communication design concepts should be as far reaching as possible towards exploring new paradigms of user-product interaction.

Different from User Centered Development, market centered software development uses in-stock technologies, and trims the user needs to fit the existing technology. As for the current software system developments, many technology companies simply follow the technical opportunities they see; hoping that the technology they create will find a market need. This strategy is market centered development and high-stakes gambling. Many innovative products do come out of this strategy, which can result in huge profits. But a lot more "great new technologies," in fact, start from an understanding of users and then find the technology to serve them, rather than the other way around.

Some technology companies respond to their user needs by growing existing products with the feature requested by customers. This process, which may or may not end up in significant improvements, refines the product the company started with. In the long term, if an innovative new system can serve in a better way, it will replace the purely renovated one. As a passive improvement process, developers wait for users to tell them what to change. Some users are reluctant to do so. If they find out the system cannot fit their needs and there are other systems on market, which seems more suitable, they will switch to new ones. Proactively figure out what users want is more important than

wait for them to tell us. Similarly, many companies have strong marketing groups doing quantitative surveys on their customer base. Nevertheless, to achieve innovation means to deliver products and services that address needs that no one else has seen.

To achieve user-centered development, innovation has to be targeted directly, looking closely at what people need, and giving planners the responsibility to invent new products that address those needs. This can produce not just a single product but a portfolio of products and services that address a range of needs for a range of users and customers. People may worry about the feasibility of user-centered development. Many companies, especially technology-oriented companies, start new products by looking at the technology available. But stepping away from thinking about those real constraints for a while at the beginning of the design phase, can lead to better products that sometimes even turn out easier to build.

2.3 Current User Research in the Construction Industry

Getting to know information users in the construction industry is not a simple task. Some unique and comprehensive studies have addressed user information preferences and habits, behavior patterns, decision making, and information content spending levels in this particular industry. This and subsequent sections will introduce the current user research in the construction industry. Before analyzing detailed user research projects , let us first discuss the history of user research in the construction industry and why its importance.

The products of the construction industry are for people to use, such as houses, roads, bridges, and hospitals. Customer satisfaction is always the primary goal of this industry that provides services to people. For instance, the Building Research Council in the School of Architecture, University of Illinois at Urbana-Champaign did some

research with the purpose to find out what the customers' preferences would be. Some construction industry researchers designed Space Use Laboratory, which was a research house, designed to permit quick changes in floor plans to test the reactions of users. Many "guinea pig" families lived in several research house versions to test user satisfactions with the various floor plans (BRC 2004). Even though the user needs of construction customers have been and are studied by a lot of researchers, the user research on the construction workers (including jobsite workers, foremen, superintendents, project managers, etc.) is mainly focused on the productivity study. Construction research institutes have the detailed productivity rates calculated based on survey results. These rates are of the average workers and vary through different crews.

With the emergence of the Information Engineering and the Information Industry, the information needs of the construction workers, especially organizers and managers, are causing the interest of researchers. In order to provide people working in the construction industry with decision support covering every aspect of the information content in the industry, giving precise intelligence and insight with confidence, researchers are helping them make the right strategic and tactical decisions regarding Information Content for their organizations.

As described in the mission statement of one information provider (Outsell 2004), they do the following research.

> We interact with the Information Content market daily and track the movements of the entire industry, from commercial vendors, through purchasing and deploying functions, to end-users. We influence Information Content vendors to create better solutions for the marketplace, and influence Information Content buyers and users to apply best practices for content purchasing, deployment and use. (Outsell 2004)

Some IT companies have become aware of the great business opportunities in the construction industry. The information needs of construction managers, which used to be

ignored before, are being more frequently researched and implemented. The author here proposes to group user research in the construction industry into three categories:

- Research on achieving the satisfaction of the construction customers. This category includes the effort in making the construction projects delivered on time, within budget, and with good quality.
- Research on improving the performance of the construction workers. This category includes the study of construction safety, new building techniques, scheduling research, estimating methods improvement, information needs, and workflow study.
- Research on continuing education.

The proposed study belongs to the second category. The main purpose of our study is to find out what information construction project management staff need, how construction managers retrieve the information they want, and how to improve the information retrieval process. This study will address the implementation of the re-configuration of the user model in the information system to fit the different user needs.

2.4 User Research Projects

From web site to digital library, a lot of designers are incorporating user research into their projects. User research presents fascinating socio-technical challenges for understanding user needs. Those provide information services to construction projects are naturally concerned that their investments pay off in terms of attracting users and making information services more effective and efficient. The design and evaluation of the systems, however, are their ability to integrate a range of functions that were previously designed, the heterogeneity of their user population, and the rapid versioning of digital objects. Some of the developing trends put more challenges into the design and evaluation of systems, such as the physically distributed nature of usage and the ability to fragment and rearrange previously integrated documents and images.

User research teams have already produced some valuable findings and described some provocative theoretical and methodological challenges. Appropriate user-centered research objectives, measures, and methods are slowly emerging. For the purpose of this study, only user research in Information Technology related fields are enumerated as follows:

- User research projects for commercial software systems
- User research associated with IT Market studies
- User research in theoretical studies
- Other types. To be specific, some construction software systems have the construction workers as their users. The designers of these systems may spend some time and effort in doing the user research. In order to assist users to setup and use the system more effective, the designers created “ReadMe”, “Help”, and some other files or functions. There may not be specific user research projects in these software systems, but from the help files and functions, the user research efforts can be traced.

2.4.1 User Research Projects for Commercial Software Systems

Before investing money and time into physical system implementation, many IT companies will have designers or other consulting firms perform user survey and user research. There exists a productive dialog among academic and industry-based researchers and usability engineering practitioners. Academic research has provided insights into methods for understanding and modeling user behavior, and industry has provided a wide range of exciting technologies for consideration by researchers in Human Computer Interaction. A personalization Server (ATG 2000) allows for the definition of rules that assign individual users to one or more user groups based on their demographic data (e.g., gender and age), information about the user's system usage, and information about the user's software, hardware and network environments. Rules can also be defined for inferring individual assumptions about the user from his or her

navigation behavior, and for personalizing the content of web pages. The operation of the Personalization Server thus follows very much the "stereotype approach" from classical user modeling research (Rich 1979b, 1989).

In order to compare the differences between the following existing projects or systems with the proposed system, a case study will be used to illustrate a situation that will happen in a lot of construction projects and the possible results derived by using these different systems. The use case that will be illustrated here is the information searching activity done by a Project Manager in finding out the delivery date of a roll-up coil door and whether that date is appropriate. The project is a restaurant. This coil door is sitting on the surface of the counter top in the reception area of the restaurant. The task required to be completed before the coil door installation is to have counter top ready, which is included in the cabinetry scope of work. The dry wall studs are in the process of being set up. After the completion of metal studs framing, hanging the drywall, and finishing the drywall, the cabinetry subcontractor can start their scope of work. It is not necessary to have all the drywall finished, however the area to receive cabinets and counter tops has to be completed. For each of the following research projects, the content of the research project, the assumptions made about the prospective users, and its possible implementation in the above coil door scenario will be discussed.

2.4.1.1 General user modeling system

The General User Modeling System (GUMS) (Finin 1989; Finin and Drager 1986) software allows programmers of user-adaptive applications to define simple stereotype hierarchies, and for each stereotype, Prolog facts describing stereotype members and rules prescribing the system's reasoning about them. At runtime, GUMS accepts and stores new facts about the user which are provided by the application system, verifies the

consistency of a new fact with currently held assumptions, informs the application about recognized inconsistencies, and answers queries of the application concerning the currently held assumptions about the user. Albeit GUMS was never used together with an application system, it set the framework for the basic functionality of future 'general' (i.e., application-independent) user modeling systems, namely the provision of selected user modeling services at runtime that can be configured during development time. When filled by the developer with application-specific user modeling knowledge, these systems would serve as separate user modeling components in their respective applications.

Let us first use the General User Modeling System (GUMS) as an example to show if the GUMS research can help the above use case. As stated before, GUMS was seldom used together with an application system. It listed the basic functionality of general user modeling system. For the coil door in the case study, once the GUMS is implemented, the system will accept and store new user information in a file at runtime. The system will then make assumptions about the user and answer user queries accordingly. Because different users can have different views of the same object, for example, when an company accountant uses this system, s/he will not need to know the installation details of the coil door. S/He only needs to know the cost of the door. In the proposed system, the intended users will be categorized into several groups. For each group, user preferences will be set based on their responsibilities. No other assumptions will be made. In addition, the answers to user request will be ordered according to their relevance to the user request and the user's class.

2.4.1.2 User-centered design in IBM Company

IBM is one of the leaders in the development of human computer interaction and human factors engineering from its inception. The IBM User-Centered Design (UCD)

approach began as a small pilot project and developed into a company-wide fundamental element in the design process for all IBM product offerings. UCD has become an industry standard for building usability into the design of IT products in IBM. As Vredenburg, et al. (2003) stated "UCD is more than a paradigm for designing usable products and systems. It is a blueprint for the practical implementation of product and system design, encompassing all aspects of the design process, from the gathering of user requirements during the early conception of a product through all aspects of the product cycle, including testing, product shipment, and beyond." IBM has expanded the focus of UCD, which is user-driven, to create User Engineering (UE), which is business value-driven. The purpose of UE is to transform the design process in order to meet business and market requirements as well as user requirements. UE is critical in the implementation of two aspects of IBM's e-business on demand strategy, for example, autonomic computing and integration.

A fundamental principle of UCD and UE involves the coordination, communication, and collaboration of disparate elements of the product design team. UCD and UE guide the development of a design team (including system architects, visual designers, marketing specialists, user research specialists, and others), which works closely together towards a single goal and product conception. The design process embodied by UCD and UE is iterative. In addition, UCD and UE are constantly changing to meet the changing needs and conditions of the information technology market place.

The IBM UCD Project Team comes from multiple disciplines and includes representatives of the fields of visual or industrial design, human factors, information development, marketing, project management, service and support, technology

architecture, and user interface design. The six stages of the UCD design process, and some possible methods for gathering user input during each stage (UCD 2004) are as follows:

> Stage 1. **Market Definition:** Define the target audience, identify competitors, and determine the core user needs and wishes that must be fulfilled for the product to succeed. The typical methods for this stage are to ask members of proposed target audiences to rate their levels of interest in a new product or product enhancement; to ask target users to list and prioritize their needs and to identify current solutions they use and prefer.
>
> Stage 2. **Task Analysis**: Identify and understand the users' goals and tasks, the strategies they use to perform the tasks, the tools they currently use, any problems they experience, and the changes they would like to see in their tasks and tools. The typical methods are to ask users to list and prioritize tasks, to observe users accomplishing tasks, and so on.
>
> Stage 3. **Competitive Evaluation**: Determine the design strengths and weaknesses of the competition. The typical methods include asking users to complete the same tasks using different products and assess their overall satisfaction with each one and asking them to list the strengths and weaknesses of products in order of importance.
>
> Stage 4. **Design and Walk-through**: Using the results from task and competitive analyses, create alternative proposed solutions, solicit feedback through design walk-through sessions with users, and choose a solution based on user input. Typical methods include asking users to evaluate "lo-fi" prototypes such as simple sketches.
>
> Stage 5. **Evaluation and Validation**: Periodically solicit user feedback on the evolving design, and iterate the design based on analysis of users' experiences with it. Typical methods include observing users accomplishing important tasks with a working prototype.
>
> Stage 6. **Benchmark Assessment:** Run a head-to-head benchmark assessment against the competition to verify that the product has met its primary objectives. If a third-party company conducts the benchmark study, positive results can become important selling points in product promotions. The Typical methods include to ask users to complete the same tasks using different products and assess their overall satisfaction with each one and to ask them to list the strengths and weaknesses of products in order of importance.

Those six stages of the UCD design process have been used in a lot of IBM system development projects. A variety of characteristics illustrating the progression from a

traditional approach to IBM's UCD approach and then to their UE approach are shown in Table 2-1 (Vredenburg et al. 2003). The most obvious difference between traditional approaches and UCD is the involvement of users. The user driven feature of UCD means that users are involved in all stages of system design and development. The user experience is designed first, and the product or system architecture is created to support this design. The UE process starts by collecting detailed market requirements, business requirements, and user requirements, creating a business model that rigorously integrates all of these requirements and focuses on the design aspects that affect the "bottom line."

Table 2-1 Progression from traditional to the UCD and then to the UE approach

	Traditional Approach	UCD Approach	UE Approach
Driven Factor	Technology driven	User driven	Business value driven
Focus	Component focus	Solutions focus	Enterprise focus
Team work	Limited multidisciplinary cooperation	Multidisciplinary teamwork	Multidisciplinary role-based work allocation
Specialization	No specialization in user experience	Specialization in user experience	Specialization in all disciplines
Competition	Some competitive focus	Focus on competition	Detailed focus on competition
Development	Development before user validation	Development of user-validated designs only	Engineering of user-validated models
View	Product defect view of quality	User view of quality	System view of quality
User Measurement	Limited focus on user measurement	Prime focus on user measurement	User measurement driven
Customer	Focus on current customers	Focus on current users and customers	Focus on all stakeholders

The IBM design and development community uses multiple design methods to implement UCD and UE. Approaches such as user object modeling, initially used in the development of the Common User Access (CUA) and OS/2 Workplace Shell interfaces 9 were further developed into the Object View Interaction Design (OVID). The OVID method draws from software engineering and related tools. It utilizes UML diagrams to

precisely specify the key aspects of a user experience, such as the user objects, their properties and relationships, and views of those objects that enable performance of user tasks. OVID models the user experience and not limited to software offerings. In fact it has been used for hardware, software, and Web site design projects. User models are becoming widely recognized as crucial to satisfy users' expectations. An overview of OVID is shown in Figure 2-2 (Berry et al. 2003). OVID is a set of methods for discovery, abstract design, and realization design.

One important feature of UCD is user driven; while for UE, it extends the contents of UCD and its design of offerings is based on a combination of factors that contribute to business value rather than solely on user input. It increases the likelihood of innovative "quantum leap" improvements as opposed to small incremental enhancements. One of the contributions of IBM's UCD and UE approaches is that they showed to other IT companies the importance of user involvement in the design of systems. They also illustrate the feasibility of incorporating user suggestions and desires into real IT products.

IBM's Collaborative User Experience (CUE) Research group, formerly Lotus Research, is a subdivision of the IBM Watson Research Center and is doing research with emphasis on the interaction between people and computer systems in support of collaboration. Their research themes are Activity Management, Large-Scale Collaborations, Information Visualizations, and Collaborative Development Environments. CUE is also involved in ODIS (On Demand Innovation Services) and developing consulting assets (software and methodologies) that support collaboration in customer engagements. For Activity Management, CUE intends to look at a variety of

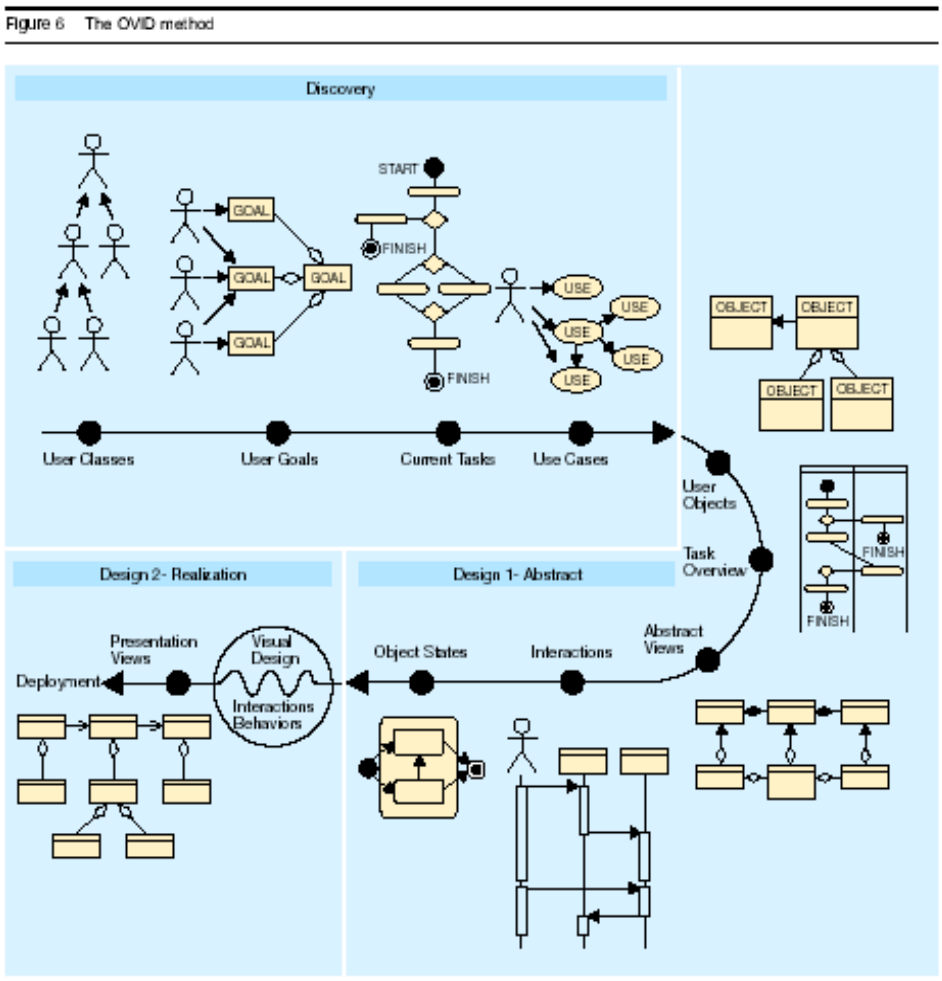


Figure 2-2 Example of IBM Object View Interaction Design (OVID) Method (Berry et al. 2003)

mechanisms to help people with interruptions and monitoring the progress of shared documents and tasks. Many of the innovations focus on supporting context switching and collecting the right information. Example research projects are Instant Collaboration and Reinventing Email. For Large Scale Collaboration and Visualization, CUE focuses on the social and technological implications of on-line communities and large-scale collaboration. Through field study and prototype applications, investigators explore how to create and support distributed teams and on-line communities and how to use visualizations to make large amounts of information accessible to the community.

Example research projects are Collaborative Environments and the Business Value of On-line Communities. The Jazz project is at the core of the research scope for Collaborative Development Environments. This is an instance of "contextual collaboration" in which collaborative components are deployed in a new context rather than as a monolithic application. For Collaboration in Consulting Engagements, CUE is studying the On Demand Innovation Services (ODIS) (IBM 2004).

Listed below are two CUE projects. One is research on why people tend to use some outmoded "material" tools such as white boards in critical projects; the other is about collaborative writing in professional settings. In the research "Big Boards: Using large displays in project management," Whittaker and Schwartz (2004) noticed that despite a wealth of electronic coordination tools, many groups choose apparently outmoded "material" tools in critical projects. To understand the limitations of electronic tools, they studied two groups, contrasting the effectiveness of both kinds of tools. They showed that the size, public location, and the physical qualities of the material tool engender certain crucial group processes that current on-line technologies fail to support. For example, a large wallboard located in a public area promoted group interaction around the board; it enabled collaborative problem solving, and informed individuals about the local and global progress of the project. Furthermore, the public nature of the wallboard encouraged greater commitment to the project and resulted in the participants updating the rest of the team on its status more often. On the other hand, material tools lack dependency tracking (IBM 2004).

In another project "PeopleFlow: Shared documents and ad hoc workflow," Cohen et al. (2004) led a series of studies on group authoring. They interviewed and observed

the attorneys and their support staff, followed selected cases in depth, and periodically attended group meetings in Lotus' Legal department. Crafting a legal document involves interactions between lawyers, paralegals, and administrative staff with frequent discussion about language, content, shaping the argument, and where to find the critical information. It became apparent that as the lawyers work that they depend a great deal on tacit knowledge about who needs to be involved in writing the document and what it must convey. They observed two related phenomena, which emphasized the need to support ad hoc, non-sequential workflows among these people. The group of people changed over time and was different for each writing project. The lawyers saved the cover letters that accompanied each draft of the document that went out to clients. The lawyer referred back to these cover letters in the course of their ongoing work to track what each person had already seen. This information is important "meta-data" that can also facilitate workflow when used as reminders of who knows what at a given time. Cohen et al. (2004) designed a "PeopleFlow" toolbar tracking and displaying the people involved in producing a document, storing the history of the document, and providing users with simple ways to connect with people who are available to discuss the document. The use of a toolbar embeds this information in the document itself and centers discussion on the relevant sections of the document (IBM 2004).

In strict terms, OVID is not a physical system. User-Centered Design (UCD) and User Engineering (UE) are the industry standards for building usability into the design of IT products. The OVID method precisely specifies key aspects of a user experience and carries out the standards into workable steps. One similarity between the OVID method and the proposed system design is they all model the user experience. The OVID method

is also useful in helping us design the relationship between User Model and the system coding. This study will use user classes in the proposed system, but unlike OVID, it will not summarize a single goal from all the different user goals. Current tasks and use cases will be analyzed in Chapters 5 and 6. In Chapter 6, abstract views and the real implementation of the views will be designed. The reason for the difference between our proposed system design and the OVID method is that the OVID method needs to consider a system's business value for the developing company.

2.4.1.3 Lumiere project at Microsoft Research: Bayesian Reasoning for automated assistance

Recognizing the importance of the user's ability to access, assimilate, and act upon available information, Microsoft seeks to base their work on theoretically sound approaches and the systematic evaluation of techniques and methods. The Adaptive Systems and Interaction group (ASI) of Microsoft pursues research on automated reasoning, adaptation, and human-computer interaction. Interests of the group include principles and applications of decision-making and learning, computation in the face of complexity, techniques for information management and search, and the development and evaluation of innovative designs for visualization and interaction. ASI is at the center of user modeling at Microsoft Research, focused on inferring the goals and needs of users from multiple sources of information about activity and interests.

The Lumiere project at Microsoft Research was initiated in 1993 with the goal of developing methods and an architecture for reasoning about the goals and needs of software users as they work with software. At the heart of Lumiere are Bayesian models that capture the uncertain relationships between the goals and needs of a user and

observations about program state, sequences of actions over time, and words in a user's query (when such a query has been made) (Horvitz et al. 1998).

Bayesian formula has been used extensively in statistics and probability. It is a useful equation from probability theory that expresses the conditional probability of an event A occurring, given that the event *B* has occurred (written P(A|B)), in terms of unconditional probabilities and the probability the event *B* has occurred, given that *A* has occurred. In other words, the Bayes formula inverts which of the events is the conditioning event. The formula is

$$P(A \mid B) = \frac{P(A,B)}{P(B)} = \frac{P(A) * P(B \mid A)}{P(B)} \tag{2-1}$$

In above formula, *P(B*) in the denominator is further expanded by using the so-called "Law of Total Probability" written as follows:

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i) P(A_i) \tag{2-2}$$

Then the Bayes Theorem can be expressed as:

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B \mid A) * P(A) + P(B \mid A^{'}) * P(A^{'})} \tag{2-3}$$

The following lists the meanings of the symbols used in the above formulas:

P(A) = probability that event A occurs

P(B) = probability that event B occurs

P(A') = probability that event A does not occur

P(A | B) = probability that event A occurs given that event B has occurred already.

The usual notation for "event A occurs given that event B has occurred" is "A | B" (A given B). The symbol | is a vertical line and does not imply division.

P(B | A) = probability that event B occurs given that event A has occurred already

P(B | A') = probability that event B occurs given that event A has not occurred already

A_i shall be mutually exclusive and exhausting all possibilities and including the event A as one of the A_i.

The Bayesian approach provides a formal process by which a-priori expert-judgment can be combined with sampling information (data) to produce a robust a-posteriori model. In software process engineering, the Bayesian formula can be used to calculate the chances if something will or will not happen. For example, a project manager who is concerned about his/her chances of experiencing a material delivery delay. Historically, it has been observed that the material vendor will delay the material delivery 20% of the time. In addition, there are a lot of conditions that might affect the material delivery schedule. Assume that there is a 90% chance that material ordered one day before the time it is needed will get delayed, whereas there is a 40% chance that material ordered two or more days before it is needed gets delayed. With knowledge of when the material was ordered, is there a way to update the prediction or belief in the likelihood of a delay? Using Bayesian analysis, an updated prediction of having a delay can be constructed as follows:

Prior(ordered late) = probability that the event “order material later” occurs. In this case, it is 20% chance.

L(ordered late | delay) = probability that the event “material delivery delay” occurs given that the event “order material late” has occurred already. In this case, 90% chance that material ordered one day before the time it is needed will get delayed.

L(ordered late | no delay) = probability that the event “no delay” occurs given that the event “order material late” has occurred already. In this case, it is 40% chance.

Prior(no delay) = probability that the event “no delay” occurs. In this case, it is (1 - 20%), which is 80% of chance.

P(delay | ordered late) = L(ordered late | delay)Prior(ordered late) / [L(ordered late | delay)Prior(delay) + L(ordered late | no delay)Prior(no delay)]

or

P(delay | ordered late) = (90% * 20%) / [(90% * 20%) + (40% * 80%)]

P(delay | ordered late) = 36%

The user is able to use information about material order time to update the prediction from 20% to 36%. In this case, the news is not necessarily what the user wanted to hear but it does represent the best prediction that can be made with the information that is accessible. To predict success in scheduling, one has to consider factors such as dependencies, resource availabilities, and technical and programmatic risks.

Bayesian analysis is ideal for updating predictions on software quality, cost, schedule, and process capability. As a matter of fact, the "learning" mechanism of the Bayesian formula would be excellent to adapt to changing environments, technology, and process maturity (accurate predictions of the future). Within Bayesian analysis, there is a method whereby the Bayesian formula can be modified to update one's prediction based on successive items of new information or evidence. By identifying the historical probabilities of the items occurring and the likelihood functions associated with the items. If there is a lack of objective data for all of these, one can insert professional judgment as a subjective likelihood for a forecast item or factor likelihood. Consequently, one can offset or supplement gaps in historical data with expert opinion.

The first Lumiere prototype was completed in late 1993 and became a demonstration system for communicating with program managers and developers in Microsoft product development groups. Usability studies have played a significant role in

the Lumiere project. For example, over 25,000 hours of usability studies were invested in Office '97 (Horvitz et al. 1998).

The Lumiere prototypes have explored the combination of a Bayesian perspective on integrating information from user background, user actions, and program state, along with a Bayesian analysis of the words in a user's query. A Bayesian methodology for considering the likelihoods of alternative concepts given a query was developed at Microsoft Research, in collaboration with the Office product group in 1993. This Bayesian information-retrieval component of Lumiere was shipped in all of the Office '95 products as the Office Answer Wizard.

Although most of Lumiere Team’s effort was focused on building an intelligent query service for interpreting natural queries, including the leveraging of dynamically sensed context about user activity and user competencies. The intent of doing this is to decide when to step forward to assist a user in a careful and conservative manner, with the express approval of users. On putting the user in control, a speculative assistance display was designed that included a featured "volume control" for interruptions; the volume control would always be available whenever such advice would appear. Such a volume control allowed users to move a slider in a "bother me less" or "bother me more" direction, changing the threshold at which active assistance would be provided. Thus, users could wrap the help system around their own preferences for the value of assistance. For example, the Lumiere Team examined the use of small windows that would not steal system focus, and that would time out with a gentle apology if the user's did not hover over or interact with the recommendations. The "background assistance tracking" feature would simply watch in the background as a user worked. An analysis,

including such compilations as a custom-tailored set of readings, would be made available for review or printing when the user requested such an overall critiquing, or context-sensitive assistance manual.

During the course of the Lumiere research, some alternative user-interface metaphors for acquiring information from users and for sharing the results of Bayesian inference with users were also considered. Beyond experimenting with embedded actions and traditional windowing and dialog boxes, the character-based interfaces was the way chosen to provide assistance services. The first phase of porting Lumiere to the real world occurred with the completion of the Office '97 product suite. The Office team has employed a relatively simple rule-based system on top of the Bayesian query analysis system to bring the agent to the foreground with a variety of tips. One example of several "assistants" from the Lumiere research result (available in Office '97 and the newer versions) is shown in Figure 2-3. The Office assistants provide a focus for event and query based interaction centering on assistance with the use of the software. This character is named "Genius."

Using the coil door in the restaurant project example; by implementing the design idea underlying Lumiere research project, what would happen? The design idea for Lumiere is a Bayesian formula, which can predict based on the known history. As shown in the material-ordering example, the Bayesian formula could be used to predict the possibility of delivery delay of this coil door. Or it can help to guess what will be the next question the system user want to ask and make the system be more intelligent, but it is very limited in retrieving the requested information from a database.

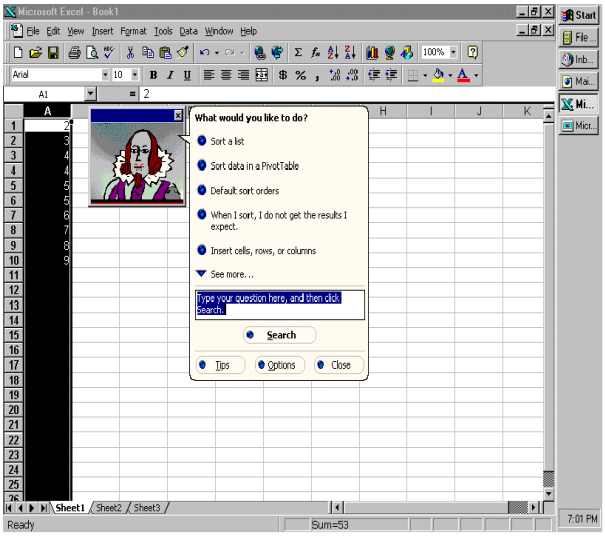


Figure 2-3 Example of Microsoft "assistants"

2.4.2 User Research Associated with IT Market Studies

Outsell (2004) did the research on I-AIM(sm) (Information About Information Markets), which is a study of the information habits and preferences of 10 user groups segments in 20 industries. A detailed break down of the responses of all 10 user-groups within the industry was provided. The intent of this study was to let commercial content vendors and internal content deploying functions use this data to understand market opportunities, size user markets, and assess groups of users or specific industries for product development efforts. Outsell claimed that based on interviews with workers in the construction industry, this report provides essential data about each major user group's spending, information use patterns, format preferences, and attitudes toward the Internet. This report is based on interviews with 308 users in the construction industry, broken down by user type. For each question, the report shows the responses of the users in each of the 10 user groups. Outsell is one of the software companies that provides services to the construction industry. The user survey and report show that user research is very closely connected with the target market and industry.

2.4.3 User Research in Theoretical Studies

There are lots of books talking about how to do user research and user modeling. Fischer (2000) talked about the difficulties in User Modeling. He mentioned the following three categories of user modeling research with high-functionality and usability.

- Exploring different domains such as: natural language dialog, human computer interaction, intelligent assistants, information retrieval, and high-functionality applications;

- Identifying important distinctions such as: adaptive versus adaptable components, explicit versus implicit modeling techniques, user models versus task models, canonical versus individual models, and long-term versus short-term models;

- Creating a number of challenging research problems, such as how to:

- Integrate different modeling techniques;
- Capture the larger (often unarticulated) context and what users are doing (especially beyond the direct interaction with the computer system);
- Identify user goals from low-level interactions;
- Reduce information overload by making information relevant to the task at hand;
- Support differential descriptions by relating new information to known information and concepts; and
- Reach a better balance for task distributions between systems and users.

The organizations that are devoted to user research include the Association for Computing Machinery Special Interest Group on Computer-Human Interaction (ACM SIGCHI), the International Federation for Information Processing Technical Committee (IFIP TC13), academic departments (i.e. computer science departments), and lots of other groups. Cooper (2003) talked about the user interface design essentials in “About Face 2.0: The Essentials of Interaction Design”. User interface design belongs to the first category: human computer interaction, as grouped by Fischer (2000). Cooper (2003) indicated that “even as technology frees us to perform great feats of invention, it

simultaneously ties us to ways of thinking that are contrary to the natural expression of human behavior." The authors of the book propose a detailed definition about design: "Understanding the user's wants, needs, motivations, and contexts; Understanding business, technical, and domain requirements and constraints; Translating this knowledge into plans for artifacts (or artifacts themselves) whose form, contents, and behavior is useful, usable, and desirable, as well as economically viable and technically feasible" (Cooper 2003). Cooper and his colleagues provided software designers many useful instructions and tips in conveying service and information to users in software design. Their user research is not associated with any specific software system, although it is inclined to help the user interface design. Some of the findings and analysis are helpful in a lot of system designs. For instance, the analysis of window behaviors, using controls, toolbars and tool tips, and dialogs, are involved in almost any system's design choices if this system has human users (Cooper 2003).

Some systems utilize user models to filter or customize web-based information. The architecture of these systems usually has functions that can select and deliver multimedia objects in a large-scale network environment. As Kay and Kummerfeld (1996) proposed in their paper for "User model based filtering and customization of web pages", the architecture involves filtering information objects saved at a publication centre and only forwarding information about interesting objects to the user. Each object is accompanied by meta-data that contains information about the object (such as ratings, keywords, etc.). The filtering is carried out using information about the user from a user model and by examining the meta-data or actual data of the objects.

The filtering process of Kay and Kummerfeld's model (1996) is carried out by agent programs that are constructed from a user model and registered at a publication centre. Agents examine objects as they are published and decide whether to ignore the object, forward it completely to the user or forward only the meta-data. The architecture is a "server push" system that allows pre-emptive caching and filtering. Popular pages are multicast to regional publication centers where they are examined by user agent programs and, possibly, forwarded to a user's home system for viewing. This avoids the problem of network overload and also aids the user in finding new and interesting items since they are notified if they are accepted by their personal filtering agent. The filtering system architecture is shown in Figures 2-4 and 2-5.

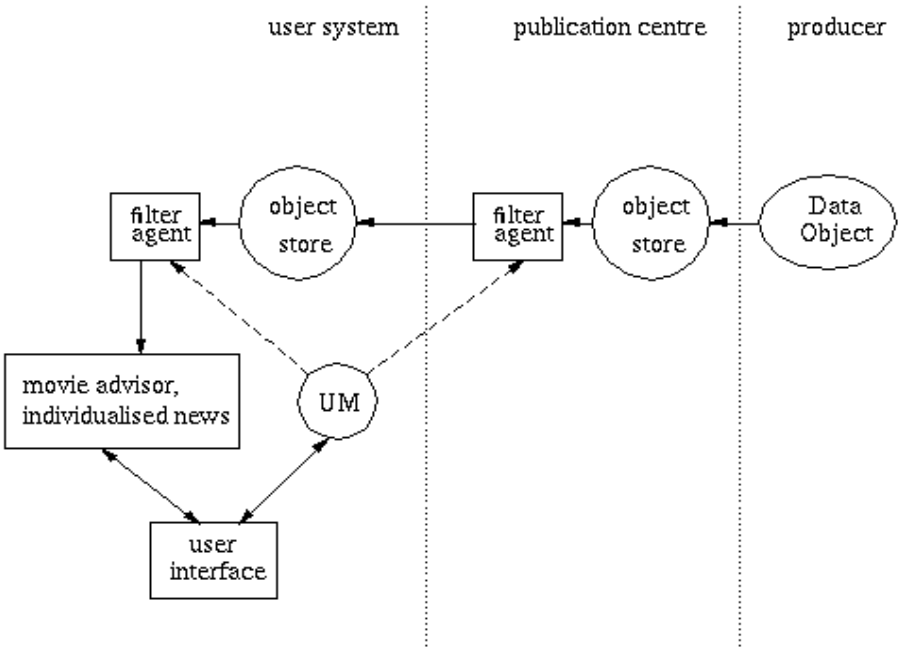


Figure 2-4 Filtering system architecture of Kay and Kummerfeld's model (1996)

Information about a user's preferences in a particular domain is extracted from the user model (UM) and embedded in a small filtering agent program. This program is sent to a publication centre using a message delivery system. Newly published objects (web

pages etc) are passed to the agent program for analysis. In order to support customization of WWW documents, Kay and Kummerfeld proposed using metahypertext.

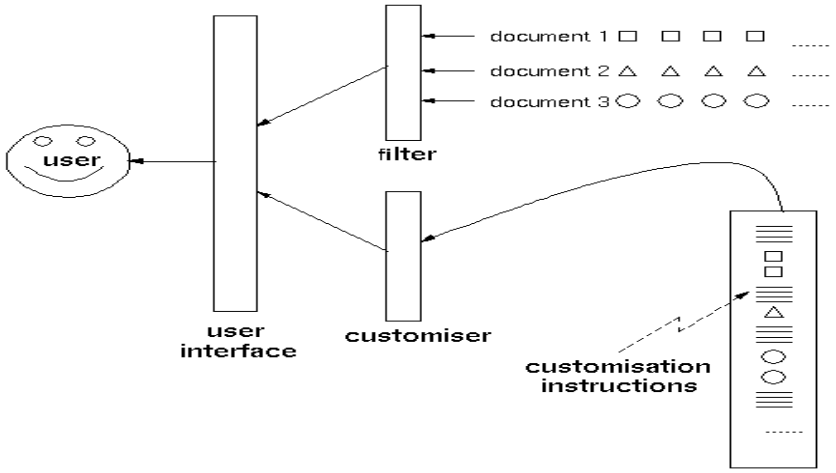


Figure 2-5 Overview of the filtering system of Kay and Kummerfeld's model (1996)

Each of the solutions for a specific group of intended users can be published at a publication centre. When the user issues a request for a topic, their agent goes to the publication centre and filters all documents to find those that are substantial hyper-books about that topic. In addition, the agent carries relevant parts of the user model, aspects about their knowledge and preferences related to programming languages and learning these languages. The lower part of the diagram illustrates an equivalent process based upon customization. In this case, the author of a metahyper-book takes account of a range of possible classes of users and publishes this at the publication centre. Then the user model must be used to customize the metahypertext to produce the deliverable result.

The metahypertext proposed by Kay and Kummerfeld concerned with the creation, management and delivery of WWW documents that match a user model. But it has a blurring between the role of filtering and customization at least from the perspective of the user who seeks simply the best documents for their current purposes.

Alpert et al. (2003) talked about adaptive interaction techniques and technologies, as well as collaborative filtering in eCommerce in their newly published paper. Alpert and his colleagues conducted user studies with initial and primary purpose as to determine which specific personalization features would be judged the most usable, valuable, and attractive to users of an eCommerce Web site. They clustered 75 techniques into 14 groups listed as follows:

- PersonalBook: a distinguished place on a site where all personal data can be accessed and modified
- Universal profile: a single place where user information is persistently stored; used by all functions on the entire site
- Subscription-based services
- Personalized service and support
- Recommendations based on profile data
- Adaptive presentation tailored to user characteristics
- Personal Preferences in page layout or format (customization)
- Adaptive navigation
- Live (chat-like or phone-based) help or sales support (Personal Shopper)
- Personalized feedback, such as indication that user is recognized, personalized messages
- Accessible transaction history
- Loyalty programs, incentives
- Future purchase considerations
- Your personal store, built by an expert

From the results of the study by Alpert and the colleagues (2003), users have fervent desire to be in control. Users will not provide personal information and will not

perform transactions on a Web site that in their opinion cannot be fully trusted to use profile information in an ethical and private manner. Users want to review, modify, delete, and add personal information at any time. Users do not want any information that has been collected implicitly by the system to become a part of their persistent personal profile. They like to have content of their search results filtered based on personal profile information. Design of personalization or user-adaptive systems (or any software technology) cannot occur “in a vacuum”, specifically, it cannot usefully proceed without assessing the value and usefulness to users of the concepts proposed and implemented by researchers and developers. Users want to feel as if they are in control, which has a significant impact as to whether the user can readily make sense of the interaction with a site.

The reluctance of users to provide personal profile information will affect the attainment of building up the filter agent. For the case study example of the coil door in the restaurant project, if the user wants to find out some information regarding this coil door, their user profile information must first be obtained. For instance, s/he may be a superintendent, a project manager, or a company president. If the user is a superintendent, s/he will need to know the length, width, and thickness of this coil door, what gauge of metal it is, and what kind of backing or blocking it requires. If the user is a project manager, besides the above information, s/he will also need to know the price and the delivery date. If the user is a company executive, s/he only cares about whether the door is within budget. Based on their different preferences, the system can present different details of the coil door to the user. Table 2-2 shows a comparison between previous research projects and the proposed project. It compared the projects based on their

research purposes, user types, data structures, designed for network uses, key properties, influence on system design, and influence on General User Interface. While previous studies involved users just at the beginning and end of the software design process, in this study the user is fixed in the task and user profiles and is involved throughout the process.

2.5 Overview of Construction Software Development and Modeling

Project management needs to consider a lot of things including the management of schedule, labor, material, and investment. The shift from mainframe software to personal computer (PC)-oriented programs has led to the usage of computers by the majority of construction companies. The rise in use of Internet brings new technological capabilities in the field of construction management. It can affect the way superintendents and engineers on jobsites to obtain product information, to write contracting documents as well as to control time, quality, and cost (Garza and Hewitt 1997).

The productivity and product quality of the construction industry can be dramatically improved by computer applications. However, during the construction project management, information fragmentation and large quantity of documentation became the characteristics of the construction industry (Garze and Howitt 1997). These characteristics have evolved with increased specialization in building codes and disciplines, and are exacerbated by the present complexity of construction projects and the industry business model. Traditional documentation on drawings and specifications captures only the final product of building design decisions. Contractors and engineers have to do a lot of imagine work based on these static and isolated information (Mays and Novitski 1997). This kind of lack gives rise to an inadequate understanding of the logic

Table 2-2 User models comparison

	Research Purpose	User Type	Data Structure	For NT	Key Property	Affect System Design	Affect GUI
IBM user model: Watson Research Center (Vredenburg et al. 2003)	Collaborative User Experience	General	On-line communities and large-scale collaboration	Yes	Users collaborate with designers	System functions are affected	Yes
User model based filtering and customization of web pages (Kay and Kummerfeld 1996)	To filter or customize web based information	General	For selection and delivery of multimedia objects in a large scale network environment	Yes	Agent prog's are constructed from user model and registered at a publication center	Searched results are affected	No
Microsoft user model: Lumiere Project (Horvitz et al. 1998)	For Inferring the Goals and Needs of Software Users	General	Developing a language for transforming system events into observational variables represented in Bayesian user models and an overall architecture for an intelligent user interface	No	predict based on the known history	No influence	Yes
General User Modeling System (GUMS) (Finin 1989; Finin and Drager 1986)	To find out basic functions of user modeling system	General	No implementation	No	application-independent user modeling systems	No influence	No
The Proposed User Model	To efficiently retrieve the needed information	Construction Project Manager	Use tree-type structure to store and organize related data and configure system environment. Use Element Ambassador and Navigation model to retrieve information.	No	User-model-driven architecture	UM configures GUI and applications	Yes

and rationale behind the design decisions that must work in concert to achieve a global set of project objectives (Vorster al. et. 1998).

The construction industry now needs to better integrate the activities in design, fabrication, construction and operation of constructed facilities through the use of computer technology (Wilson and Bryan 1994). Integration of systems has become a familiar topic of interest to many professionals, including those in the AEC industry. From an information systems perspective, integration can be defined as the design and development of information systems that combine several hardware and/or software components to cooperate and carry out a joint task that would be beyond the capabilities of any one of them individually.

System integration in the AEC industry can be applied at three levels:

- Inter-application integration: This involves combining the computer applications of one company into one integrated system. These applications can then share data and call each other's procedures.
- Inter-system integration: This exists when one company's applications integrate with those used by other project participants.
- Industry-wide integration: This will allow any project participant to communicate electronically with any segment of the industry (e.g., owners, designers, suppliers, financiers, regulators, etc.). It would also ensure consistency across different projects.

A more widely used term in the construction industry to represent integration is Computer Integrated Construction (CIC) (Karttam and Levitt 1990). CIC can be defined as a business process that links the project participants in a facility project into a collaborative team through all phases of a project. Figure 2-6 shows an overall CIC framework. The framework includes both computer-aided drawing/design (CADD), and visual computing and computer aided engineering (CAE).

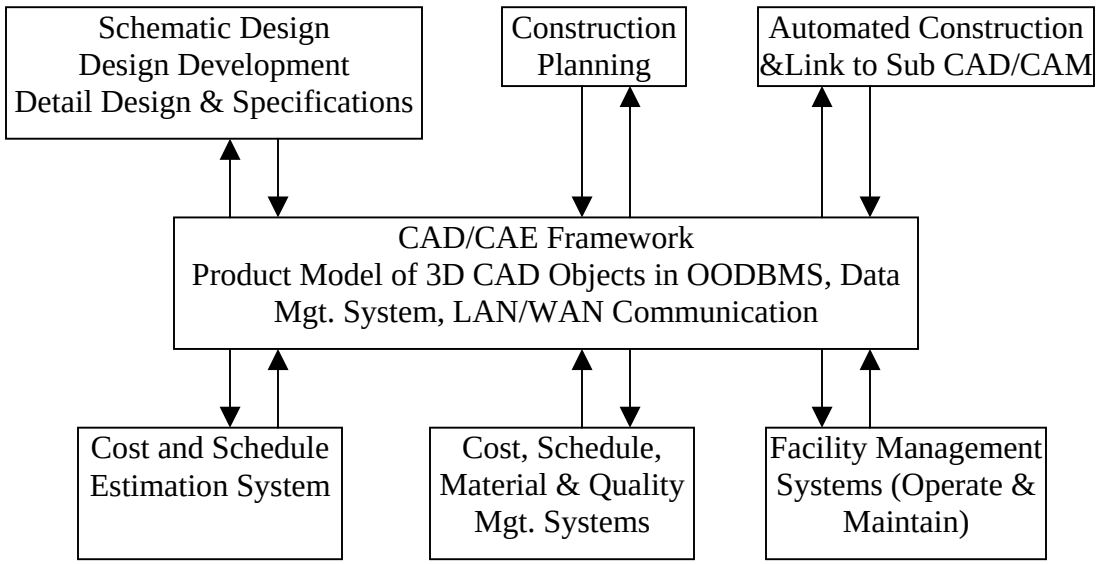


Figure 2-6 Computer Integrated Construction technology framework

One characteristic of construction software systems is that the systems divide the multi-task and complex work into separate jobs and one specific system will only provide service to solve problems in one field. For example, Timberline software primarily eases the task in estimating; AutoCAD helps with the drawings; while a lot of people use Primavera Project Planner (P3) to do schedule control and other use Primavera Expedition to manage construction contracts, change orders and other documents.

If dividing current construction software products according to their targeted market, they can be grouped into two kinds: one is for general construction management; the other is for special contractors, for example, heavy construction (highway, utility, tunnels, earthwork, bridges, environmental, paving, dams, wastewater, pipeline, marine and railroad, etc.). The grouping criteria can also be set as the different stages in the construction process. Then the software will belong to one of the categories (such as design, bidding, earthwork, structure, concrete, etc.). Since construction projects have many participants, it is also reasonable to divide software according to users (real estate agents, contractors, material suppliers, designers, estimators, schedulers, etc.) This study

divides the construction software into the following 11 categories to help the analysis and comparison of them. These are listed as follows:

- Document Management;
- Estimating;
- Engineering and Qualitative Risk;
- Graphical Design Tool;
- Job Cost and Project Accounting;
- Resource Management;
- Project Management;
- Project Planning and Scheduling;
- Training;
- Web-based Collaboration Tools;
- Other (Professional Services Automation, Mind Mapping, etc.).

Almost all of these construction software systems were built on process models, even though some implementations of user research can be found in the General User Interface (GUI) designs of several construction software systems. But the business process model is not the same as the user model. Business process modeling can help assess the impact of change, find an optimum business solution, and show where the business is today so people can draw a map of the future. Business process modeling is a powerful tool that can be used to analyze, document, and improve complex business processes. For example, the model can clearly document important factors such as what activities are needed, how they are performed, and what resources are needed. This provides an integrated picture of how an organization gets things done, from small department workflow models to complex node tree diagrams.

2.5.1 Estimating Software

The estimating output of Timberline includes spreadsheet estimating, productivity tools, and customizable databases. Timberline has modules for the cost estimating process, from conceptual estimate to final purchase order of material. It fits the usage in commercial and industrial construction, residential construction, electrical construction and many other specialties.

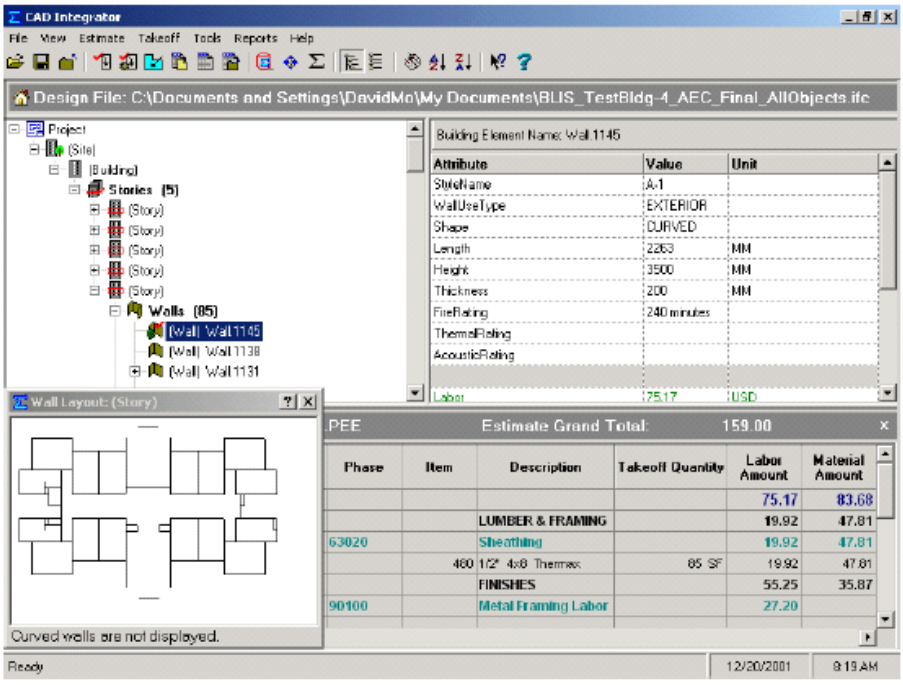


Figure 2-7 Computer Aided Design (CAD) Integrator Interface (Timberline 2004)

Timberline Estimating for the Windows operating system has been enhanced with several automation capabilities. It has the Windows versions of the IFC-compliant CAD Integrator, which assures compatibility with all IFC (version 2.0) compatible CAD applications (Timberline 2004). In addition, this newest release contains a number of enhancements such as the ability to calculate or re-price estimates using a number of different cost indexes based on the geographical location of project. Figure 2-7 shows how the CAD Integrator can recognize CAD objects and attributes, and maps them to

assemblies/items to eliminate any need for manual takeoff. Design changes made to CAD drawings can be automatically synchronized with the estimate to update cost information. With this integrator, a lot of labor and time can be saved, and it eliminates the risk of inconsistency due to a multitude of changes. For instance, if customer wants to remove a wall from the drawing of a building, the removal will be reflected in the drawing, the attribute sheet, as well as the quantity takeoff sheet. The automation and the convenience are built on the basis of standards. Suppose there is a project that has a lot of unregulated design items or parts and a building company needs to estimate the price of it. If the CAD integrator could not find the corresponding item records in its database, it cannot link the CAD objects with any attributes. In this situation the estimators of this company have to do the takeoff and estimating manually.

2.5.2 Graphical Design Tool

The usage of graphical designing tool is no longer limited to high-level technicians. With some training, ordinary users can design their ideal house or building by themselves. AutoCAD has tried to get rid of those hard-to-remember keyboard instructions, which used to be a necessary part of using it. As mentioned earlier, the standard components of a project can be found in the AutoCAD library and can be directly used in design. Integration gives further freedom and a more enjoyable experience in the design of a project. For some simple structures or buildings, users can draw drafts by hand on a digitizer and some computer software will help to translate the sketches into drawings.

Based on the 3D objects, people are adding another dimension: time, to form the 4D CAD. Schedule control is one of the main concerns of the project manager. With 4D representation, project managers will view the construction process and time control in

graphical format. 4D CAD will bring the virtual reality into the management of projects. If cost information is added into the 4D CAD file, it becomes a 5D document, which can at the same time be used for schedule control and cost control.

2.5.3 Project Planning and Scheduling

Primavera Project Planner® (P3®) is used in planning in the engineering, construction, architecture, utilities, and telecommunications industries. To organize projects of up to 100,000 activities, P3 provides unlimited resources and an unlimited number of target plans. P3 offers a single database solution (ODBC-compliant) that provides simultaneous access to project files by multiple users throughout the project. It also supports a wide variety of data exchange formats. It uses Object Linking and Embedding (OLE) to link information from other applications and provides Client/Server operation for accelerated processing and SQL access for reporting on the project database (Primavera 2004).

P3 is one of the software systems that concern users' needs and provide many functions to ease usage. Multi-project control of P3 allows users to roll up information to analyze and report on project progress. Hence the user can define any number of interrelated projects and project groups to reflect complex links among individual projects. On the other hand, the multi-user capability helps project teams simultaneously update, analyze and report on the same project. P3 uses individual record locking, providing maximum concurrent access to data. P3 can restrict access to certain project data by task, resource, phase or user-defined filter. Project managers can assign read/write, read-only, exclusive-access or no-access on a project level. There are over 150 customizable reports and graphics to report on project goals and priorities in P3. Layouts and reports can be saved as HTML format for use on a project web site

(Primavera 2004). There are 24 user-definable activity codes in P3 for selection and sorting, custom data items, and project template library (fragnets).

P3e/c™ for Construction (P3e/c) empowers engineering and construction professionals to mitigate project risk through powerful schedule analysis, accurate cost forecasting and streamlined coordination among designers, contractors, and owners. P3e/c is an integrated solution with web-enabled, client/server, and desktop software that provides role specific capabilities to satisfy each team member's needs, responsibilities and skills. It simplifies contractor schedule integration, enhances collaboration, and has the versatility to support teams that manage single projects or complex programs. Primavera Mobile Manager enables companies to take advantage of the mobility offered by handheld devices. Utilizing handheld devices with the Palm OS® or Pocket PC platform, project managers and team members can review and update project status information at the jobsite, on the plant floor or in a conference room. It helps to collect status information from project teams efficiently and effectively.

2.5.4 Project Management Software

Besides dramatically reducing the expenses associated with paperwork, postage, printing, travel, telecom and courier services, project management software is also a fast, reliable delivery method for crucial documents without the limitation of time and location. Some of them support integration with other enterprise applications through comprehensive XML based APIs. With everyone following the same rules about the distribution of information, productivity will be increased, project management costs will be reduced, and control over the flow of information will be reduced.

ProjectNet is one of the products provided by Citadon for online collaboration on the design, construction and operation of capital projects, in order to reduce project risk,

shorten project schedules and reduce overall project cost. With full access control and password protection, documents will only reach the people who are intended to see them. Leveraging team resources and putting them to better use can save time for every team member. By automating communications, ProjectNet reduces the delays normally associated with global engineering and building projects (Citadon 2004).

2.5.5 E-Commerce Systems

Local area networks (LANs) are emerging in some of the US construction companies, which connect through a wide area network (WAN) to remote location. E-bid and e-Builder™ can be regarded as some of these applications. Vendors are working on business process integration that can work across company boundaries. For example, Sharp Electronics has deployed a B2B integration engine that uses XML document interchange messaging to communicate with Sharp's trading partners. But the integration solutions often have limitations, such as inability to interoperate with platforms from other vendors. While they all support XML, it does not necessarily mean that they support the *same* XML. For instance, people have found that even with the help of sophisticated integration products, differences in XML vocabularies can still require human intervention.

One promising new XML-based B2B standard--the Universal Description, Discovery and Integration (UDDI) specification--could facilitate ad hoc relationships by allowing companies to describe their services in a central registry so they can be discovered and utilized by other businesses.

2.5.6 Future Trends of Process Modeling in the Construction Industry

As illustrated before, process modeling has been widely used in construction software design. In process modeling, the design perspective is of, from and for an

organization. In doing process modeling, inefficient, defective, or redundant activities need to be detected (and subsequently improved, replaced, or eliminated) in accordance with the corporate objectives. In addition, the impact of the change must be fully understood and communicated before a system can be designed to accommodate it. The model should automate many of the housekeeping tasks, which are usually associated with building process models, and provide the logic needed to ensure correct and consistent results (consistency is especially important when the model changes). In addition, the model needs to support User Defined Properties that can capture information relevant to user needs. The functions supporting User Defined Properties are only combined parts of the process model. Chapter 6 will discuss how user research and user modeling of the proposed project will be integrated with the business process and implemented through the whole project. In the proposed project, users are not ignorable symbols in organizations. The needs of users are not limited to the services provided only by the attached patches of the system. In the proposed project, the design perspective is of, from and for users and organization.

The model should allow multiple design groups to analyze separate parts of the business, so a global view can be easily created. Globalization of markets and brands is compelling companies with different operating units round the world to act as one integrated company. Increasingly sophisticated consumers are expecting more choices. Increasing cost pressures are focusing attention on every part of the value chain. These factors are creating a demand for a coordinated and integrated business strategy that is only possible through the sophisticated business integration of computer systems. For many companies, automated business processes enable innovation. Accordingly, when

people talk about integrating their business processes, a large part of that is improving the collaboration between their computer applications.

For complex operational environments, reuse becomes an issue that explores the time-varying (dynamic) interactions of business processes. Resource allocations and process flows can then be optimized to meet target workloads. Simulation provides the ability to explore the effect of change dynamically. Different scenarios can be tested before change is implemented, ensuring an optimal solution to a business requirement.

2.5.7 Comparison between Workflow Model and User Model

When establishing a new user in a system, it also instantiates and activates a set of workflows, each associated with a context in the user model structure. So, for example, the system will create a user model context for the user's knowledge of leave processes within the organization. This starts one workflow for that user for each of the forms of relevant leave. Similarly, the user model context for handling examination scripts is associated with a workflow for that process (Davis et al. 2004). When a workflow is initiated for a user, a context is created for that workflow. The user model for that workflow is initialized to a stereotypic set of values and operated as default assumptions, which are intended to be overridden once more reliable information becomes available.

In 1996, the Workflow Management Coalition published a glossary of all useful terms related to workflow. It defines workflow as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (Allen 2004).

Systems acquire knowledge about their users and about the degree of knowledge sharing among them (User Models) and about user's roles and expertise (User Profiles).

The User Model is a partial view of the work domain, which maintains the knowledge each member acquires in the communication with other group members.

2.6 Implementation of User Research in Information Retrieval

The general area of information access and retrieval aims at modeling, designing and implementing systems able to provide fast and effective content-based access to a large amount of information. The aim of such systems is to estimate the relevance of the documents to a user's information need. Information can be of any kind: textual, visual, or auditory. The objective is pervaded with subjectivity, vagueness and uncertainty.

2.6.1 Information Retrieval versus Data Retrieval

Data retrieval, in the context of an Information Retrieval (IR) system, consists mainly of determining which documents of a collection contain the keywords in the user query, which, most frequently, is not enough to satisfy the user information need. In fact, the user of an IR system is concerned more with retrieving *information* about a subject than with retrieving data that satisfies a given query. A data retrieval language aims at retrieving all objects that satisfy clearly defined conditions such as those in a regular expression or in a relational algebra expression. Thus, for a data retrieval system, a single erroneous object among a thousand retrieved objects means total failure. For an information retrieval system, however, the retrieved objects might be inaccurate and small errors are likely to go unnoticed. The main reason for this difference is that information retrieval usually deals with natural language text, which is not always well structured and could be semantically ambiguous. On the other hand, a data retrieval system (such as a relational database) deals with data that has a well-defined structure and semantics. Data retrieval, while providing a solution to the user of a database system, does not solve the problem of retrieving information about a subject or topic. To be

effective in its attempt to satisfy the user information need, the IR system must somehow "interpret" the contents of the information items (documents) in a collection and rank them according to a degree of relevance to the user query. This "interpretation" of document content involves extracting syntactic and semantic information from the document text and using this information to match the user information need. The difficulty is not only knowing how to extract this information but also knowing how to use it to decide relevance. Thus, the notion of *relevance* is at the center of information retrieval. In fact, the primary goal of an IR system is to retrieve all the documents, which are relevant to a user query while retrieving as few non-relevant documents as possible (Baeza-Yates and Ribeiro-Neto 1999).

2.6.2 Information Retrieval for Adaptable System

In the past 20 years, the area of information retrieval has grown well beyond its primary goals of indexing text and searching for useful documents in a collection. Nowadays, research in IR includes modeling, document classification and categorization, systems architecture, user interfaces, data visualization, filtering, and languages. Despite its maturity, until recently, IR was seen as a narrow area of interest mainly to librarians and information experts. Such a tendentious vision prevailed for many years, despite the rapid dissemination, among users of modern personal computers, of IR tools for multimedia and hypertext applications. In the beginning of the 1990s, a single fact changed once and for all these perceptions: the introduction of the World Wide Web.

The Web has become a universal repository of human knowledge and culture, which has allowed unprecedented sharing of ideas and information in a scale never seen before. Its success is based on the conception of a standard user interface that is always the same no matter what computational environment is used to run the interface. As a

result, the user is shielded from details of communication protocols, machine location, and operating systems. Further, any user can create his / her own Web documents and point to any other Web documents without restrictions. This is a key aspect because it turns the Web into a new publishing medium accessible to everybody. As an immediate consequence, any Web user can push his personal agenda with little effort and almost at no cost. This universe without frontiers has attracted tremendous attention from millions of people everywhere since the very beginning. Furthermore, it is causing a revolution in the way people use computers and perform their daily tasks. For instance, home shopping and home banking are becoming very popular and have generated several hundred million dollars in revenues.

Despite so much success, the Web has introduced new problems of its own. Finding useful information on the Web is frequently a tedious and difficult task. For instance, to satisfy his information need, the user might navigate the space of Web links (i.e., the hyperspace) searching for information of interest. However, since the hyperspace is vast and almost unknown, such a navigation task is usually inefficient. For naive users, the problem becomes harder, which might entirely frustrate all their efforts. The main obstacle is the absence of a well-defined underlying data model for the Web, which implies that information definition and structure is frequently of low quality. These difficulties have attracted renewed interest in IR and its techniques as promising solutions. As a result, almost overnight, IR has gained a place with other technologies at the center of the stage.

The motivation to develop a scalable data-warehouse and on-line analytical processing (OLAP) framework tackle the issue of scaling the whole operation chain of

construction industry, including data cleansing, loading, maintenance, access and analysis to keep up with the input data rates (Chaudhuri and Dayal 1996). Typically, challenge is to provide continuous, rather than one-time, analysis and mining of Call Detail Records. The summary information at multiple levels of granularity, e.g., hourly, daily, weekly, and monthly, must be stored and incrementally updated. This requires an architecture that supports multilevel, incremental data mining and efficient maintenance and combination of partial results. Further, the architecture must support the retirement of data from the warehouse, integrated into the process of multilevel data reduction and summarization (Chen et al. 2000).

The purpose of Information Retrieval (IR) is to find the information according to requestors' needs and user modeling servers to find out the right ways to response to these needs. Current commercial user modeling is very much behavior-oriented. Observed user actions or action patterns often lead directly to adaptations. In the context of this research, we have developed principles and arguments for the trade-off between *adaptive* and *adaptable* approaches in user modeling (Fischer 1993). The comparison between these two approaches is listed in Table 2-3. Adaptive system dynamically changes itself to fit current task and current user. It can actively help systems and user interface customization. Adaptable system is changed by user. User has more control to adaptable systems than to adaptive systems. Adaptable system fits better to domain models and domain-orientation. Our proposed system is domain specific and it is an adaptable system. But the disadvantage of adaptable system is that systems become incompatible; user must do substantial work; and complexity is increased (user needs to learn the adaptation component).

Table 2-3 Comparison between adaptive and adaptable systems

	Adaptive	Adaptable
Definition	Dynamic adaptation by the system itself to current task and current user	User changes (with substantial system support) the functionality of the system
Knowledge	Contained in the system; projected in different ways	Knowledge is extended
Strengths	Little (or no) effort by the user; no special knowledge of the user is required	User is in control; user knows her/his task best; system knowledge will fit better; success model exists
Weaknesses	User has difficulty developing a coherent model of the system; less of control; few (if any) success models exist (except humans)	Systems become incompatible; user must do substantial work; complexity is increased (user needs to learn the adaptation component)
Mechanisms Required	Models of users, tasks, and dialogs; knowledge base of goals and plans; powerful matching capabilities; incremental update of models.	Layered architecture; domain models and domain-orientation; "back-talk" from the system; design rationale
Application Domains	Active help systems, critiquing systems, differential descriptions, user interface customization, information retrieval	Information retrieval, end-user modifiability, tailorability, filtering, design in use

2.6.3 Utilizing Unified Modeling Language and Extensible Markup Language in Information Retrieval

The Unified Modeling Language (UML) was introduced in November 1997 by the Object Management Group (OMG). Founded in April 1989 by eleven companies, OMG began independent operations as a not-for-profit corporation. The OMG is structured into three major bodies, the Platform Technology Committee (PTC), the Domain Technology Committee (DTC) and the Architecture Board. The process of gathering and analyzing an application's requirements, and incorporating them into a program design, is a complex one. One characteristic of UML (in fact, the one that enables the widespread industry support that the language enjoys) is that it is methodology-independent. By using XMI (XML Metadata Interchange, another OMG standard), people can transfer the UML

model from one tool into a repository, or into another tool for refinement or the next step in development process (OMG 2004).

An older effort at business process standardization is RosettaNet, a consortium formed to develop standards for the IT, electronic components and semiconductor manufacturing industries. A key product of the RosettaNet effort is its set of Partner Interface Processes (PIPs), which are specialized XML-based dialogues that define how business processes are conducted between trading partners. The PIPs define processes for a range of business activities, such as inventory, pricing, sales management, order handling, product configuration and shipping (RosettaNet 2004). The Open Applications Group, a nonprofit consortium of software vendors and customers, has defined a number of XML-based business transaction definitions and business process scenarios in its Open Applications Group Integration Specification (OAGIS). And yet another newcomer is ebXML (electronic business XML), a joint effort of the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS). ebXML is an attempt to create an architecture for defining business processes and their associated messages and content (OAGIS 2004). Electronic exchanges promise to provide the next step in business process integration by acting as translation hubs, which enable companies to create more complex relationships using XML and specialized business process servers. Today these are limited by the lack of standards between different e-business platforms, but the efforts described above will eventually make it much simpler to integrate across platform boundaries.

The media hype surrounding XML might lead the average person to believe that it is possible to create an XML interface for all of a company's documents, as well as instant business process integration. But the pioneers in this arena are finding that it's not that easy. Integrating applications requires a considerable amount of effort even for simple things like sending an electronic purchase order. And it will only get more complex as companies try to perform more elaborate types of integration, such as tracking the flow of work between business partners or looking for new partners that can provide a particular service or capability. The problem is that different people in a supply chain have their own views of the universe, and they tend to look at products and their specifications differently. Companies might have various terms for the same item, such as calling a computer monitor a "display unit," a "screen" or a "flat panel display." Manufacturers might have different levels of depth to describe an item. For example, a company data element might just include a name, address and URL on one system, while a trading partner might commonly use the name, address, revenues, and industry sector codes. These different views will also arouse problems in the information retrieval systems. The questions that an information retrieval system needs to take care off are how much information and to what level of detail a user wants to get that information from the search result.

Chapter 3 will talk about Information Tunnels and the connectivity of them. The intention of Chapter 3 is to show that in the construction industry, Information Tunnels exist in different software systems and databases. People need to integrate systems to retrieve information.

CHAPTER 3
CONNECTIVITY OF INFORMATION TUNNELS

This chapter will discuss what information tunnels are, how they are formed, and how they should be connected. This chapter will discuss the software developers' efforts in information storage and organization.

3.1 The Formation of Information Tunnels

Large volumes of data are involved in the construction project lifecycle and that data is used in daily operations. Data is generated, captured and maintained by the corresponding parties. For example, the Architect generates AutoCAD drawings; Engineer generates the detailed site information; Project Manager (PM) generates the estimating, scheduling, and contracting information; the Material supplier generates the data about the ordered products. Despite this data abundance, many construction companies are unable to fully capitalize on its value because information implicit in the data is not easy to discern. However, even if a project manager was provided with all the information about a project, the decision making process is still a time-consuming effort because the project manager must be able to identify and utilize information hidden in the collected data.

Figure 3-1 shows some of the typical databases used by project managers or other stakeholders in the construction process, based on the characteristics of the information stored and its relationship to the project management process. In order to fulfill a task as simple as the one described earlier in the material management problem, the PM needs to go through all the databases to retrieve information.

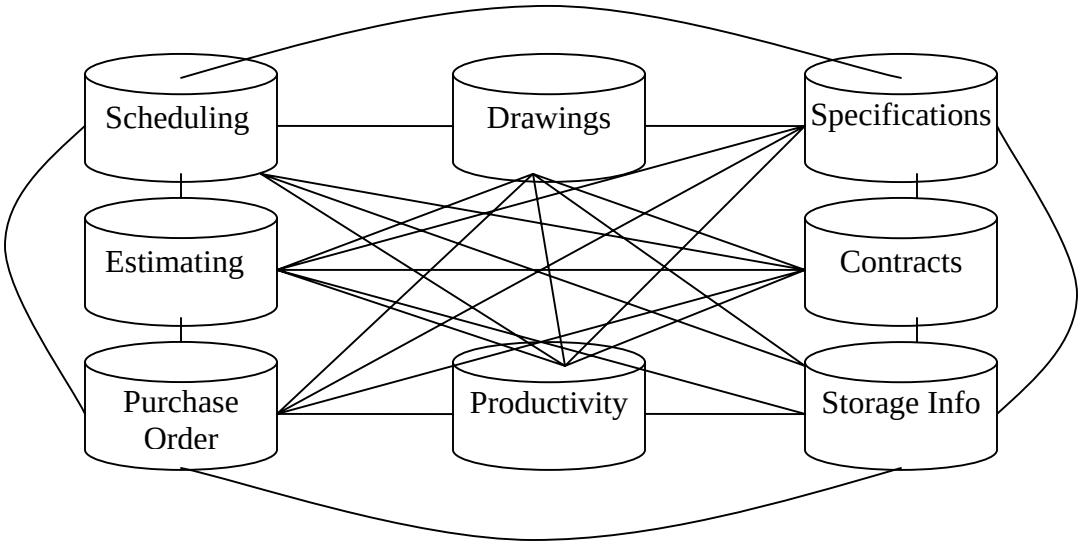


Figure 3-1 Relationships between Construction Management Databases (**Note**: There are (N*(N-1))/2 two-way point connectivity required to link all these databases. Here N = 8, so 28 edges are required to link eight databases.)

As Hendrickson and Au (1989) pointed out, both project design and control are crucially dependent upon accurate and timely information, as well as the ability to use this information effectively. They also concluded that too much unorganized information presented to managers could result in confusion and paralysis in the decision making process. In a construction company's database management system (DBMS), if a project manager wants to do some queries about the project status, such as the example of the material repository introduced earlier, he or she will face the situation described by Tannenbaum (2002) as information tunnels. According to Tannenbaum, "We are organized and often reimbursed based on the "tunnel" with which we are associated (e.g., accounting, product development, customer). It is only natural therefore, that supporting information is in line with that perspective. Information tunnels are typically based on processing requirements. Even though the same data may be required in more than one tunnel, it is usually not easy to go from one tunnel to another. In any case, data is

typically converted to tunnel-resident information, reflective of a limited set of requirements" (pp. 75-107)

In this case, the DB shown in Figure 3-1 is a simplified version of the information sets needed for project management. Each database in Figure 3-1 can be viewed as an information tunnel. In reality, not all databases are completely connected with each other. The relationship analysis should include integration points, which are the places where a given process touches or intersects with other processes. For example, management of the customer relationship and management of any contractor relationships often have few overlaps, hence few points of integration; but database management and disk-storage management likely will have many overlaps, and thus have numerous points.

Processes generally are of minimal interest to end users, who typically want to access the corporate network, Web, or other IT service without thinking about the infrastructure. The services define an organization's operations to end users within the company, who are in essence the IT department's customers. The services are what these customers "buy" from the IT organization. Upper management and executives rarely have to deal with the issues of information redundancy and integration. Their concern is whether or not the software system can answer all or some of their queries in an accurate and timely manner. But different queries have various depth of complexity. For example, queries about current material inventory will be easier to handle than the queries about material inventory after 2 weeks of storage. The degree of difficulty in answering these queries depends on how many databases the query needs to check. If the answer is available by only going through one database, the task of the query will be fast and easy to handle. Oftentimes, for the purpose of answering some meaningful queries by the

project manager or other users, the DBMS must be able to manage queries to multiple databases, such as the scheduling, the specification, and the estimating databases. To manage and query the multiple databases, the DBMS also has to build up the connectivity among the databases.

3.2 Connectivity of Information Tunnels

System integration over the past decade has primarily focused on the connectivity issues at the lower layers of the Open Systems Interconnection (OSI) network model. Proposed solutions exist for connecting systems at the Hardware level or the Transport level (Alban 2000). This kind of data sharing is based on common architectural features among relational database management systems (DBMSs); that there are middleware products to mix and match data from heterogeneous sources; and that the SQL (Structured Query Language) Access interfaces to allow non-relational data architectures to participate also. At present the Remote Procedure Call (RPC) is the most common technology used to implement this multipoint connectivity needed for application-level interoperability. Point-to-point solution is the intuitive method to implement the graph relationship in Figure 3-1 and it is supported by RPC. This implementation provides a quick fix with the trade off of adding complexity to the system. In addition, the point-to-point connectivity requires each object system to have a specific protocol to connect with one another (Xie et al. 2003). A standard (i.e. CORBA) is required if systems builders are to create a flexible, layered structure for connectivity between object systems and avoid point-to-point connectivity (Alban 2000). Again, huge amount of work has to be done in order to implement the CORBA specification.

3.2.1 Database Management System Connectivity

In the research and development of DBMS applications, Microsoft's Open Database Connectivity and Java Database Connectivity are two topics that attract a lot of attention.

Microsoft's Open Database Connectivity (ODBC) is an API providing a standard method for accessing databases independent of which database products are being targeted. This means that a program using ODBC does not need a separate access method for each database product that it supports. Java Database Connectivity (JDBC) is one of the technologies providing cross-DBMS connectivity and most database products that are compatible with ODBC are also compatible with JDBC. JDBC is advantageous when using the Java programming language, because its routines are native to Java. However, Microsoft programming languages have built-in capabilities for using Microsoft's ODBC, but none for JDBC. The Synchronizers of Microsoft's applications were developed using Visual C++ and Visual Basic.

JDBC is used to connect to the SQL Server. Applications can request not only relational SQL commands but also user-defined commands through JDBC, and JDBC gets values from the SQL server, then the values are sent back to applications. The JDBC is the tool to connect the Java applications to the relational DBMS. To establish the connection with the DBMS, there are three steps to do as follows:

Step 1: Loading the Drivers: It involves one line of code. In order, for example, to use the Oralcle driver, the following code loads it:
Class.forName("oracle.jdbc.driver.OracleDriver");

Step 2: Making the Connection: The second step in establishing a connection is to have the appropriate driver connect to the DBMS. The following line of code illustrates the general idea:
Connection con=DriverManager.getConnection(url,"Login","Password");

Step 3: Creating the JDBC Statements: A Statement object is what sends the SQL statement to the DBMS. For a SELECT statement, the method to use is executeQuery. For statements that create or modify tables, the method to use is executeUpdate. The following line of code is used to create the Statement and select records from database:
object ”stmt”:
Statement stmt = con.createStatement();.
Stmt.executeQuery(“select * from portfolio”);.
ResultSet rs= stmt.executeQuery(“select * from portfolio”);.

3.2.2 Interaction between the Modules

Most database systems accept SQL command (insert, delete, update, select, etc) or ECA command (extended create trigger statement which includes event definition). SQL commands are directly sent to the SQL Server through JDBC. ECA commands, on the other hand, are sent to the ECA Parser. The ECA Parser scans the ECA command and checks if the trigger name and the event name of the event are duplicates. Then the ECA Parser generates a Java source file.

Many software applications have the ability to connect database to database or database to General User Interface. The software applications, such as CGI, LiveWire, and WebObjects applications, connect to Web servers and relational databases. In addition, they compare and contrast these system architectures based on performance and data maintainability. Performance, in this context, can be measured by the speed that the Web server can serve Web pages to the browser under light as well as heavy usage. Varying memory, CPU, and file I/O demands of the described Web database systems can affect performance.

3.2.2.1 Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is one of the most powerful methods of providing dynamic content on the Web. CGI is a generic interface for calling external programs to crunch numbers, query databases, generate customized graphics, or perform

any other server-side task. ASP, PHP, Java servlets, ColdFusion , and CGI are the most ubiquitous server-side technologies on the Web. In the CGI approach, the database periodically dumps a table or a view into a tab-delimited text file. The database schedules this dump regularly or triggers it whenever data in the underlying tables change. Untouched Web pages are implemented by static HTML files. This static CGI approach suffers from poor performance because the Web server loads, executes and terminates a new CGI program for each user access. In addition, the approach involves extra processing and file I/O whenever data updates require rebuilding the text file. Large data sets and complex queries especially burden the system because data querying must take place in the CGI program instead of in the database (Lazar and Holfelder 1997). Rather than delegate complex queries to a relational database on another machine, the CGI program itself must perform such complex operations based on the text file data. Figure 3-2 shows the how CGI can be used in connecting the Web Server with the relational database.

3.2.2.2 WebObject

The three-tier WebObjects architecture cleanly separates data access, business logic and user interface, making it easy to develop flexible, scalable applications (See Figure 3-3). WebObjects generates HTML dynamically from the database as requested by the browser. As shown in Figure 3-3, the browser, ERP, or eCommerce systems requests Web pages from the Web server. Using ISAPI (Internet Server Application Programming Interface) or NSAPI (Netscape Server Application Programming Interface), the Web server passes the request along to a small program, called the WebObjects Application Server. The WebObjects Application Server, in turn, starts up the appropriate WebObjects application executable if it is not already running. The WebObjects

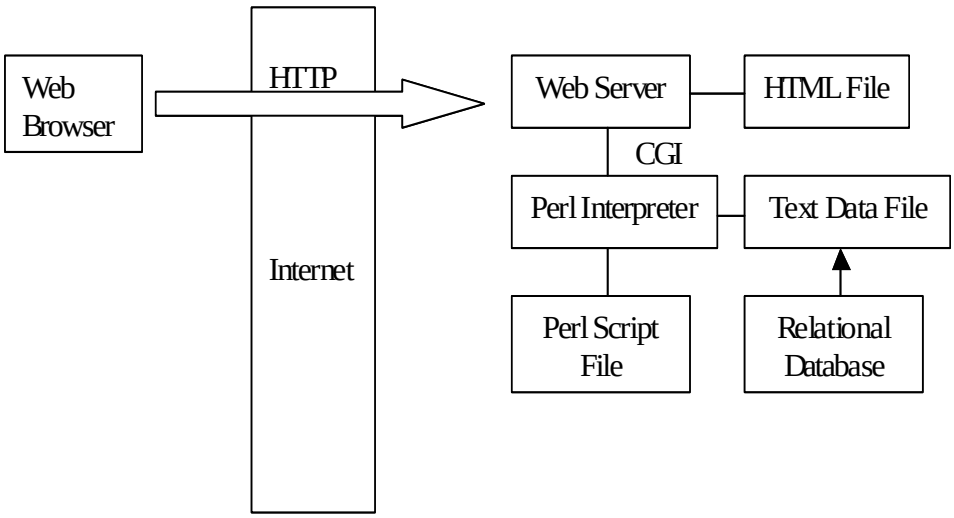


Figure 3-2 CGI architect using Perl (Lazar and Holfelder 1997)

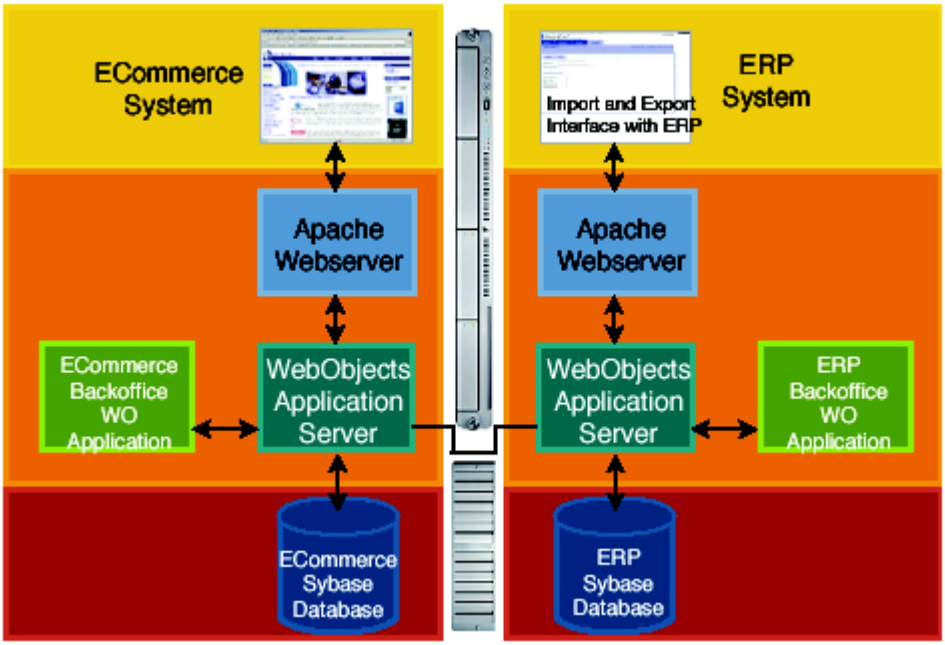


Figure 3-3 WebObject development workflow (Olson 2003)

Application Server serves as a communications bridge between the Web server and the application executable. Finally, the application executable generates HTML based on live data from the database, a user interface defined in the HTML template file, and application objects and logic defined in the declaration and script files (AppleGroup 2004; Lazar and Holfelder 1997).

WebObject Application Server can be distributed over more than one machine. In addition, the application executable remains running for multiple user accesses and only the relatively smaller application server program is instantiated with each user access. The lifetime of an application executable is called a *user session*. Developers can configure user sessions to last for a single Web page access or multiple accesses. The application executable helps minimize processing overhead by keeping database connections open for the duration of a session. If the database data changes, the subsequent Web access immediately sees this change (AppleGroup 2004; Lazar and Holfelder 1997).

In the material repository user model, it is assumed that the eight databases are object-oriented databases (OODB) and that the information about the project is kept on a hierarchical tree structure. These two assumptions are reasonable and critical to our information retrieval structure. First, in OODB, a system builder could select objects from several vendors and connect them easily. Individual programming modules (objects) would exist independent of the application for which they were created, free from recombination with other modules in new contexts and for new applications. The flexibility and independency make OODB an attractive choice in database design. Nowadays many software product vendors are providing object-oriented programming and OODB now. The second assumption about the hierarchical tree structure data organization is based on the features of the construction object and the project organization method. Hierarchical tree structure data organization is very common in OODBMS, whether it is at the initial design stage or at the maintenance or document audit stage, when people talk about a project, they usually start with the following

sequence: Building A → Floor B → Room C → Wall D →… This kind of query can be easily handled by a tree architecture (Xie et al. 2003).

3.3 Information Retrieval in Information Tunnels

As long as people want to retrieve information from more than one database and/or they need to derive results through two or more software programs, the information tunnel scenario will exist. In order to search for information from multiple databases, people need to use search procedures to go down each of these databases and find the data in their computers' hard drives or from servers. The metaphorical example of the search procedure is a train running inside a tunnel. If there is no connectivity between tunnels, the train will have no way to turn into another tunnel. The same condition will happen to a task involving multi-programs. Synchronized multi-procedures can be used to run these programs to save time, but the integration or connectivity among these procedures is still a topic of ongoing research.

Information retrieval and information integration or connectivity are undividable. In this chapter, the information connectivity research is organized into three levels. The first level is data level connectivity, the example for which is metadata research. The second level is application level connectivity. One example is XML applications. Another example is wrap application; many wrap applications have information integration attribute. The Third level is the system level (which is comprised of standards research, system integration, OLAP, etc.).

3.3.1 Data Level of Information Connectivity

Imagine how many times a construction project manager needs to go back to the original estimate spreadsheet to find out information? For example, a project manager needs to know whether a lockset or a closure for a door has been included in the estimate.

But if the project manager could not find such information in the estimate spreadsheet, the next step will be to find the information in the subcontractor's original proposal. Sometimes, the original proposal is so vague; there is a need to go beyond that immediate scope, generally phone calls or facsimile to the subcontractor requesting the information are the obvious ways to find out exactly what data is available. No matter what kind of computer applications people are using (i.e. ERP systems) Data warehouses, and e-commerce applications, to name a few, the purpose of using these applications is to redesign systems to make information retrieval and analysis easier. Thus, what data is stored in the systems is an unavoidable question faced by the database designers. Besides this question, the other concerns are what does the data mean, where is it, how did it get there, and how do people get it (Tannenbaum 2002). Database developers will confront these questions when they start thinking about the data structure of the databases.

Data is useless without metadata. Metadata is often defined as the "data about the data" and most organizations perceive it as representing descriptive information (element names, definitions, lengths, etc.) about the populated data fields. In fact, metadata is much more. True metadata involves not only the descriptive information pertinent to the users of the items in question, but also the generation, maintenance, and display of the items by tools, applications, other repositories, and the users. Unfortunately, very few of today's metadata deliverables consider this bigger picture (Tannenbaum 2002). The importance of metadata to database applications is evident and people are aware that metadata provides the solutions to the information connectivity problem in the database systems.

Imagine the following situation: a construction company is building a very large project, with millions of dollars, thousands of work crews, hundreds of subcontractors, and tremendous amounts of data. The project manager of this project wants to do some "cross analysis" to compare the data that is stored in different applications. As is expected, one of the most frustrating issues with performing this type of analysis is the user's inability to link or combine information about the same (or what appeared to be the same) item. Due to the lack of standards the same item may not always be identified or even named consistently across applications. Suppose a construction company has a material management database for this project. The project manager is the individual responsible for direct interface with the data. The ideal process of designing a database would be to first gather the requirements on metadata, then to coordinate them, then to design a metadata store (a standalone small client database, often implemented in the DBMS of convenience), and then to define the database. This explains why so many self-implemented standalone metadata stores are small MS Access databases, MS Excel spreadsheets, or MS Word documents. Nevertheless, in all situations, the documentation on the data has a place to reside, and the person who puts the documentation there knows exactly how to find and retrieve it (Tannenbaum 2002).

The benefits of building metadata-stores are that they give the user improved ability to locate information; interpret information; and to integrate data. The methods of accessing and exchanging metadata include, as shown in Figure 3-4, which shows the details of metadata management flow, include the follows:

- Application Program Interfaces (APIs), standard routines that can be called from within metadata access programs, remaining in that program's control;

- Remote Procedure Calls (RPCs), standard routines that are executed by metadata accessing programs, outside of that program's control
- Embedded Processes, which are associated with distinct components of a standard meta model;
- Standards-based Languages, which are defined by standards organizations and available for use by metadata accessing programs, In many scenarios, these languages are combined with interfaces and required architectures (Tannenbaum 2002).

The three-tier, Web-enabled architecture, as shown in Figure 3-5, resembles many data warehouses, with a batch ETL function populating the metadata repository independently of its presentation capability. A custom developed SQL Server database will serve as the centralized metadata repository, populated on a regular basis via scheduled batch refreshes (every two weeks). The central metadata store contains all common metadata, but also represent the only integrated view of its connections to the unique and specific metadata that remained in its source metadata stores. This metadata solution represents an integrated metadata solution architecture, since the central resident metadata is available to software that lies outside the framework. In addition, the sources of metadata are well integrated with the central metadata store through automated refresh practices.

3.3.2 Application Level of Information Connectivity

To support information sharing, the access to both existing and reconfigured information has to be a part of the plan. In the best situations, not only is the data itself standardized, but also its access. When searching or buying data, to some extent a decision needs to made as to which files to select, their inherent file types being part of the package deal. On the other hand, what type of software application and how to implement it are the decisions that relate to the certain data types stored in databases.

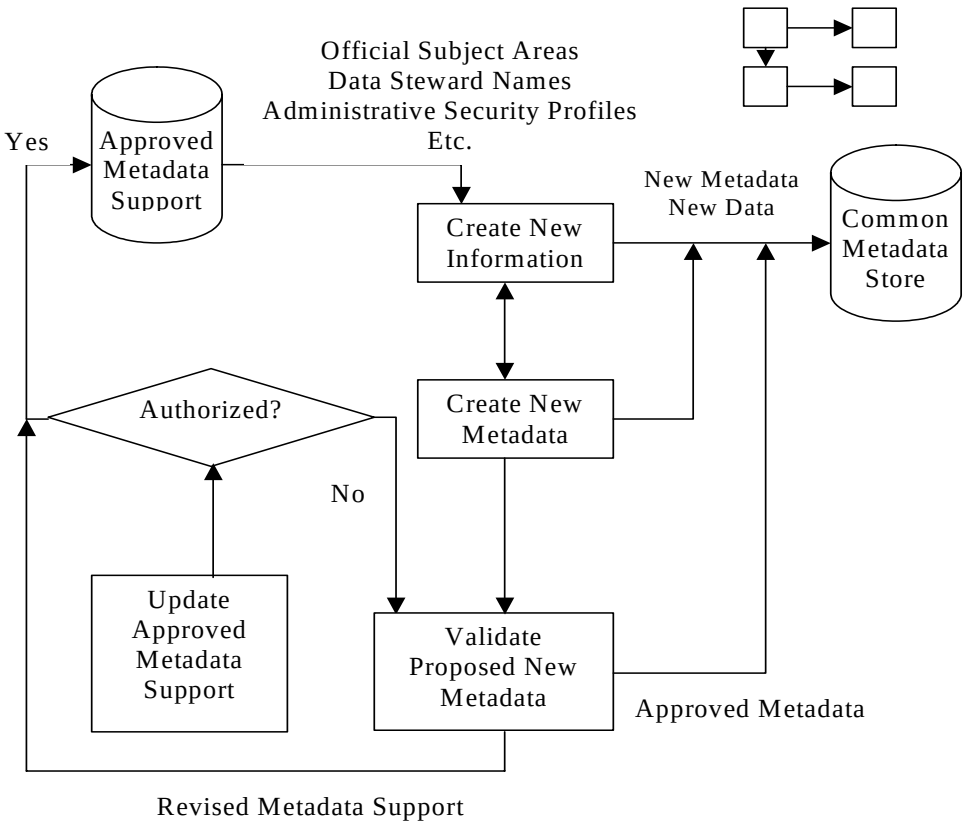


Figure 3-4 Sample metadata management flow (Tannenbaum 2002)

Within a machine and within an application, there are many options for how data is distributed. The storage and indexing options for the data (as provided by tools) and the mapping of any particular schema to a particular set of storage and indexing options determine the order-of-magnitude performance differences between different choices for both options. Pendse and Creeth (1995) discussed and compared the differences between some commercial multidimensional structuring techniques in their compression or indexing techniques, speed efficiency, space efficiency and their typical usage. For example, in order to quickly access and efficiently store sparse data, users may store data in sets of small, relatively dense arrays, and indexes into the arrays. For moderately sparse data without too many dimensions, data can be stored in a fixed record table with B-tree indexing technique, which allows for both reading and writing.

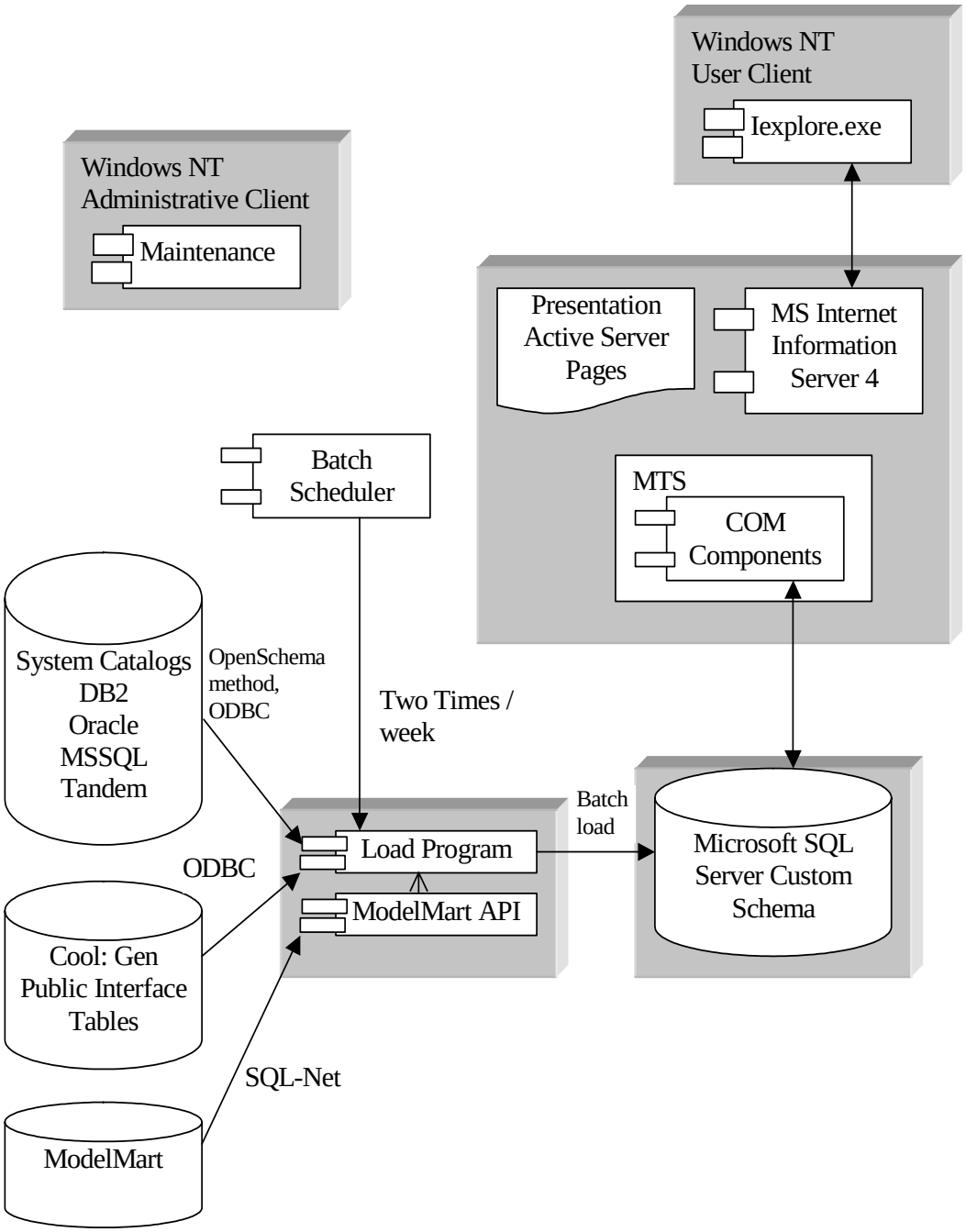


Figure 3-5 Metadata solution architecture (Tannenbaum 2002)

Software applications can act as linkages of databases to the external data world. For example, the linkage will be built up if the databases need to receive data. Those linkages may be as simple as a table import facility that is periodically invoked. A manual approach to maintaining synchronicity between data sources and cubes usually is

implemented here. All data processing products, from spreadsheets to statistics packages (i.e. SAS) have these capabilities. Some tools support user declared links without procedural code; some ask users to code the transformations (i.e. a script or program) by hand. Links provide a temporary or persistent infrastructure for importing and exporting data and metadata. They may vary as a function of the type of information brought into the databases and the type of data structure from which the information is obtained. Basically the links include structure link, attribute link, and content link. Tables that contain structural information (member identity and hierarchical relationships) are linked with structure links. Tables containing information other than structural relationships that are associated with members can have that information brought in through attribute links. Tables containing values for variables need to be attached with content links.

Links can be read only or read/write. Links can be static or dynamic. Static links are not capable of processing changes made to the source and are used only by the caller process when it loads or refreshes its information from the source. Dynamic links maintain an open connection between the source and the database wherein changes to the source are propagated to database.

For multiple users to share within a departmental or an enterprise framework, there will be a variety of machines existing in some kind of network topology and the linkage of data sets will be affected by the distribution of the data. The factors that influence the decisions of how best to distribute data across machines include machine functionality, bandwidth over different links in the network, machine locations of majorities of sources and/or queries of data, and firewall location. This topic is beyond the scope of work of the proposed research.

Among these diverse applications, Extensible Markup Language (XML) achieves a lot of attention. XML is a technology that attempts to address the areas associated with categorizing Web content. Briefly, XML allows previously unstructured organizational data (e.g., internal reports and email messages) to be categorized automatically with rudimentary, searchable metadata structures. These metadata structures are probably the first to separate style from content. In fact, the natural partner to XML is XSL (Extensible Style Language), which addresses the formatting aspects of Web documents. Because XML was intended to be a markup language, it was designed with meta-tags in mind. However, as with most limited metadata solutions, the meta-tag perspective is focused mainly on the ability to locate the information described by or surrounding XML tags. The proper use of XML would allow predefined XML values to result in consistent retrievals, with virtually no variation. XML was designed to be strictly standard and able to be parsed via "XML parsers" with same results every time (Tannenbaum 2002). XML is a standard syntax for defining custom tag collections. XML is a meta-language that is language for defining languages, which standardizes the syntactic rules whereby users can define their own sets of tags, suited to the needs of a specific application domain.

3.3.3 System Level of Information Connectivity

The information-tunnel phenomenon will continue to exist because of the difference in developing software systems between companies and the resulted un-integrated systems. Besides the data level and the application level methods of connecting information tunnels, the system level of information connectivity is very promising in eliminate or alleviate the isolation between information tunnels. Possible ways of system level information connectivity include making standards for software system design, system integration, and special systems dealing with fast and efficient data

access. CORBA is an example of system design standards proposed by OMG (Object Management Group) and will be discussed in 3.3.3.1. System integration is comprised of lots of integration applications. The proposed project of this research can be categorized into this group. The detailed analysis of the structure of the system can be found in Chapter 6. As an example of special systems dealing with data access, OLAP (On-Line Analytical Processing) will be discussed in 3.3.3.2. The commercial multidimensional structuring techniques can be organized into four groups: array, fixed record table, fixed record table in an RDBM, and variable length record structure. Table A-1 in Appendix A shows the comparison of these four structures.

3.3.3.1 Common Object Request Broker Architecture (CORBA)

It is possible to make those download files fit into a basic file system meta-model by associating specific access procedures with Web-based file types. The decision as to which software product and/or executable to associate with a particular Internet file type is left to the person responsible for populating this meta model. In most cases the workstation user has to make the decision. The accessing of file-based information is restricted to file-name-based access, and requires the file's name and type to get to the file's contents. If file names are neither indicative of their contents nor identifiable, this meta-model's weaknesses start to unfold. When considering the need to access files on the Internet, the model would require additional location-specific information beyond a file's name and type and would include network and protocol specifics.

Standardization efforts attempt to resolve the information tunnel phenomenon in a reusable way. The OMG (Object Management Group) strives to standardize the language used to define new interfaces, as they are required. In the Common Object Request Broker Architecture (CORBA), a request broker sits as the basic mechanism for receiving

and channeling requests. The Interface Definition Language (IDL) is the standard language required for these requests, and is independent of programming language, but contains mapping to popular programming languages so that applications can send and receive requests to CORBA facilities. The overall architecture defines the structure, interfaces services, and objects (Tannenbaum 2002).

3.3.3.2 On-Line Analytical Processing (OLAP)

On-Line Analytical Processing (OLAP) can be used for essential business applications (including sales and marketing analysis, planning, budgeting, etc.). Unlike typical end-user applications, OLAP products are regularly called upon to process large amounts of periodically refreshed data. OLAP systems need to have the capability to establish persistent links with external sources of data (and dimensions and attributes), such that when changes occur in the external data sources or on a timed basis, they are automatically brought into the multidimensional database. The external data need to follow the input links shown in Figure 3-6 to connect to an OLAP system. Figure 3-7 shows the changes that may affect the data cubes of the OLAP multidimensional system.

OLAP products and applications are generally separate from the systems that input external data and these links also serve as transformation functions. They indicate how to transform table- or spreadsheet-formatted data and metadata into dimensional data and metadata. As yet, no easy methods or tools exist for performing the necessary schema transformations.

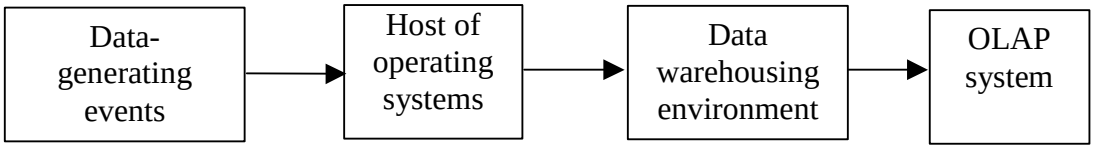


Figure 3-6 Process of external data connect to an OLAP system

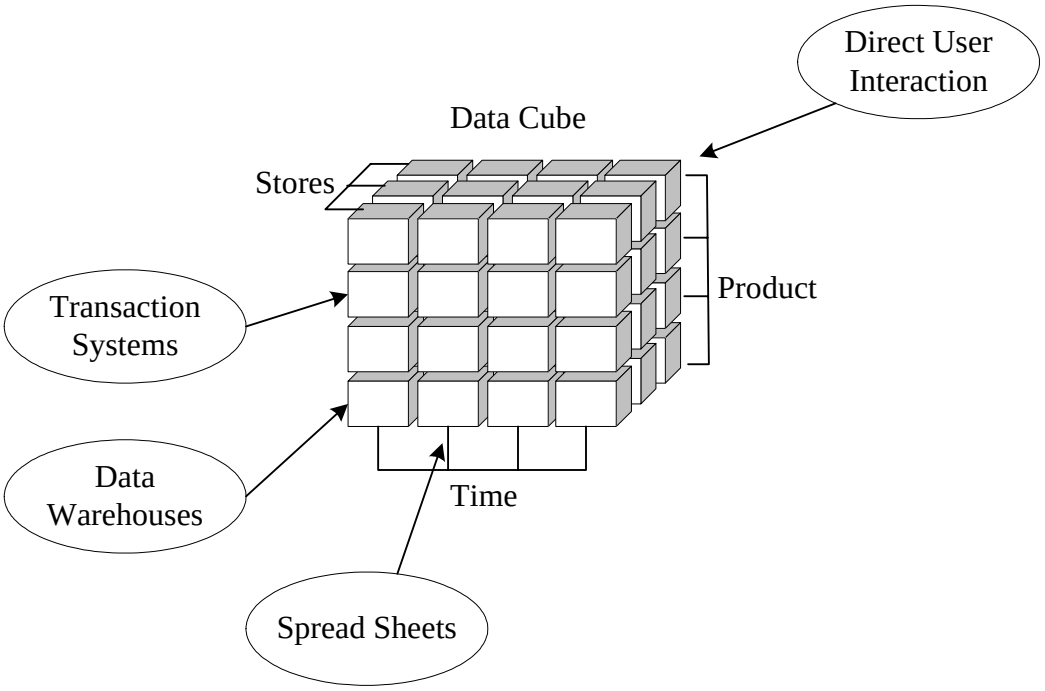


Figure 3-7 Changes may come from many sources (Thomsen 2002)

There are many ways of implementing OLAP compliant applications, and no single piece of technology is officially required. To summarize the OLAP definition in just five key words, they are "***F***ast ***A***nalysis of ***S***hared ***M***ultidimensional ***I***nformation" (or FASMI for short). This definition was first used in early 1995. Fast means that the system is targeted to deliver most responses to users within about five seconds, with the simplest analyses taking no more than one second and very few taking more than 20 seconds. Independent research (Pendse 2004) has shown that end-users assume that a process has failed if results are not received with 30 seconds, and they are apt to restart the system unless the system warns them that the report will take longer. Even if they have been warned that process will take significantly longer, users are likely to get distracted and lose their chain of thought, so the quality of analysis suffers (Pendse 2004). This speed is not easy to achieve with large amounts of data, particularly if on-the-fly and *ad hoc* calculations are required. Vendors resort to a wide variety of techniques to achieve this

goal, including specialized forms of data storage, extensive pre-calculations and specific hardware requirements, but none of these products is yet fully optimized. In particular, the full pre-calculation approach fails with very large, sparse applications as the databases simply get too large, whereas doing everything on-the-fly is much too slow with large databases, even if exotic hardware is used (Kriwaczek 2004; Pendse 2004). Even though it may seem miraculous at first if reports that previously took days now take only minutes, users soon get bored of waiting, and the project will be much less successful than if it had delivered a near instantaneous response, even at the cost of less detailed analysis.

Analysis in the OLAP definition means that the system can cope with any business logic and statistical analysis that is relevant for the application and the user, and can keep it easy enough for the target user. These capabilities differ widely between products, depending on their target markets. Shared in the OLAP definition means that the system implements all the security requirements for confidentiality (possibly down to cell level) and, if multiple write access is needed, concurrent update locking at an appropriate level will occur. This is a major area of weakness in many OLAP products, which tend to assume that all OLAP applications will be read-only, with simplistic security controls.

The system must provide a multidimensional conceptual view of the data, including full support for hierarchies and multiple hierarchies, as this is certainly the most logical way to analyze businesses and organizations. The basic data structure of OLAP can be an array, a fixed record table with hashing, a B-tree, bitmap, or a sorted compression or indexing technique. The two major goals of developers of multidimensional database storage and access routines have been fast access and efficient storage of sparse data. As

a result, most multidimensional databases store data in sets of small, relatively dense arrays, and indexes into the arrays (Thomsen 2002). Then the OLAP applications can be built up based on the multidimensional feature of the databases.

OLAP provides partial solutions to the information-tunnel problem. The reason for a partial solution is because OLAP tools are an expensive investment for any company. For years, companies have been building general-purpose decision support systems with relational databases and ad hoc query/reporting tools. These tools work well for basic data retrieval and frequently provide a number of data formatting and report writing options. But they fall short of meeting many of the requirements organizations place on their decision support systems. Their greatest shortcoming results from the fact that end-users are still directly involved in the SQL definition process and it is difficult for even experienced database professionals to write, and impossible to construct with single-statement relational report writers. OLAP tools provide users with fast access to multidimensional stored data with the analytical sophistication of these tools. But at the expanse of the direct data access functionality of the ad hoc query/report writers of relational databases, OLAP tools don't have an open, scaleable architecture.[1]

In the proposed system, a metadata method model will be implemented, but their will not be XML or wrap applications in the proposed system. XML attempts to address the areas associated with categorizing Web content. For the current design of the proposed system, no Web technology will be used. Even though there are some

[1] In order to combine the advantages of relational databases tools with those of OLAP solutions, people propose Relational OLAP. Relational OLAP depends on the rational data models to access the data. Therefore, data is not stored in the multidimensional model as it is with OLAP. The timing of queries can be significantly larger with ROLAP due to the translation that the ROLAP tools have to perform from the relational model to the multidimensional model that the user is looking for.

similarities between several components of the proposed system and wrap applications, in reality, the system does not plan to wrap the existing data. As will be discussed in more detail in later chapters, a visitor pattern and Navigational Model design will be discussed and compared. The ElementAmbassdor and Navigational Model design will be implemented to infiltrate into each node of the project data tree and to retrieve the needed information. Next in Chapter 4, the methodology used in this research will be discussed. Following that, the system design and implementation will be described.

CHAPTER 4
METHODOLOGY

In the past, the predominant model for data collection has been the scientific method and involved large-scale social surveys. The process includes problem definition, phenomena classification, hypothesis formulation, data collection, hypothesis testing and so forth, and the interaction of theory and method. It is still relevant to a lot of studies. While the quantitative social survey methods are powerful data collection devices, inadequate usage of these survey methods would often result in misusage or the collection of a great deal of data without having a theoretical (or even descriptive) framework to use the data (Wilson 2000). The inefficiency in time and the waste of money are partially caused by the misunderstanding of user research.

Similarly, many companies have strong marketing groups doing quantitative surveys on their customer base. That kind of work is essential, but it cannot raise true innovation because it only shows how the market works at that point in time, not how it could work after introducing something new.

In the construction industry, construction software serves different categories of users. Through quantitative surveys the information needs of users in a certain category can be determined. The problem is that most information researchers generally failed to make an impact within the construction discipline because their research work lacks integration with the theories and workflow within the discipline. Questionnaires can be created and sent out to hundreds and thousands of potential users, however, controlling the quality of the survey results must also be a concern when studying user needs. If

some people, who are not quite familiar with the industry, formulate the survey questionnaires, the questionnaires will not help to find out what the users are really interested in. When the users receive the survey forms, some of them will simply throw the forms away. This is probably the main reason for the low feedback in most surveys. Usually different people have different needs for a particular item. It is difficult to list all the user needs and tailor them into several pages on the survey form. In addition, a lengthy survey form may exhaust the patience of the respondents.

Besides quantitative survey methods, the author suggests that user research can also use qualitative methods. The qualitative research proposed here is an alternate way in finding out user preferences and needs. One of the methods in the qualitative research is through observation. In order to scrutinize the daily information needs of a project manager in the construction industry, the author accepted a job in a construction company and experienced the role of project manager on real projects. At the same time, the author contacted other project managers and observed their work methods. All the experiences and observations are analyzed in Chapters 5 and 6. This research also utilizes the quantitative survey results from several publications. By combining the qualitative and quantitative methods, this research will show that the proposed user model and ***EX***tensible ***T***ask ***E***lement ***T***ree (EXTET) as part of the User Model Driven Architecture is feasible and represents the user needs in the design of software. It is a user centered design and will serve user needs better when used in project management on real construction projects. It will help in configuring the GUI and system's application layer for information retrieval in construction project management. The detailed steps involved in this research are as follows:

1. Data collection: In the work described in the following Chapters, observations and interviews were the main approaches used.

2. Data feedback and Discussion: The feedback may be done by written report or by presentations at meetings. The Construction IT Group in the Rinker School of Building Construction at the University of Florida has a routine meeting every two weeks. The group discusses the observations of the members, makes presentations, and comments on the reports from those in attendance. The group discussion environment and the sincere help from the participants and friends partially compose the data feedback part of this research.

3. Training Programs: Most members of the Construction IT Group have obtained or still in training in computer and construction courses. They could be eligible users of the proposed system. To validate the system, some users who have experience gained on real construction projects and who were potential users of the proposed system were invited to help test the system. As described in Chapter 9, the users were offered a short training class to familiarize them with the system.

4. Implementation and monitoring: For the proposed system, the Construction IT Group will perform this process. It will be another phase of data collection.

5. Evaluation and Feedback: It involves the assessment of all data collected during the monitoring process (both quantitative and qualitative).

6. Problem re-definition: This step begins a new round of research based on the work did before. In the Conclusion part of this study, the author discusses possible future research topics.

This study will include analysis of third-party software and hardware vendor catalogues, various published literature (papers, articles, and books) related to modeling, and personal contacts (such as engineering and construction company software users and developers). Solving a practical problem offers the possibility of testing social/behavioral science models and theories in the “real world”. Given sufficient case studies involving similar situations, an appreciation of the congruence between theory and reality can be achieved. The process of testing theories in the real world will lead to modifications of those theories. The process will result in the formulation of hypotheses, which require empirical, quantitative testing.

Chapter 5 discusses some use cases and the conceptual design of the proposed system architecture. Chapter 5 includes 3 detailed case studies and based on these case studies, three observations are made which will help in building up the system functions and user models

CHAPTER 5
CASE STUDY

5.1 About the Company

In order to analyze the user activities and expectations in construction management, let us first build an imaginary construction company. The name of the general contractor is called Dreamland Construction, and its main business includes industrial, residential, commercial, and heavy civil construction. There are 37 people working in this company including one president, four (4) vice presidents, and seven (7) office workers, who support the daily office process, such as sending and receiving faxes, letters, and packages; answering telephone calls; assisting construction and other document management; book-keeping of the construction records and daily project field reports; and accounting. Ten (10) project managers and fifteen (15) superintendents also work for this company.

Dreamland Construction values the relationships with its customers, architects, and the sub-contractors, which is very important in competing with other general contractors. For instance, during construction processes, a lot of useful advise and expert suggestions come from subcontractors. A lot of General Contractors (GC's) agree that if two bid proposals from different subcontractors are quite close or within a 5% difference, GC's will usually use the subcontractor that they used before if they had done a good job for them. Because construction project usually involves a lot of uncertain factors, such as the underground conditions and unforeseeable weather changes, the GC has to negotiate with them how these unforeseeable changes will be handled. If the GC knows that the

subcontractor well, the GC will feel confident that the subcontractor can perform a good job judging from their previous cooperation. When working with unfamiliar subcontractors, GC's have to pay extra attention to them and spend time developing a relationship with them.

Suppose there are 3 people working on a construction project. The vice president of the company is managing the business of the project and is responsible for contact with owner. The Project Manager (PM) is responsible for detailed project management and is the decision-making person working on the cost control of the project. The project Superintendent is in charge of day-by-day jobsite work and coordination of all the works of subcontractors. This research is focused on the activities of the PM and Superintendent.

5.2 Description of the Project

The project is located in a city in southwest Florida. Hurricane season there is from June till November each year. FEMA (Federal Emergency Management Agency) requires that building in that area is able to resist winds of certain speeds. In addition, the SWFWMD (Southwest Florida Water Management District) has regulations that the project should be able to retain the first ½" rainfall inside the project property. For example, if the water depth of the rainfall is 3", the first ½" should be able to drain into the sump or pond inside the intended project property area, and the rest 2-1/2" can be drained into a drainage pipe and is allowed to flow into a city or county drainage system. The SWFWMD needs about 6 months to review a project. There is another organization that may impact the project, the local Development Review Committee (DRC). Commercial projects with foot print areas equal to or more than 6,000 SF, are subject review and approval by the DRC. The function of the DRC is to evaluate the influence of

the project on the surrounding areas from the aspects of economic, political, future development, transportation, and community. There are other impact fees for buildings, utilities, roads, sewers, water impact fees, and permits (i.e. building, trailer, storm water, FDOT (Florida Department of Transportation), driveway, etc). It is suggested that all these fees and permissions should be applied for and paid directly by the owner, instead of going through the General Contractor. By doing that, the estimated building price will be lowered; on the other hand, it is simply a payment process. If the owner pays the fees directly to these organizations, then the owner can save the contractor's markup.

For the aforementioned project, there was a preliminary set of drawings and specifications. At that time, in order to give the owner some rough idea of the cost of the project, the GC asked proposals from a few subcontractors and estimated the job partially according to the preliminary set of drawings and specs. The preliminary design left out the landscaping, detailed cabinets, and equipments drawings. So these works were taken off and estimated according to some other similar projects. After approximately 4 month, the owner and the architect finalized the project and the architect provided the final drawings. As a result, the subs need to do the estimating again.

The project is called the James Center and its floor plan is shown in Figure 5-1. The designed foot print area of this building is 9,000 Square Feet (SF). The main part of the building has CMU (concrete masonry unit) walls. The interior partition walls are metal studs with drywalls on both sides. The roof trusses of this building are metal roof trusses with a 5 / 12 slope. The manufacturer of this truss system is Florida Lumber, Inc. The supporting framing of this truss system is on 7 units of double extra strong steel pipe

columns and steel beams. Concrete masonry bonding beams sit on top of the masonry walls.

The James Center project has rigid insulation boards on the exterior walls and batt insulation in interior walls. The insulation for the metal roof is Poly-wrapped roof insulation and has been included in the steel building roof trusses. Most interior doors are 6-panel wood doors. The metal frames of the interior doors are installed by the subcontractor that does the metal studs and drywall for partition walls. The wood doors and the finish hardware are provided and installed by door subcontractor. The exterior doors are made of metal and have metal frames. The metal frames of the exterior doors are provided by the door subcontractor and are installed by the masonry subcontractor. The exterior-door slabs are provided and installed by the door subcontractor. The main entrance door is a glass impact door. The glass door and the window glazing is provided and installed by a glass door and window subcontractor. The masonry subcontractor installs all the wood backing for the door and window headers and the jambs in the exterior walls. The drywall subcontractor installs all the wood backing parts for the door and the window frames in interior walls, the grab bars in bathrooms, cabinets, doorstops, and counters.

The exterior wall finish for the James Center project is stucco and the Owner will select the colors. The interior wall is furred with gypsum wallboard, with light knockdown finish and will be painted. The bathrooms have ceramic tile floors. The kitchen has resilient flooring (VCT or Vinyl Composition Tile). All other areas are covered with carpet. The ceiling has acoustical ceiling tiles. The windows have marble sills as decoration. A floor protection allowance was included in the estimate. All the

specialties including bathroom accessories, fire extinguishers, mirrors, interior and exterior signs will be the responsibilities of different subcontractors. The James Center project has completed plumbing, HVAC (Heating, Ventilating, and Air-Condition), and electrical systems. Equipment (such as kitchen equipments) and furnishings (furniture, drapes & blinds, etc.) will be chosen, purchased, and installed by the Owner.

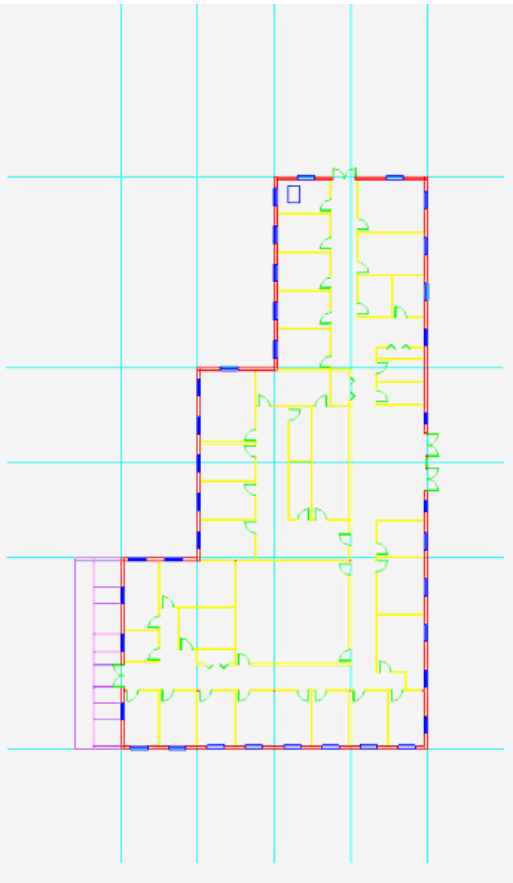


Figure 5-1 Floor Plan of the James Center

5.3 Case Studies

The following case studies address the information needs of two different user groups: Superintendents and Project Managers. The case studies are composed of three cases including information retrieval activities for material handling and management, the Request For Information (RFI) process, and the maintenance of a punch list. All the

tasks relate to project management and information-seeking activities, which means the superintendent and/or project manager need to collect information from the construction jobsite and also search for related information from the companies' database system(s). Under the Analysis of Information Connectivity section, we identified eight databases (Scheduling, Drawings, Specifications, Estimating, Contracts, Purchase Order, Productivity, and Storage Information databases) that are required for construction project management activities. To simplify the complexity of the analysis in this Chapter, the databases referenced here are the CAD Drawings Database, the Scheduling Database, the Estimating Database, and the Contracts and Purchase Order Database. The case studies also include three different construction projects, which vary in size, types, and nature. In order to illustrate the complexity of the information needs, the following scenarios will be considered:

- At the same time, for the same project, with the same question, different users have different expectations and requirements for their search results.
- With large volumes of information and multiple integrated databases, providing different users with the same list of all possible answers (relevant or non-relevant) is not a very plausible idea.
- With regards to the customization of the data access and data maintenance, a static content filter mechanism will not be able to adapt to the dynamic feature of the users' information retrieval.

The first two scenarios are self-explanatory. In the third scenario, the static content filter refers to a filter which is system environment independent or user independent. Once the rules of the content filter mechanism have been set, they will not change when users change or when the users' tasks change. So the static content filter mechanism does not have the ability to adapt to the change of users and the change of user tasks. Thus far, content filters have been used in the following areas (SphinxSoftware 2004):

- Vandal / Virus Protection, which protects against viruses, vandals, worms, Trojans and other types of mal-software;
- Application Filter: can block the unauthorized application traffic (such as P2P, IM, Spyware, etc.);
- Anti Hacking Tool: blocks exploitation of known security holes by hackers and malicious code;
- Web Filtering: filters web access;
- Anti-spam Module;
- Secure Email: protects email traffic against email security threats.

There are many separate applications providing content security. A majority of these applications guard people separately, separate programs for ad blocking, for spam suppression, for privacy and so on. All of them process the Internet content from their own, separate point of view. The proposed system will be able to reconfigure the environment and customize the data access and system response. One difference of the proposed system from the static content filter mechanism is that the proposed system employs user models to reconfigure the system environment; and the system is extensible. So using the proposed system allows the user model to address sophisticated searching for information from many sources for multiple classes of users.

In developing these case studies, the author worked with several project managers and superintendents on multiple projects, experienced the complete project management process, contacted and met with many construction people from other companies, and performed multiple informal interviews to find out their information needs. All these experiences are referred to in the rest of this study. These case studies reflect the daily work of several superintendents and project managers.

5.3.1 Case Study 1: Material Handling and Management

The James Center project will be used for this Case Study. The case study was conducted on a purchase order process to reflect the Material Handling Management function for this 5-storey building. The designed foot print area of this building is 9,000 Square Feet (SF) with an estimated cost of USD 12 million.

After the hanging of drywall, the Project Manager and the Superintendent need to determine the delivery date of doors and hardware. To simplify the analysis, only use the information regarding one type of door, Door #2, will be used as an example. When the Project Manager asks the system: "Door #2 Delivery", what s/he wants to retrieve are four things: (1) the CAD Drawing showing the location of the Door #2, (2) the Schedule showing the starting date of installation of Door #2, (3) the information from the Contracts Database showing who will furnish, deliver, and install Door #2[2], and (4) information from the Estimating Database showing the cost of labor and material. The query of the PM and the expected results of that query are shown in Figure 5-2.

When the Superintendent submits the query: "Door #2 Delivery", what he/she wants to find out are two things: (1) the Quantity Take-off of Door #2 in order to determine how many Door #2 he/she will receive on the job site, and (2) the Schedule so that s/he knows when the doors will arrive. The Superintendent's query and the expected results of that query are shown in Figure 5-3. By comparing the expected results for the same query from different user groups, the PM and the Superintendent, it will be shown that at the same time, for the same project, with the same question, different users have different expectations and requirements for their search results. If the expected results

[2] Other information (such as the confirmed delivery date, etc.) is stored in other databases. To simplify the analysis, we leave that information unmentioned.

from the estimate database in Figures 5-2 and 5-3 are compared, it will be determined that the Manager was paying more attention to the cost control and financial management, so that the lump-sum cost of doors, frames and hardware will fit his requirement. A Superintendent on the other hand needs to make sure the right materials and quantities are delivered to the job site. Accordingly, the superintendent needs to know the detailed description of the doors, frames and associated hardware and their quantities. Another

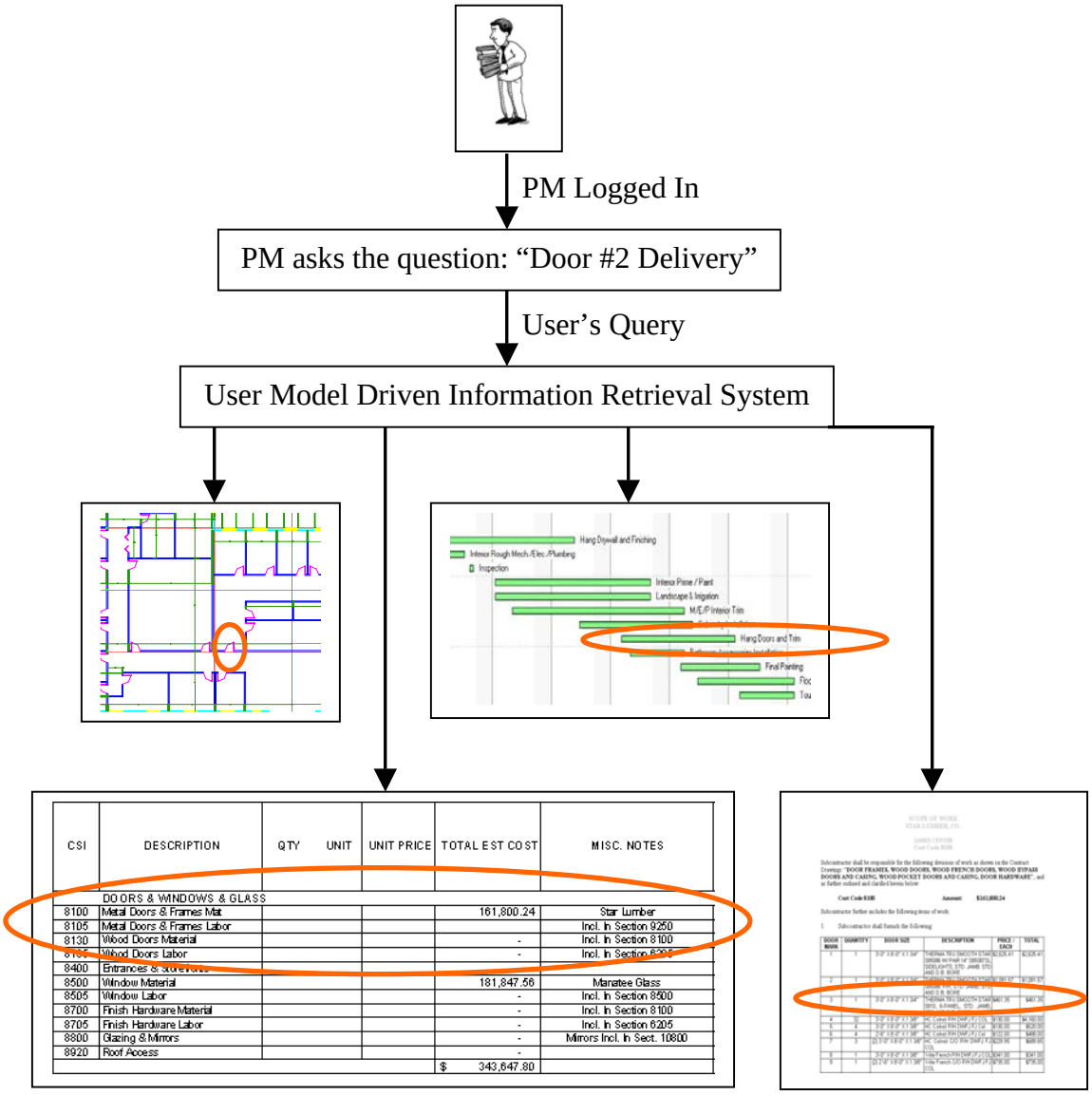


Figure 5-2 Requirements analysis for Project Manager (PM)

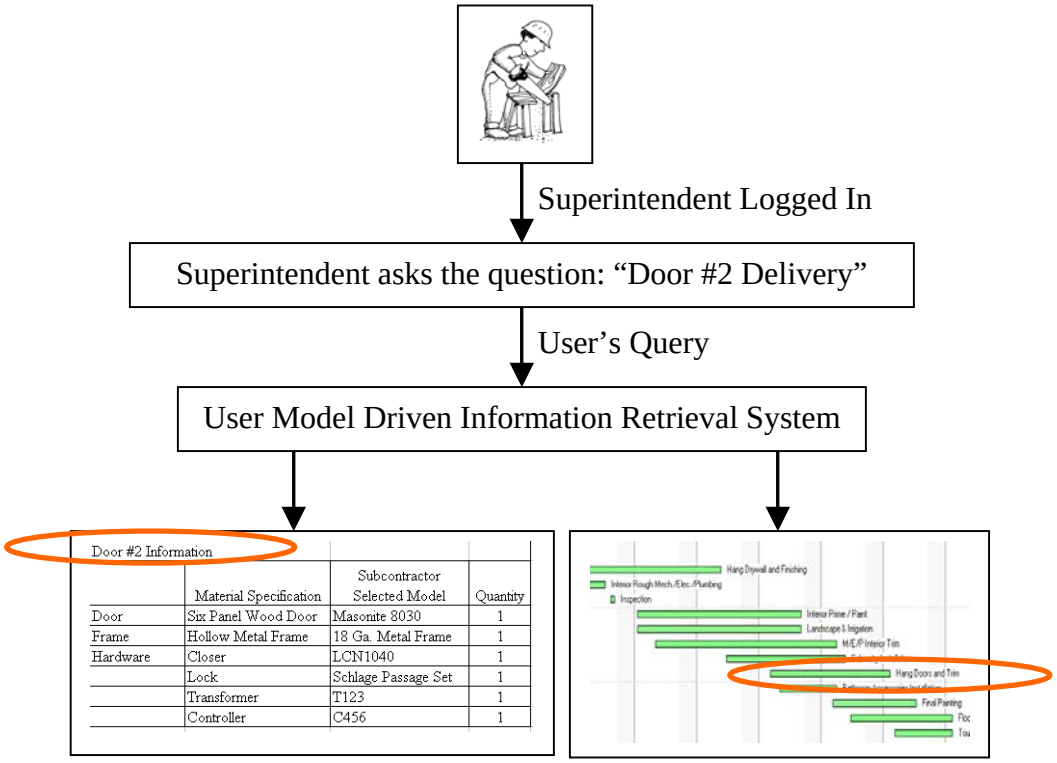


Figure 5-3 Requirements analysis for Superintendent

observation from comparing Figures 5-2 and 5-3 in this case study is that if a user’s query is answered with a list of all possible answers shown on the screen, the search results presented to the PM (as shown in Figure 5-2) and the Superintendent (as shown in Figure 5-3) would be similar. The Superintendent does not need a lot of the information found in the query results shown in Figure 5-2. This may cause confusion and a waste of time for the user.

5.3.2 Case Study 2: the Request For Information (RFI) Process

The second case study concerns a condominium construction in Miami, FL. It is a 44-story residential tower. The project is a combination of 448 condominiums and 13,000 sq. ft. of office space on a 1.5-acre site. The estimated construction cost is $130 million,

with a construction duration of 17 months. The residential part of the building consists of 1-bedroom, 2-bedroom, and 3-bedroom units.

The Request For Information (RFI) process is typically used before or during the construction of a project. The construction documents (agreement, drawings and specifications, calculations, etc.) are generated by the Engineer or Architect and document the conditions agreed upon between the Owner and the Contractor. These documents reflect the understanding that each party has with regard to the intended construction of the project, but not every set of Construction Documents is clear, unambiguous, and complete. The construction documents will not adequately address every single matter that might happen during construction. There may be gaps, conflicts, or subtle ambiguities with site conditions, current construction techniques, or even the construction abilities of the Contractor. When this happens, the Contractor has to consult with the Engineer or Architect to clarify the confusions. The goal of the RFI is to act as a partnering tool to resolve these and to eliminate the need for costly corrective measures.

For the condominium project, one superintendent was assigned to supervise the construction site. An assistant superintendent is scheduled to start helping her/him in the final 6 months of the project when a lot of subcontractors and trades are working on the job site. Table 5-1 shows the formal format of the RFI process. As shown in Table 5-1: after the Concrete and Elevator Subcontractors were awarded the contracts, they started bidding out some parts of the project (i.e. digging footings), ordering material (ready mixed concrete, masonry units, etc.), and preparing Material Safety Data Sheets (MSDS) and shop drawings (material data submittals, rebar shop drawings, etc.). After reviewing the contract drawings and the submittal data and shop drawings from the Elevator

Table 5-1 Formal use case for RFI process users

Request For Information
Primary Actor: Project Manager
Goal in Context: Project Manager requests information from Architect or Engineer and informs Superintendent and Subcontractors
Scope: Business – The overall Contract Documents Changing Process as seen by the people in the company.
Level: Summary
Stakeholders and Interests Project Manager: Wants to resolve questions and clarify confusions he/she had and find out if there is any change in time and/or cost. Superintendent: Wants to resolve questions and clarify confusions he/she had. Subcontractor: Wants to do right work.
Precondition: None
Minimal Guarantees: Every RFI sent out has been approved by a valid authorizer – PM.
Trigger: Project Manager wants to find out answers because (a) Substitution/Construction Modification, (b) Clarification or Additional Information, (c) Construction Deficiency (During Construction Phase), or (d) Construction Document Deficiency exists.
Main Success Scenario:
1. Superintendent / Project Manager / Subcontractor (Requestor): initiate a question request.
2. Project Manager (Checker): Check the construction deficiency or Contract document clarification.
3. Project Manager (Sender): Send the question to Architect.
4. Architect / Engineer (Approver): Research on the RFI and return answer to Sender.
5. Project Manager (Receiver1): Validate Approver's signature. Send the answer to Superintendent and all the influenced Subcontractor(s).
6. Superintendent (Receiver2): Check the status of the requested item on job site and verify if all the influenced Subcontractor(s) know(s) about the RFI answer.
7. Subcontractor(s) (Receiver3): Receives the answer to RFI from Receiver1 and responds to Receiver1 if there is any change in construction schedule and cost.
Extensions
• 1a. During the Bidding Phase, a RFI for a substitution or a Construction Modification should be sent to the Architect in writing at least 10 days prior to bid opening.
• 1b. During the Construction Phase, a RFI shall be sent to the Architect immediately.
• 2a. Project Manager shall identify the substitution noting ***the specification or plan sheet involved*** and the reason for the substitution and/or correction of the deficiency. This would include ***schedule impacts*** and ***costs***.
• 2b. Suggestion shall identify the construction modification and/or procedure for correcting the problem, using the detail and plan sheet location.

Table 5-1. Continued

• 3a. If Checker finds out that the Architect takes too long to answer the question, the PM will send a second request about the RFI.
• 4a. Approver reviews the RFI for completeness and return to Sender if it is not complete
• 4b. Approver reviews the RFI request and responds with the appropriate clarification, additional information.
• 4c. Approver reviews the RFI and verifies of the deficiency and responds with the proposed corrections to Sender.
• 5a. Receiver1 ***checks with Subcontractors*** if the clarification or additions result in an impact on schedule and project costs. If that is the case, the Receiver1 will respond to the Approver within 2 working days. No work is authorized until the Approver agreed on the schedule and/or cost change.
• 6a. If Receiver2 finds out the answer to RFI is not appropriate or not fit the site condition, Receiver2 responds to Receiver1 immediately.
• 6b. Receiver1 starts another process of RFI.
• 7a. If Receiver3 finds out no impact on time or cost, end the RFI process.
• 7b. If Receiver3 finds any influence on time or cost, Receiver3 reports to Receiver1 and Receiver1 starts the Change Order process.
Technology and Data Variations List: Estimate and Schedule
Priority: Response ASAP
Release: Release to Subcontractor(s) immediately
Response Time: 1. Project Manager Reports To Architect Time should be less than 2 working days. 2. Architect Response Time should be less than 5 working days. 3. Project Manager Informs Superintendent and Subcontractor(s) Time should be less than 1 working day.
Channel to Primary Actor (PM): Fax, UPS (or any other express mail), hand delivered paper work.
Channels to Secondary Actors: Fax, UPS (or any other express mail), hand delivered paper work.
Open Issues: None

Subcontractor, the Project Manager found a discrepancy between the information provided by the elevator manufacturer and the specifications on the drawings. The Project Manager sent a fax to the Concrete Subcontractor to hold the corresponding work in case the Architect selects a different elevator model. If that is the case, the footing size and reinforcing will all be changed. At that time, site preparation and soil treatment were

just completed. The next activity is “Excavating the Underground Parking Floors”. There are four stories of underground parking for the condominium residents or tenants and the office workers.

The PM, Superintendent, and the Subcontractor may conduct a site visit together to reach agreement on the problem, and to help the PM describe the problem and explain their suggestions to the Engineer. To accurately describe the problem and give suggestion, the PM will need to visit the problem area on the job site and the surrounding areas with the Construction Drawings. The PM will need to mark out the area and uses a note pad or recorder to describe the problem in detail. The Architect will be responsible to contact the structural engineer to answer this question. When the Engineer and the Architect agree on the solution to the problem and send the answer to the PM, the solution will become a legal part of the Construction Documents. When the PM informs the Superintendent about the answer to the question, the Superintendent needs to make sure that s/he understands the answer and needs to find out which subcontractors are impacted. So when the Superintendent submits the query to the system: “RFI Elevator”, the two databases the Superintendent is interested in are CAD Drawings and Contracts and PO databases[3]. The former database will help him/her understand the changes and latter one will help him/her make sure all the affected subcontractors are working together to solve the problem.

Besides these two databases, the PM will also need to search the Estimating and Scheduling databases to find out how this change will affect the construction cost and

[3] Please note that the databases referenced in this chapter are composed of CAD Drawings Database, Scheduling Database, Estimating Database, and Contracts and Purchase Order Database. Under the practical situation of Case Study 2, the information from RFI and RFI Log Database should definitely be presented to users. To simplify the analysis, it is assumed that the users know about the RFI and RFI Log. i

project schedule. Figure 5-4 shows the answers the Superintendent is looking for: the Drawings and the Contract and PO databases. Figure 5-5 shows the user needs of the Project Manager when s/he submits the query “RFI Elevator”. It includes the 2D layout of the problem area of the construction drawings, the Contract and PO database, the elevator cost, and the schedule. If we compare this Case Study with Case Study 1, we will find that the expectation of the Superintendent in this case is different from that of Case Study 1. If we use a static user preference filter, the results will not match what the users need. Another observation from the comparison is that under different tasks, the same user will have different requirements for the databases. Hence, if we want to build the user-model-driven system architecture for information retrieval, the user model must include the user profiles and the task profiles.

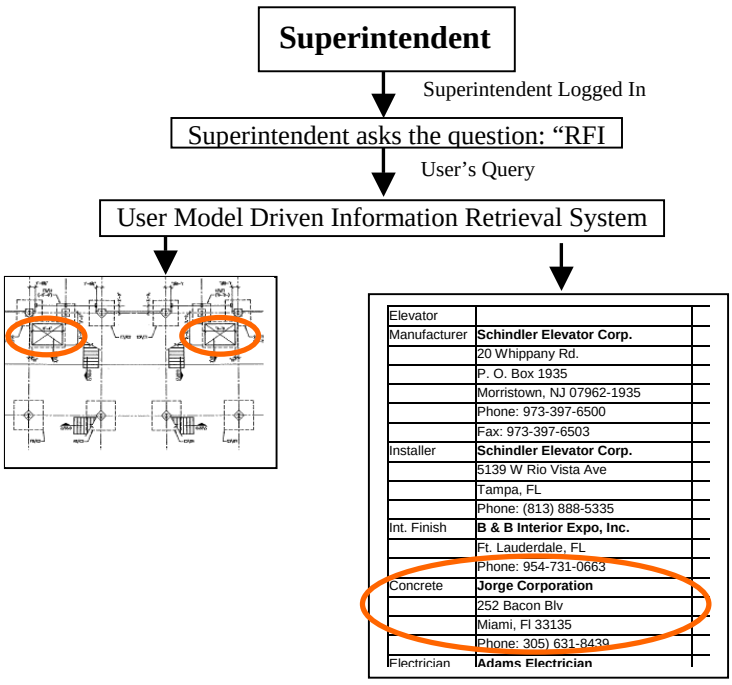


Figure 5-4 Requirements analysis for RFI query by the Superintendent

5.3.3 Case Study 3: Maintenance of the Punch List

Reinhardt (2003) discussed case studies on creation and maintenance of Punch Lists. The case studies he performed were mainly focused on the data collection tasks for

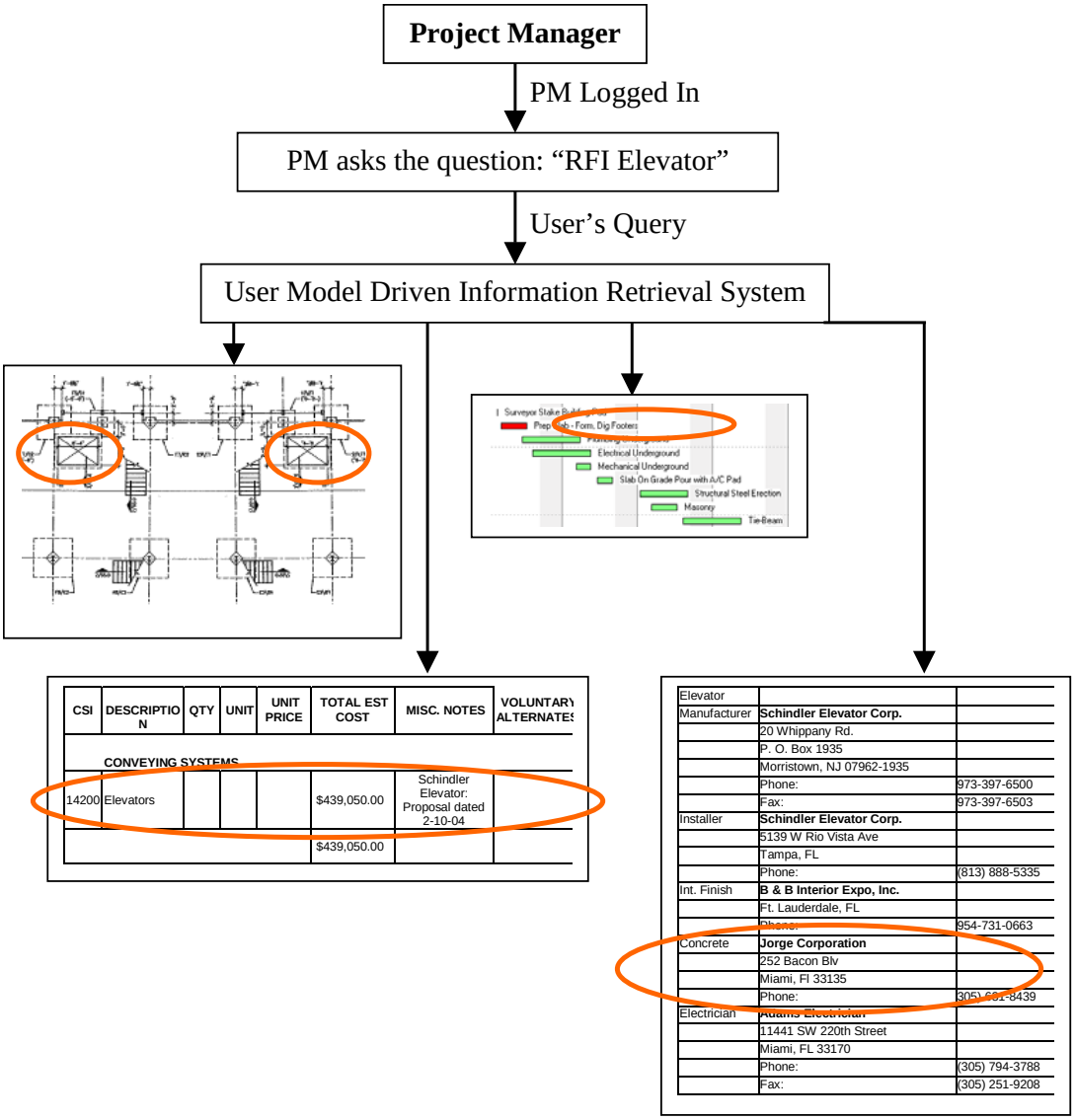


Figure 5-5 Requirements analysis for RFI query by PM

monitoring progress and for the creation and maintenance of punch lists. The goal was to find out what information the Project Manager and Superintendent needed to retrieve from the system to maintain the punch list. The study is based on an industrial plant

located in North Canton, Ohio. The building size is 300,000 sq. ft. with 36-acre site work. The ceiling height is about 28 feet, with a crane available. The expected construction duration is about 7 months including civil work. The focus of this study was the field punch list inspection and maintenance process and the related information tracking procedure.

As is the case with most of the industrial buildings, this industrial plant has structural steel and masonry as the main structural system of the building. The erection of the steel structure and the masonry work will last 3 months. Even though the structural inspection has been performed by inspectors and passed, some unforeseeable factors might affect the building structure during the construction after the building shell has been set up. For example, the installation of some equipment might influence the structure, or some workers might damage the structure when they are doing their own scope of work.

The weekly assessment on the job site for the current project situation is carried out by the Superintendent and sometimes with the help from the Project Manager. When they marked out on the drawings the current status of the punch list and update it, they need to first find out the punch list data. Let us assume the punch list data were stored in the CAD drawings database. Each time the Superintendent maintains the punch list, s/he needs to ask the system: “punch list”. The expected results should include the following information:

- The CAD drawings with the location of the punch list items. ← This information is stored in CAD drawings database
- The names of the subcontractors who should be responsible for the punch list items. ← This information is stored in Contract and PO database.

The information on the CAD drawings of this query by the Superintendent shall be specified every item with the location, the status, and the description.

For the Project Manager, s/he also needs to have information from the above two databases for the query of “punch list”. But the information on the CAD drawings does not need to be as precise and distinct as that for the Superintendent. The PM only needs to have a text-based description of the punch list and the names of the subcontractors who should perform the punch list. In this case, the Superintendent needs more details in the resulting information than the PM does. Thus when different users retrieve information from the same database, they might need different levels of information detail.

5.4 Bidding Process Case Study

The previous case studies are focused on the users’ activities in the construction process. The case studies analyzed and compared two groups of users: Project Managers and Superintendents. For the project bidding process, the Superintendent does not have much need for information retrieval. Instead, the PM and estimator are the users who need to collect as much information as they can to generate the estimate and proposal for bidding on the project. The following content studies the information needs of users during the project bidding process.

In May 2002, the President of Dreamland Construction, Inc. met with the Owner representative to discuss a new office building. Three months later, in August 2002, the Owner asked proposals from four (4) architecture companies for the project design and hired the one with the lowest price and that also satisfied the Owner’s requirements and expectations. The project is initiated by a written Program Narrative, which defines the mission, the function and details of the project. This Narrative is then developed into a "Program" which is mutually established and approved by the design professionals to

establish detailed requirements and is used to establish the planning budget. This narrative serves as the baseline criteria for the balance of the project. Based upon the descriptions established in the Program Narrative, the architect prepares a preliminary design of the project for review by Owner and GC. In September 2002, the Owner started to invite general contractors to bid on this project. On September 8, 2002, the president of Dreamland Construction received the Invitation-To-Bid letter from the owner and a notice from the owner saying that there might be some changes on the drawings later. Then, the Architect and Engineers worked with the Owner to fine tune the design and incorporate structural and engineered details for the building systems as well as the end user specific systems. Interior design work also begins developing at this stage. The PM of Dreamland Construction began to invite subcontractors in the preliminary design stage and submitted a preliminary price proposal. After the final set of drawings was ready, the PM would conduct a second round of bidding and generate a more accurate estimate.

At the time, there were plentiful ongoing construction projects managed by Dreamland Construction, which raised some concern about attracting skilled subcontractor labors. As early as the bidding stage, Dreamland Construction demonstrated innovative thinking by submitting value engineering suggestions and a list of 7 associates who had worked on commercial projects in the south Florida areas from 1995 to 2002 with good reputations. All of them were available for the project.

After deciding on which subcontractors they are going to use, the general contractor has the final estimate almost completed. The next thing that the PM will do is to add in the contingency, contractor overhead, contractor profit, construction management fee, and bond premium if applicable. Then the PM will prepare the bid

package (which will include the estimate spreadsheet, the schedule bar chart, the subcontractor proposals, certain required submittals, etc.).

The bid for the James Center is in a competitive format. Site presence cost was estimated at a monthly rate, and the direct labor and equipment were estimated at hourly rates. Bid documents required a detailed budget for the project. To accomplish the task, Dreamland Construction painstakingly created staffing, labor and equipment plans for the project and the site establishment. Data were provided in hard copy and electronic format so that Owner could thoroughly evaluate Dreamland Construction's proposal against other competitors.

Before awarding the contract, Dreamland Construction prepared a presentation with incentive plans to the owner. Dreamland Construction showed that their personnel had developed a cost-savings estimate that would result in shared savings with the Owner. The estimate was structured to contribute the cost savings to the owner according to a predetermined percentage. After being awarded the contract, the PM will start writing subcontracts and negotiate with subcontractors regarding their scopes of work. Prior to construction, all managers, supervisors and staff who would work on the jobsite were invited to a customized project seminar. The purpose of the seminar was to teach behavior that would result in an accident-free, injury-free jobsite.

5.5 Suggestion for Automating the Bidding Process

The information needed at the bidding stage for a construction project is voluminous. The Project Manager needs to be familiar with the project. The PM should know what type of project it is, a commercial building, a residential building, or industrial project. It is the author’s opinion, that the computer system in a construction company should help people to arrive to right decision but should not try to replace

human interaction or replace humans in the decision-making process. The human decision process is a very complicated one. It includes many (i.e. social, economical, and psychological) decision rules. To make decision, people will first cognate the situation and then try to use the rules and make decision based on their experience and information. For a computer system, between the two steps of the human decision-making process, there should be a notational system to change the cognation result into a computer understandable notation. Blackwell and Green (2003) described concisely the difficulty in writing a simple computer program, "Correctly dictating even a short program straight from the head is nearly impossible, and the editing commands provided by this system made the problem even harder because they required the program to remain syntactically valid at all times" (p. 105-108).

Black and Green (2003) suggested using the following relatively well-defined series of steps to design a system when applying cognitive-dimensions analysis:

- Get to know the system.
- Decide what the user will be doing with the notation
- Choose some representative tasks.
- For each step in each task, ask whether the user can choose where to start, how a mistake will be corrected, what if there are second thoughts, what abstractions are being used, and so on for the other dimensions. This will generate an observed profile.
- Compare the observed profile with the ideal profile for that type of activity (p. 105-110).

As mentioned before, the desired system design is a system that can integrate the necessary information about the project together quickly and provide the requested information to people in the way they need. Thus the system is designed to satisfy the users' needs and will provide information to users according to their work requirements.

In practice, a lot of users of a certain system may have already adapted to a notation system. Take the proposed system as an example. There are many construction terminologies that the users feel familiar with, such as “bid”, “take-off”, “sub”, and “fur”. This increases the difficulty of designing the proposed system.

In the observed activities of the James Center PM, the representative tasks were chosen and described by the PM’s behavior in the bidding process. In order to bid (tender or prepare) for the James Center project, the PM should be able to do the work of quantity calculations, estimates, and project management.

Quantity calculation is involved at the bidding (tender) Stage and the Construction Stage. During the bidding Stage, the PM is involved with assessing the risk of undertaking the job that the contractor is bidding for. During the Construction Stage, the PM's quantity calculation is responsible for measuring the quantity of work done and preparing a bill, which is submitted to the client for payment to be made to the contractor. The quantity calculation also can be used to evaluate any costs that are incurred regarding changes in the original design.

The estimating work includes producing cost estimates for the work to be carried out upon successful submission. This estimate would form the basis of the entire contract. The project manager is the top man overseeing daily operations on site. During the Preparation Stage, the contractor appoints the project manager to be in full control. The PM’s job includes supervising and organizing daily site activities, coordinating with various parties and controlling progress of the project, and making sure that the schedule is kept. To work effectively as a PM, the following attributes would be useful: aptitude for figures, knowledgeable about construction materials, good communications skills,

good public relations and organizational skills, and enjoy problem solving, most importantly, have an eye for detail.

A good PM should be able to remember a lot of things. But as Anderson and Milson (1989) found, there are particular mathematical functions that characterize forgetting from human memory. Memories tend to decay very rapidly if they are not used frequently or have not been used recently. The specific mathematical form of this decay function was shown to be optimal given the likelihood of information recurring in the environment and the assumption that the cost of searching memory increases with the number of items stored. So no matter how well a PM can remember things, he or she still needs something to record and document information. For example, the PM needs a name list, telephone book, or card book to record all the subcontractors. As Simon (1992) put it, in order to efficiently allocate attention to information, the amount of relevant information encountered by a user needs to be increased as a function of the amount of time that the user invests in interacting with the system. If a user can attend to more information per unit time, then the user's information processing capacity is increased, thereby amplifying cognition. The structure of the physical and virtual worlds determines the time costs, resource costs, and opportunity costs associated with exploring and finding information (p. 150-161).

Chapter 6 discusses the user requirements and the user interfaces. This chapter will also analyze the different types of information stored in a project database.

CHAPTER 6
SYSTEM REQUIREMENTS AND USER INTERFACE

In Chapter 5, a case study about a construction company on a light commercial project was presented. In addition to the material handling and management in Cast Study 1, case studies on the RFI process and on the maintenance of punch list were also presented. The case studies showed that at the same time, for the same project, with the same question, different users have different expectations and requirements for their search results. In addition, not all of the construction users need to read the list of all possible answers. With regards to the customization of the data access and data maintenance, a static content filter will not be able to adapt to dynamic user needs. To find out the method to adapt to dynamic user needs, a configurable system needs to be built. Included in this Chapter is the analysis of the different types of information stored in a project database. The project database will be the data source for the user queries. Due to the existence of different types of construction projects, the types of data stored in the project database will be different and the project database is required to adapt to changes.

System integration is one of the developing trends in the current construction Information Technology Research (FIATECH 2004). The more databases the system can integrate, the more information it can provide to users. The intention of designing integrated systems is to save user's time and effort in going through all the different data sources (i.e. databases or paper works). If the system can only answer a user query by sending back all the related information, the user will still need to spend a lot of time to

dig out the useful information s/he wants from the long list of results. Based on the analysis of the project database and the use cases, the requirements for the UMDA (User Model Driven Architecture) system architecture will be established, followed by the user interfaces for the UMDA system. In illustrating the user interfaces, some of the system functions will be discussed, and these functions are used to fulfill the system requirements.

6.1 Analysis of Data Stored in the Proposed System

Information is considered to be all of the data, both original and derived data. There are many considerations regarding information in the computer IT field, for example, data duplication, RAM required, disk space utilization, performance, and integration with data warehouses. The proposed work in this study is focused on the construction information management and how to ease the work of retrieving the needed information.

Figure 3-1 in Chapter 3 shows the relationship among the databases in construction management. It also shows the types of data stored in these databases have. It is a simplified version of the data that will be needed in the design/build process of the construction project. The databases shown in Figure 3-1 are the Scheduling Database, the Estimating Database, the Purchase Order Database, the Drawing Database, the Productivity Database, the Specifications Database, the Contracts Database, and the Storage Information Database. In reality much more information will be involved in this process, for example, the Subcontractor and Supplier Database, the Building Code and Local Regulations Database, and the Local Weather Database. All these data have one overarching theme: THE PROJECT.

As explained in the previous chapters, database developers should not expect the end users to be experienced computer professionals, although some may be very skillful

in using databases or other computer systems. Remember how many times people in construction companies wrestle with Microsoft Excel to make a correct and crisp-looking estimate spreadsheet for a project? Even if they have used Excel for years and most of Excel's functions are self-explanatory, the users still can face problems in building up the spreadsheets they want. As to DBMS or information retrieval systems, they require more knowledge in computer science. Construction personnel need to do a lot of information management and they need to have some systems that "understand" their intention and that help them retrieve information. Here the "understand" does not mean artificial intelligence or machine learning, instead, it means the system should be designed according to user needs and reflect these needs.

Two widely used delivery systems for construction projects are lump sum and negotiated jobs. A construction delivery system includes the following stages: the Project Information Collecting stage, the Bidding stage, the Construction stage, and the Project Close-out stage. Figure 6-1 shows these four stages and the description of the activities for the UMDA system. In the Project Information Collecting stage of a lump sum job, the General Contractors (GCs) usually find the advertised project from public media, such as news papers, TV ads, and Internet. Then GC will begin collecting information about the project through all possible means. Normally the entity representing the Owner of the project will invite the interested and qualified GCs to visit the job site and furnishes them with the drawings, specifications, and all the job related documents for bidding. For a negotiated job, the Project Information Collecting stage usually lasts longer. When the Owner wants to build a project, s/he may invite one or more GCs to bid the project. Sometimes, the GC can help the Owner to design/build the project. When the GC is

involved in the design stage of the project, the GC will know a lot of details about the project, thus avoiding the risk of misunderstandings. After the project information has been collected, it will be stored in a database or as paperwork if the GC does not have a DBMS. By the end of this stage, since the project information has been roughly completed, the UMDA system will be ready to start building the project data tree. Then the information will be used in the Bidding stage to get a solid project estimated cost. Next the type of information and data that needs to be stored in the Bid Manual will be discussed.

Project Information Collecting →	Bidding Stage →	Project Construction Stage →	Project Close-out
Start building the project data tree	(1) PM Continues building the project data tree (2) PM loads information on the project data structure of the OODB (3) Retrieve information from existing company DBMS	(1) Using configurable visitor to retrieve information from project data tree; (2) Continue decorating the already built-up data structure	Using configurable visitor to retrieve information from project data tree

Figure 6-1 PM activities during various project stages

6.1.1 Bid Manual

In a lot of construction companies, the Bid Manual is a folder or a set of books storing the related documents for a single project that is in the Bidding stage. It normally is prepared by the General Contractor (GC). When the Owner agrees to the contract price and all the contract conditions of the project, the GC and the Owner sign the General Contract. Subsequently the GC will start the construction stage of the project.

The followings sections are included in the Bid Manual. For detailed information about all the items included in Section 9 to section 24, please refer to the Construction Specifications Institute (CSI) MasterFormat (CSINet 2004).

- Estimate: The estimate section is a summary of all the rest sections of the Bid Manual. During the bidding process, the Project Manager (PM) may estimate the price of the project several times in order to better control the estimate. Hence there will be multiple, different copies of estimate spreadsheets. Each of these estimate spreadsheets should be kept in this section. The PM or estimator may also make some notes or do some calculations about why changes were made to the previous project estimate. All notes and calculations should also be stored in this section.

- Permit: Different projects have different permit requirements. For a commercial construction projects in Florida, such as the project illustrated in Chapter 5, the drawings and specifications need to be sent to the South Florida Water Management Department (SFWMD) for site development plans review. If the SFWMD approves, then the drawings and specifications need to be sent to the county building department for the utility, electrical, landscaping, and architecture review. Usually the local redevelopment agency, wildlife protection agency, and other government agencies will review the drawings and specifications if the project will impact the local or regional nature or development. For hard-bid jobs, this section usually will usually be empty. For the design/build jobs, because the General Contractor has to be involved in the design process, all the permit documents are included in this section.

- Architect Correspondence: This section includes all the facsimiles, letters, notes, memos, sketches, drawings, calculations, and transmittals from the Architect to the GC. The characteristic of this section is that the materials included in it are project related, generated by the Architect, and cannot be stored in other sections.

- Engineer Correspondence: This section includes all the facsimiles, letters, notes, memos, sketches, drawings, calculations, and transmittals. from the Engineer to the GC. The characteristic of this section is that the materials included in it are project related, generated by the Engineer and cannot be stored in other sections.

- Contractor Correspondence: This section includes all the facsimiles, letters, notes, memos, sketches, drawings, calculations, and transmittals from the GC to others, such as the Owner, the Architect, the Subcontractors, the Suppliers, and other firms. The characteristic of this section is that the materials included in it are project related, generated by the GC and cannot be stored in other sections.

- Owner Correspondence: This section includes all the facsimiles, letters, notes, memos, sketches, drawings, calculations, and transmittals. from the Owner to the GC. The characteristic of this section is that the materials included in it are project related and generated by the Owner.

- Notes: All the important notes that cannot be stored in the rest of the sections are stored here. For example, when the PM reads the drawings or specifications, s/he may notice some discrepancies or mistakes in them. It is a good habit to make a note and keep it in the Bid Manual, because these discrepancies or mistakes shall be considered when estimating the project price and they may cause project change orders later. Another example of information to be stored here is the comments and suggestions from other people on the project team (superintendents, project engineers, senior project managers, etc.).

- Finish Schedule: Usually the Owner specifies in the general contract the project start and complete dates. Sometimes the Finish Schedule also needs to be inserted in the general contract. Similar to what the PM will do with the Estimate section in the Bid Manual, all the final or preliminary schedule charts and notes are included in this section.

- General Condition: Section 9 to section 24 is set up based on the Construction Specifications Institute (CSI) format. For the 16 divisions of the CSI format, some subcontractors could possible do the work in more than one division. For instance, the concrete subcontractor may also do masonry work. But some subcontractors can only do parts of the work included in one division. For instance, the flooring contractor can only do part of the job in the Division 5-Finishes. The GC will sub out most of the project to a lot of subcontractors. In the bidding stage, the PM will contact two or more subcontractors in each of the areas of scope of work. Too many subcontractors in one scope of work will complicate and delay the PM's estimating work. In the General Conditions, the documents included are the proposals from the dumpster and waste management firms, the proposals from jobsite trailer rental suppliers, the proposals from general equipment rental suppliers, and the proposals from the construction cleaning contractors. All the proposals from subs that relate to Division One (General Condition) of CSI format are included in this section.

- Site Work: All the proposals from subs that relate to CSI Division Two (Site Work) are included in this section, such as the proposals from site development subs, the proposals from surveyors, and the proposals from landscapers.

- Concrete: All the proposals from subs that relate to CSI Division Three (Concrete) are included in this section (such as the concrete subcontractors, concrete sidewalk subcontractors, etc.). Most concrete companies can do the furnishing, delivery, and installation of concrete reinforcement work and will include the reinforcement in their proposals. Some structural embed materials, such as steel column base plates, are usually provided by the structural subcontractor and are installed by the concrete subcontractor.

- Masonry: All the proposals from subs that relate to CSI Division Four (Masonry) are included in this section. As described previously, some concrete subcontractors will have this scope of work included in their proposal.

- Metals: All the proposals from subs that relate to CSI Division Five. (Metals are included in this section, to illustrate a few, furnishing and installation of steel beams, steel columns, pipe bollards, and aluminum works, etc.)

- Carpentry: All the proposals from subs that relate to CSI Division Six (Carpentry) are included in this section. This section is a little complicated. It not only includes the rough lumber material and labor (such as the furring material for drywalls and siding and the related labor), but also includes finish carpentry material and labor (such as the molding material and labor). In addition, it also includes the material and labor for cabinetry, and trusses. Most forest products providers supply trusses, rough lumber materials, and finish carpentry materials. They usually also supply doors, frames and hardware and windows.

- Thermal and Moisture: Included in this section, are proposals for insulation material and labor (such as the blown insulation on trusses, insulation board on masonry walls, and matt insulation for interior drywall partitions), roof (for instance, metal roof material and labor), and some special thermal moisture protection proposals from subcontractors (for example, vinyl siding or hard plank siding).

- Doors and Windows: The proposals from subcontractors for the furnishing and installation of doors, frames and hardware, windows, storefront, coil doors, and automatic doors are included in this section. One situation that will happen in construction projects is that a subcontractor can do both the furnishing and installation of doors and windows. Some times, the supplier and installer are different subcontractors. Thus it will be very important to double check the quantities in their proposal and make sure that both of them understand their scope of work.

- Finishes: All the interior and exterior finish materials and labor are included in this section. For example, drywall (including studs and gypsum boards material and labor), stucco and plaster, painting, wallpaper, flooring (carpet, ceramic tile, VCT, etc.), and acoustical ceiling. Usually each of them furnished by different subcontractors.

- Specialties: Specialties includes all the bathroom accessories, such as tissue holder, mirrors, towel dispenser, grab bars, mailbox, flagpole, and signage. One important specialty is fire extinguisher and/or fire extinguisher cabinet. The proposals from subcontractors for this section of the works are included in this section.

- Equipment: The proposals from subcontractors for the furnishing and installation of kitchen appliance (microwave, oven, refrigerator, ice maker, etc.), audio/video systems, and some other equipment (such as washer and dryer, ice cream machine, etc.). Subcontractors under this section of work have to cooperate with the plumbing and electrical subcontractors to finish this scope of work.

- Furnishings: The proposals from subcontractors for furnishing and installation of drapes and blinds, which are for window shade system, will be included in this section. Other works include: fabrics, artwork, manufactured casework, furniture and accessories, rugs and mats, multiple, seating, interior plants and planters. A lot of times, the Owner of the project would like to select the subcontractors for these works, or even didn't include these works on drawings or specifications. Except for the drapes and blinds, which require having backing materials on walls, most of the works included in this section can be done after the construction of a project. If the Owner elects to do these works after the delivery of the project and contracts with the individual companies by him or her -self, s/he can save some money.

- Special Construction: A lot of special construction items are included in this section. For details, people can refer to CSI website. Special floor finish and swimming pool works are the most frequently occurring ones and are included in this section.

- Conveying System: This section includes all the conveying systems specified in the construction documents. Elevators and escalators works are included under this section. Again, elevator companies have to work together with the concrete / masonry subcontractor and the electrician to make sure the installation is proper.

- Mechanical: This section includes all the mechanical work, for example, plumbing, water line extensions, sewer line extensions, underground utilities, fire hydrants, fire sprinkler system, HVAC, and water line certificate. Site developers furnish and install all the water line, fire line and sewer line piping, and manholes outside the building. Site developers usually stop their piping or tie-in location at 5-10 feet outside the building. Accordingly, the plumber will need to do all the inside piping and extend the pipes to 5-10 feet outside of the building and connect with what site the developer has done there.

- Electrical: The electrician is usually responsible for the Site electrical, site lighting, and light fixture. For the TV and telephone systems, the electrician usually is responsible for cabling and stubbing to the TV or telephone outlets. The cable-TV company and the telephone company will find the closest tie-in location for the cables and tie the cables to the building (The telephone company has to do the telephone board also). Then these companies will be responsible for installing the outlets and adjusting the systems.

Compared with the eight databases shown in Figure 3-1, the following contents are not included in Bid Manual: Purchase Order Database, Drawing Database, Productivity Database, Specifications Database, Contracts Database, and Storage Information Database. Purchase order and contracts databases are not included because there is no contract or purchase order at the bidding stage. Drawings and Specifications are provided

by the Architect and are usually separated from the Bid Manual. As shown in Figure 6-1, drawings, specifications, and project documents are already organized into the project data tree. During the bidding stage, the project data tree will be populated with information about the subcontractors, the estimate, and the schedule. The productivity database and storage information database store the statistical data of the company and usually shared by all the projects that the company has or will have.

6.1.2 Bid Action

Detailed description of the work of a PM in the project bidding stage and the business use case for the complete project bidding process can be found in Xie, et al. (2003). Besides the Bid Manual database and the above-analyzed databases, some other databases involved in bid action are the subcontractor database and the drawings and specifications database. Next the user models done in UML and some samples of the functions specified by the bidding process user model will be proposed.

Shown in Figure 6-2 are the databases involved in the project bidding process and the relationships between them. Figure 6-2 is different from Figure 3-1 in Chapter 3. Figure 3-1 in Chapter 3 is a general-purpose figure showing the relationships of the databases in construction management, but Figure 6-2 is done from the Project Manager's view. Figure 6-3 shows the conceptual design of Graphical User Interface for collecting information from the Subcontractor Database. The main operations on the Subcontractor Database are add, edit, and delete. The delete operation on Subcontractor Database is not recommended, because unless a subcontractor quits business or goes bankrupt, the General Contractor should keep as many subcontractors in the database as possible. Figure 8-6 in Chapter 8 shows the implementation of the design idea in Figure 6-3.

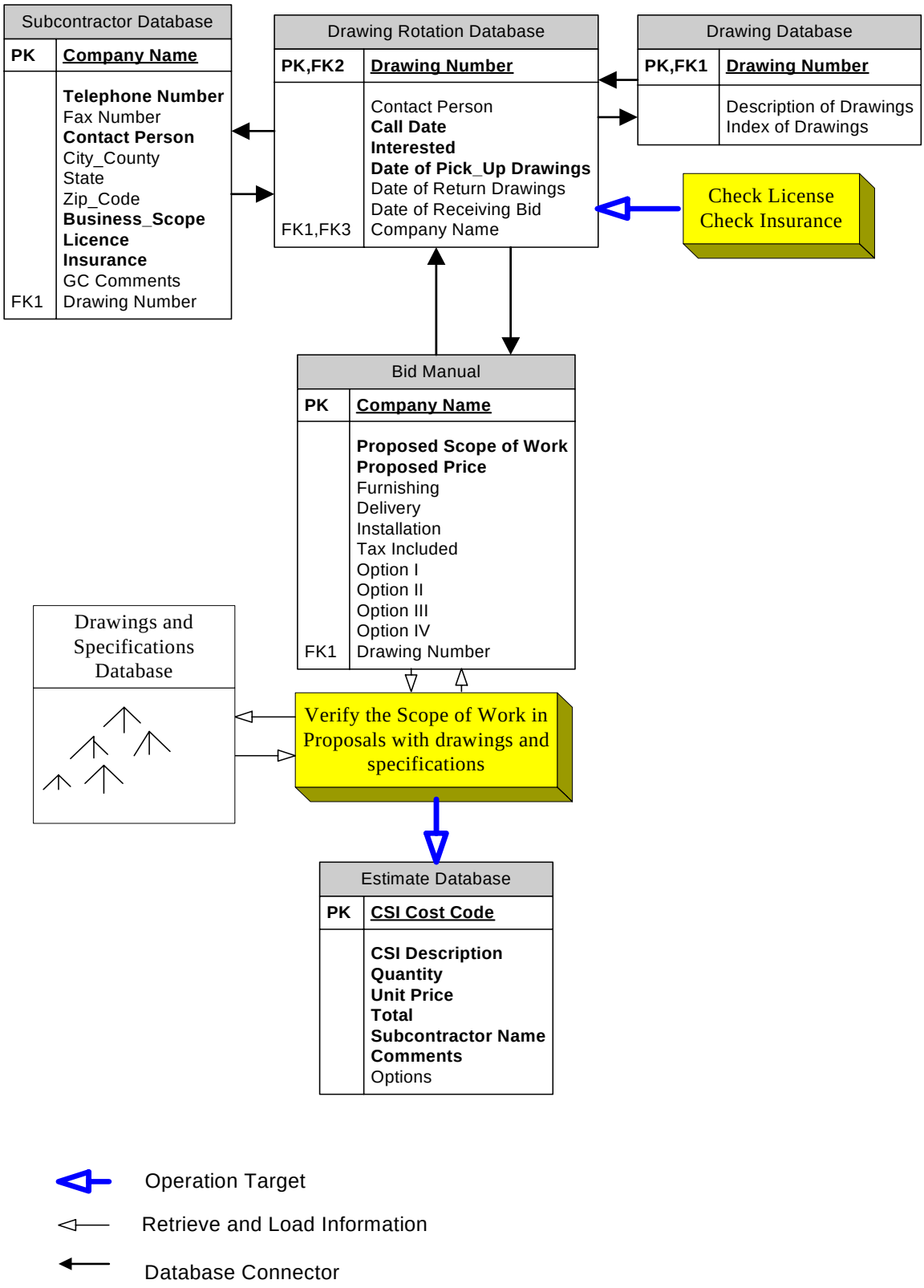


Figure 6-2 Class Diagram and the relationships involved in a project bidding process

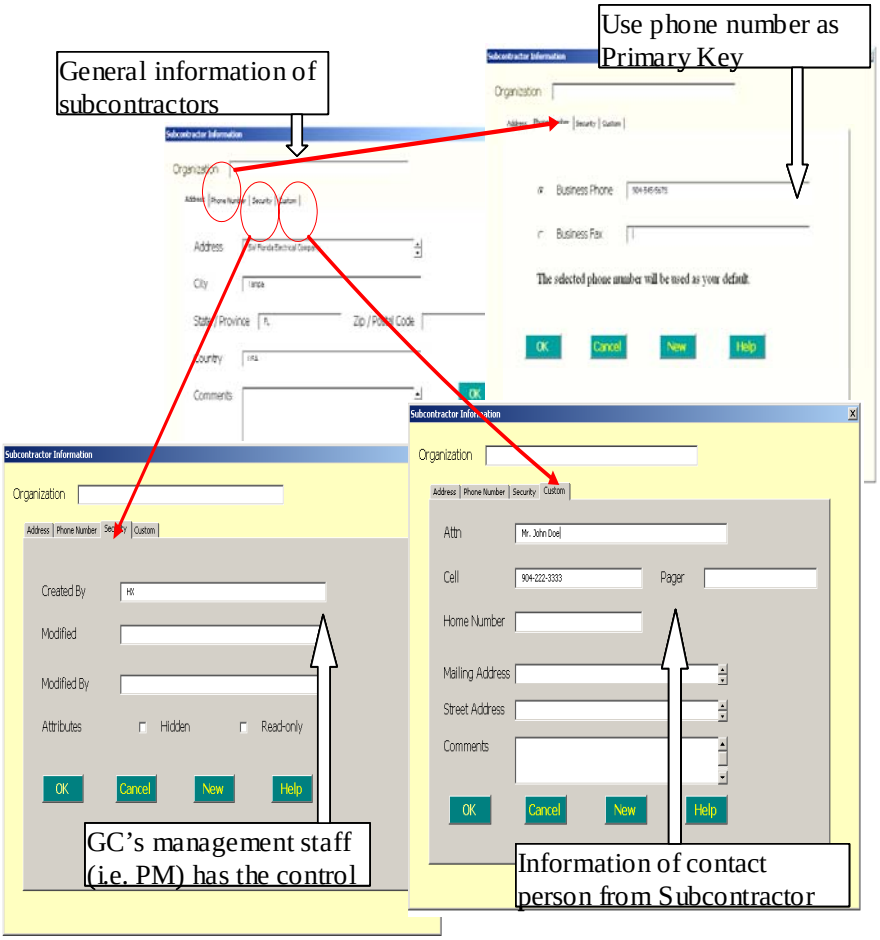


Figure 6-3 Conceptual design of graphical user interface for collecting subcontractor information

Figure 6-2 shows two operations on the databases involved. One operation is to check for the license and the insurance of all the subcontractors invited to bid on the project. It is required that all the contractors on a project have appropriate licenses and insurance including the correct workman compensation. For example, the UMDA system will provide an interface, asking users to enter the necessary information or conditions on the subcontractors the users want (i.e. the subcontractor) shall be located within 100 miles of the jobsite. Usually GC will avoid using subcontractors from places faraway from jobsite to minimize the impact of travel and make the pricing more competitive.

After the user clicks the “Find” button on the interface of the UMDA system, the system will start searching for the qualified records accordingly.

The following are some system design issues: What if the qualified records are less than wanted? What if the selected records are not what we want? In order to resolve these problems, it is proposed that two tables be used to store the temporary and final search results. One table is for all the needed records. The system will filter out the unneeded ones from the company subcontractor database. For example, if a project does not have an elevator, all the elevator subcontractors can be filtered out from “AllNeeded” table. The “AllNeeded” table is very useful and undividable. “Selected3” is the table storing the final selection records. If a user needs to add new records from external sources (i.e. new subcontractors) the records will be inserted into the company subcontractor database, the “AllNeeded” table, and the “Selected3” table.

The other operation is to verify the scope of work in the subcontractors’ proposals with the project drawings and specifications. The purpose of this operation is to build up the estimate of the project. At the same time, the PM will load information on the tree structure of the Object Oriented Database (OODB). So during the bidding stage, there are some Entity-Relation (ER) models. With the completion of the estimate spreadsheet, the PM will make the data structure of the project database “solid”, which means loading most of the project information onto the data structure. As will be discussed later, during the construction phase, the PM will search for information on the developed data structure.

6.1.3 Construction Stage Contractor Book

There are a lot of similarities between Construction Stage Project Book and the Bid Manual. The only difference between the Bid Manual and the Construction Stage Project

Book is that there are a lot of documents from subcontractors in the Project Book. In the Construction Stage, the subcontractors who have been awarded their scopes of works will get involved in the construction phase according to the project time schedule. Another difference is in the volume of the paper work included in these two types of books. The Bid Manual has fewer documents than the project book.

In the proposed research, the User Model will reflect how people retrieve project information in their daily work. In order to build a database and the management system that can help people manage information efficiently and retrieve information quickly, we need to first analyze how the project information get managed usually by a project manager and discuss where can we improve the information management. For a typical project similar to the one proposed herein, all the information related to bid stage is grouped into the Bid Manual; then all the information related to construction phase is grouped into the Project Management Book. At the end, when the project has been finished, all the close-out documents and drawings will be filed into the Close-out Book.

The Project Management Book should include the following contents:

- Contract: include the contract between the Owner and the General Contractor.
- Permit: including all the material related to the permit application process and the correspondence between the General Contractor and the local building department.
- Schedule: includes the original time schedule and the entire updated time schedule. During the construction of the project, there may be some un-predictable factors that affect the duration of some trades and these in turn, may affect the total duration of the project. On the other hand, some trades may finish their work early and save time in the schedule. So the project schedule should be updated every two weeks. The entire updated schedule will be stored in this section. Besides the time schedule, other materials related to the schedule could also be stored there, such as the record of the names of the subcontractors who have already received copies of the schedule.
- Sub-List: the next section is the sub-list. All the names of the subcontractors that have subcontracted for the project will be stored in this section. For multi-phased

projects, the list of all the subcontractors is not complete at the beginning of the construction project. During the construction of the project, the General Contractor would find more subcontractors and fill their names into this list.

- Miscellaneous Notes: the following information could be included in this section: (1). Index of Drawings: there may be multiple revisions of the drawings. This index reflects all the drawing and revision dates. (2). Miscellaneous notes. Basically, all the notes that relate to project but could not be categorized into all the other sections are included in this section.

- GC Correspondence: Similar to the features of the documents stored in the Contractor Correspondence in the Bid Manual. But the only difference is that the documents stored here are from the date that the GC was awarded the contract.

- Architect Correspondence: Similar to the features of the documents stored in the Architect Correspondence in the Bid Manual. But the only difference is that the documents stored here are after the date that GC was awarded the contract.

- Engineer Correspondence: Similar to the features of the documents stored in the Engineer Correspondence in Bid Manual. But the only difference is that the documents stored here are after the date that GC was awarded the contract.

- Owner Correspondence: Similar to the features of the documents stored in the Owner Correspondence in the Bid Manual. But the only difference is that the documents stored here are from the date that the GC was awarded the contract.

- Meeting Minutes: Included in this section are all the records of the meetings related to the project. For example, the meeting between the Owner and the General Contractor, or the meeting between the General Contractor and the subcontractors to solve some problems on the project.

- Supplemental Instructions.

- Construction Change Directory: The Project Manager, Project Engineer, Project Superintendent, or subs may find some possible future changes in the project either from drawings or caused by project site condition. But these possible changes have not affected the project yet. All the record or materials about these possible changes are put into this section.

- Architect Field Report: When the Architect visits the project jobsite, he or she may make notes of the project process, the existing problems if there is any, or some instructions or clarifications on the project. All these notes with the Architect's signature and date will constitute the legal file of the project.

- Request For Information (RFI): All the RFI documents sent from the GC to the Architect and the response from Architect will be stored in this section. Usually the

answers to the questions related to individual subcontractors will be sent to them separately.

- Steel Test Report: The Architect may specify in project documents or on drawings the requirement for Steel Test Reports. These reports may be required for high-rise buildings, steel structure buildings, or industry projects.

- Concrete Test Report: The concrete tests are done by independent laboratories and the results will be analyzed and listed in a test report. Copies of the concrete test reports will be sent to the Engineer/Architect and the General Contractor. Each of the concrete test reports needs to be put into the project book.

- Soil Test Report: Similar to concrete test reports. Copies of the reports will be sent to Engineer/Architect and General Contractor.

- Fire Sprinkler Test Report: Similar to the concrete test reports. Copies of the reports will be sent to the Engineer/Architect and the General Contractor.

- Test & Balance of Specified Pipes: Similar to concrete test reports. Copies of the reports will be sent to Engineer/Architect and General Contractor. Usually for the HVAC system or the fire sprinkler system, the specifications may require the General Contractor to provide test and balance on the system. In such a situation, the General Contractor can ask independent laboratories to do such a test.

- Medical Gas Certificate: For medical projects only. The General Contractor keeps this for their records.

- Fire Alarm Certificate: For the project with Fire Alarm system installed, the General Contractor needs to apply for Fire Alarm Certificate. This Certificate will be sent to the Fire Marshal and is required on the application for a Certificate of Occupancy. The General Contractor keeps this for their records.

- Elevator Certificate: For projects with installed elevators, the General Contractor needs to apply for an Elevator Certificate. The General Contractor keeps this for their records.

- Certificate of Occupancy (COO): Issued by the Building Department of the local government. The COO grants the Owner the right to move into the building. The General Contractor will have a copy of the COO for their records.

- Request for Proposal: Includes the forms and transmittals from the General Contractor to subcontractors for their proposals.

- Pending Change Orders: Includes the Change Order proposals from General Contractor to the Owner regarding the changes in the project.

- Change Orders: Includes the Change Orders approved by the Owner.

The following sections are related to the CSI cost code and individual subcontractors. Depending on the different types and sizes of the project, the project book will have different groups of information included in it. For example, on a large project, which is a new construction and very complicated, may have a lot of activities listed in the CSI MasterFormat (CSINet 2004) (for example, Asphalt Paving; Landscape; Concrete; Soil Treatment; Masonry; Structural Steel; Light Gauge Framing; Aluminum Work; Rough Carpentry; Trusses; Finish Carpentry material and labor; Cabinetry; Insulation; Roofing; Siding; etc.). For a small project, with only interior renovations, the activities included in the project book will be simpler (for example, Doors / Frames / Hardware; Automatic Doors; Store Front; Windows; Stucco; Drywall; Ceramic Tile; Acoustical Ceiling; Resilient Flooring; etc.). For the first case, each activity may be further divided into sub-activities. The followings are examples of how the activities are further detailed.

- General Condition: Included in the General Condition section are all the items that were put in Division One (General Condition) part of estimate. For example, the fee paid to county utilities for fire hydrant meter installation permit; Engineer fees; portable toilet; mobile office trailer (monthly lease, delivery freight, block level & tie-down, knockdown, return freight, steps, etc.); dumpster; and test services (compaction test, moisture density relationship, concrete test cylinders, etc.).
- Survey: This section actually could be included in Division 2 (Site Work) and Division 3 (Concrete). The reason to separate this scope of work out is that it is usually provided by different subcontractor (surveyor) from the site work subs and concrete subs. The materials should be put into this section are the Purchase Order made from the General Contractor to the Surveyor attached with a detailed scope of work. All the survey reports, foundation survey, and survey as-built drawings should be included under this section.

6.1.4 Construction Stage Actions

After the project manager or estimator has finished estimating the price of the project, the Contractor can start to negotiate the price with the Owner or submit a lump

sum bid on the project. After the Owner accepts the entire estimate and the conditions of the general contract, the project will start with the construction phase.

During the construction phase, the Project Manager needs to do multiple tasks. For the proposed synthetic project, the information relates to it is too voluminous and needs to be categorized to ease its management.

The PM's actions during the construction stage can be summarized as follows:

- Maintain the Project Manager's book (including all the project documents RFIs, Memos, etc.)
- Prepare the Contract
- Prepare the sub-contracts or Purchase Orders, and Scope of Work
- The quantity take-off and the estimate done in the bidding stage can be used to prepare the subcontract and purchase order.
- Prepare a detailed schedule of the project
- Collect and check the submittals from the subcontractors and maintain submittal log
- Collect and read the field report and perform project process control.
- Talk with the superintendent; find out what will be needed on the project.
- Negotiate with the subcontractors.
- Pending a Change Order: prepare a letter to architect
- Change Order: maintain a Change Order Log
- Site visit: make sure about the critical points.
- Collect close out documents (inspections, as-built drawings, warranties, etc.)

6.2 System Requirements on the Proposed Architecture

In order to ease the analysis,the author only focus the system design on two groups of users: the construction Project Managers and Superintendents. As shown in Chapter 2,

the user model connects closely with software design. The IBM OVID project is an example of a software design process which is based on the observation and analysis of multiple users to generate a single user model. This synthesized user model will be able to fit the needs of at least one group of users. In addition, the IBM OVID process is in a circular design process, and users will be able to input opinions after the primary system model has been set up. Another application of the user model is to set up the users' accounts according to different users and provide log-in for multi-users. Examples of these systems are Windows XP and other "distributed" Operating Systems. In these systems, the user models are represented in the different user accounts. For each user account, there is a different user document space, different user preferences for favorite websites, and different program packages.

In the UMDA system, the users are divided into "individualized" user groups, for example, Project Managers and Superintendents. For each user group, configure the GUI and Application Layer will be configured according to user and task profiles. The extension of the configuration from fitting each user group to fitting each person will be addressed in future research.

6.2.1 Primary User Interface

The UMDA system applies a User Model Driven Architecture to help users retrieve the needed information. The first thing this system needs to have is the interface to communicate with users. The program language for this proposed project is Java. Shown in Figure 6-4 is the conceptual design of graphical user interface for the User Model Driven Construction Information System. It provides login interfaces for different groups of users. Users from outside of the company are not allowed to read or search for the project information generated and maintained by the construction company. After these

company visitors log into the system, the system will lead them to an interface where they can provide information about themselves and that information will be stored in the company's visitor database. The information about one group of visitors is very important: the subcontractors. Figure 6-5 shows the interface of the system asking the subcontractors to fill in the information.

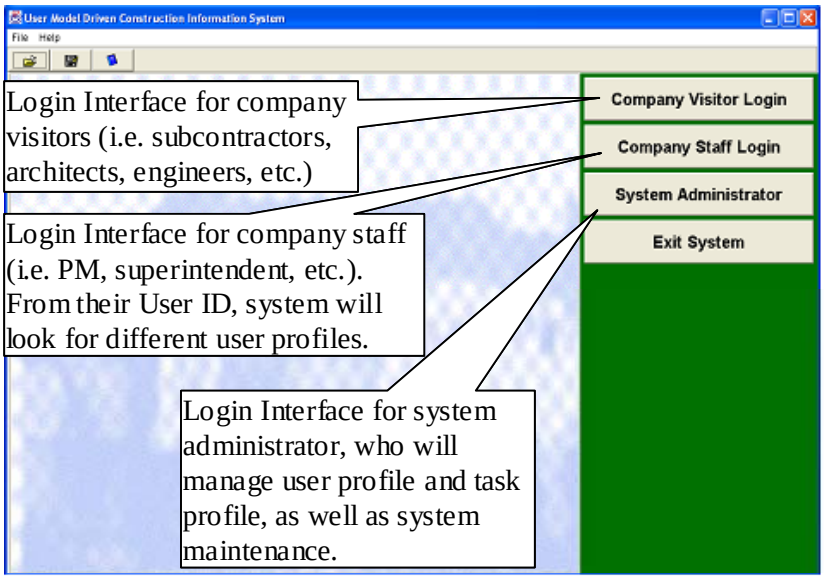


Figure 6-4 The graphical user interface for the User Model Driven Construction Information System

6.2.2 User Input

When a user logs into the UMDA system, he or she needs to provide some personal identification information, such as the user name and password. From the user name the system will know which user group this user belongs to. The password check is to make sure that no unauthorized person will be able to reach classified information. This function will provide a certain degree of security to the system. After user logs into the UMDA system successfully, the system will use knowledge engineering in adaptive interface and user modeling. In the approach proposed herein, Terminology Databases

and Compare and Match Networks will be used to process the query tokens and to find out the best suited system command for the task process queue.

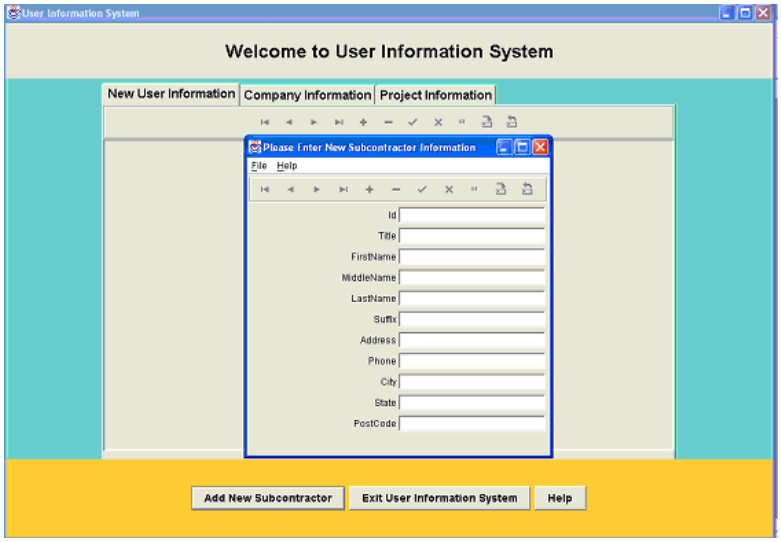


Figure 6-5 The interface for subcontractors to fill in their company information.

6.2.3 Customization of system design

In the proposed approach, the core part of the user modeling is stored in User Profiles and Task Profiles databases. The Task Analysis Module will activate the User Profiles Database when the user logs in and it will activate the Task Profiles Database when the Task Process Queue is received.

6.2.4 User Model Driven Architecture

After user and task profiles are established, they will be used to configure the GUI and application layers of the User Model Driven Architecture. When a user asks a question to the system, s/he may not know how many databases they need to access and to retrieve the required information. One way is to open all the databases and get information from all of them. It is relatively easy to implement on the software design side and the search result will include all possible answers. But on the user's side, few users will have the patience to read through all the records in the list and to try to find out

the useful information for themselves. In the above example, when the system disassembles the query "Show me all possible work in next 2 weeks" into tokens and finds out that none of the tokens can be replaced by the terminologies, the system will pass all the tokens to the Task Analysis Module and the Module will compare the tokens with the data in the task profiles. All the useful tokens will be sent out to the ElementAmbassador and the Modified Navigator with the attributes requirements, which specify the databases the user needs to open. This process represents the configuration of application layer.

6.3 Connection with project database

There are a couple of possible ways to integrate the project databases together and retrieve information from them. The Visitor design pattern and the ElementAmbassador and Navigator pattern are both Java language design patterns and are suitable for the purpose of connecting to the project database.

6.3.1 Visitor Design Pattern

After the discussion of data structure (in Chapter 3, section 6.1.1 and section 6.1.3) and the discussion of user modeling (in Chapter 2, section 6.1.2 and section 6.1.4), the project information will be organized onto a tree type data structure, and will design the approach to load information onto and retrieve information from the data structure. The system has been developed based on the user model. The next step is to look at the user's request, which will define the Visitor Manager's functionality.

The computation is activated by a user's request for some content or service. The request is intercepted by the Visitor Manager (VM), who is responsible for deciding which action to take for servicing it. The VM dispatches the request, in the form of a "request for action," to the suitable component of the system model. The system model

incorporates the business logic for performing the action, and executes such logic, which updates the state of the application and produces a result to be communicated to the user. The change in the state of applications triggers the most appropriate View, which builds the presentation of the response. Such presentation typically embodies interaction objects, whereby the user may pose a new request and reactivate the computation process.

The system model encapsulates the business actions required for answering a user's request and keeps the state of the application. Actions are the actual components that implement the business logic. They are designed to be reusable by different applications, possibly using different front-ends. The possible actions are contained in the system model in the form of object-oriented components, sometimes called action classes. The system model should ignore the format in which requests are posed, and the way in which the response is constructed and presented to the user.

The View embodies the presentation logic for assembling the user interface. An application may have a single View or multiple Views, and a View may be composed of sub-Views, relevant to different types of results. The View should ignore where the results to be presented come from and the details of the request originating such results.

The VM is the traffic cop of the architecture, responsible for interpreting the user's request, producing the appropriate request for action, examining the result of each action, and deciding what to do next. The VM is totally unaware of the business logic of the action it invokes, and of the presentation logic of the View. The VM maps the request to the suitable action, by creating an object of the action class and calling one of its functions.

Each action class is a Java class wrapping a particular application function, operating on the state of the application. In the simplest situations, an action class implements all the business logic needed to serve the request. In more complex scenarios, the action class may collaborate with other objects for fulfilling its tasks. For example, execution of a database query could be an action class. If the invoked action needs to update the state of the application, it may create or modify appropriate objects of the system model, called state objects. Another way to update the state of the application is to decorate or modify the attributes of objects. The state objects may last just long enough for servicing the request, or persist between consecutive requests; for example, they may store the result of a data retrieval query. After completion, the action communicates the outcome of execution to the VM, which decides what to do next.

In the typical flow of control of an application, after an action completes, the VM invokes a Report Manager, which controls the view to represent to the users. The Report Manager is responsible for presenting the updated state of the application to the user. In doing so, it accesses the state objects or attributes of the system model, where the current state of the application is stored, and sends information back to the user and the result of a database query will be displayed.

The Visitor Manager, which dispatches the client requests to the actions that serve them, is typically implemented as a configurable servlet, which reads the binding between user requests and actions from the configuration file. With this solution, the result to display after executing an action can be changed simply by editing the action mapping in the configuration file, without updating the code of the Visitor Manager.

The Action Classes, which are invoked by the Visitor Manager to serve requests, present to the Visitor Manager a very simple interface, typically consisting of a single function with a fixed name. In this way, the Visitor Manager needs to know only the name of the action class to call for any given request, and remains unaware of the details for invoking the real business services. Such details are known only to the action class, which "wraps" the business functions and plays the role of a "mediator" between the Controller and the business services.

The effect of executing action classes and business functions is recorded in the system model, as a set of state objects. These are typically very simple objects, with a standard interface consisting of functions for setting property values (called setters) and for getting the value of properties (called getters). In the Java world, objects of this kind are called JavaBeans. The JavaBeans representing the application state are typically produced by the execution of actions, and consumed by the Report Manager of the View.

If one component of a software system has been updated, how could your make sure that all the other system's component parts can be plugged together to work as required? In response to this question, it is proposed that a visitor pattern be used for the information retrieval process.

6.3.1.1 Visitor Pattern in Information Retrieval

The Visitor is a design pattern that describes how to traverse complex data structures without changing the class of the object being traversed (Gamma 1995). The visitor pattern is a technique for separating the code executing the instructions from the users from the code for the data structure itself while traversing a data structure. The Visitor Pattern allows the action taken at a node to depend on the type of the node. One possible implementation of the visitor pattern is in compilers because compilers often

traverse a data structure several times and based on the different type of nodes on the data structure, the compiler can apply the appropriate actions. In the proposed project, it is assumed the construction project data are organized on an object-oriented tree structure. It is further assumed that the applications for loading information and organizing data onto the data structure have been designed. The visitor pattern can be implemented in the applications to traverse the data structure. The visitors can traverse the data structure once for type checking and computation of binding information, and once for code generation. Because the method executed at a node depends on both the type of the visitor and the type of the node, this technique is called double dispatch. One benefit of this visitor pattern approach to system design is that it helps to draw the line between two developing teams. One developing team can work on the data structure design and the associated algorithms without worrying too much about the user end, while the other team only needs to focus on the user analysis and provide feedback to users. For the latter team, the knowledge is concentrated in the visitor implementation, which makes it easy to generate different types of documents and works well with interfaces. It also has the advantage of localizing the generated code for the node on the data structure in one or more specialized classes, rather than littering it through the application's object model.

Grand (2002) discussed the visitor pattern and the possible ways to implement this pattern. The Visitor pattern allows the client object to be closely coupled to the construction of the new complex object. Instead of describing the content of the objects to be built through a series of method calls, the information is presented in bulk as a complex data structure. The Visitor pattern can be used to encapsulate operations in a single class that would otherwise be spread across multiple classes. It also allows

encapsulating the logic for simple manipulations of a parse tree in a single class. Grand mentioned, “One way to implement an operation that involves the objects in a complex structure is to provide logic in each of their classes to support the operation. The Visitor pattern provides an alternate way to implement such operations that avoids complicating the classes of the objects in the structure by putting all the necessary logic in a separate class. The Visitor pattern also allows the logic to be varied by using different Visitor classes” (Grand 2002, p. 427).

Johnson (2003) talked about using the visitor design pattern to facilitate XML generation. In his approach, each class in an object graph implements a `Visitable` interface. An XML visitor is responsible for traversing the graph and generating an XML document. This is a much superior approach to hard-coding element generation in application objects. Johnson claimed: “Making each class in the graph implement the `Visitable` interface may prove useful for other tasks; the Visitor design pattern is very powerful.” (Johnson 2003, p. 241)

Basically, under the following circumstances, the visitor pattern would help to facilitate system design:

- There are a variety of operations that need to be performed on an object structure.
- The object structure is composed of objects that belong to different classes.
- The types of objects that occur in the object structure do not change often and the ways that they are connected are consistent and predictable (Grand 2002).

In the UMDA system, the implementation of the following visitor classes can be planned:

- **InformationSearcher:** This class starts the entire visitor processes and information retrieval.

- **ProjectVisitor.** This is an abstract class and provides logic for its subclasses. Its subclasses explore the objects that constitute the entire data structure —

construction project tree, on which we have organized all the information regarding a construction project. The concept is that instances of subclasses of the ProjectVisitor class visit the objects and gather information from each object, and then operate on the information.

- **TOCVisitor.** This is a subclass of the ProjectVisitor class. It is responsible for generating a table of contents. It works by examining each Division object directly owned by a ConstructionProject object. When it finds a Division object with a style that is in the internal TOC table, it generates a corresponding table-of-contents entry. The TOCVisitor may go very deep down to the leaf node of the tree data structure. From a parent node, when the TOCVisitor finds the child node object match with the searching condition, it will use the child node to replace the existing table-of-contents entry. In Figure 6-6, the TOCVisitor builds up the pointers in the first column (pointers to project data tree).

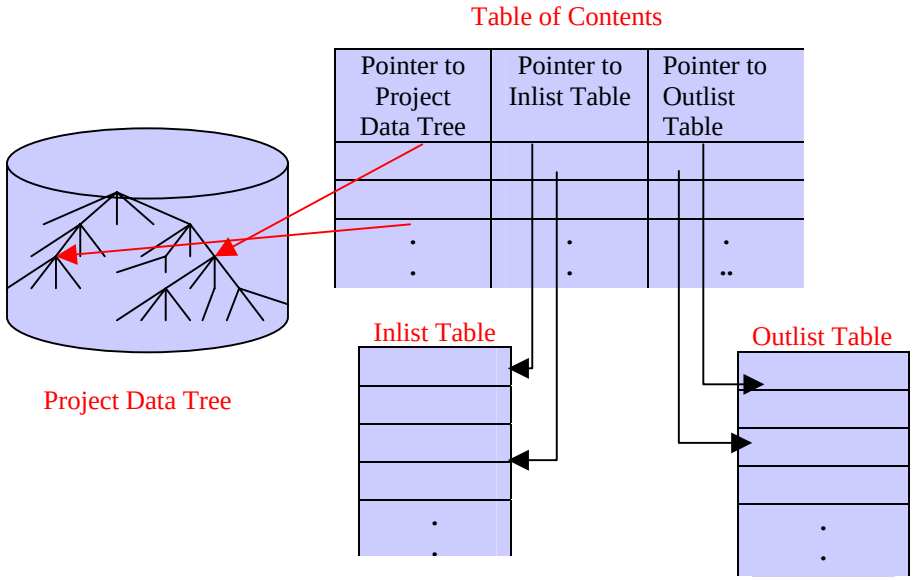


Figure 6-6 Contents and Project Data Tree tables

- **ReportVisitor.** This subclass of the ProjectVisitor class is responsible for report back to ReportManager process. It begins by being told to look for sub-nodes that correspond to certain levels of organization in a ConstructionProject object. It finds those levels of organization in the internal TOC table. It fetches the styles associated with those levels of organization from the table. It then follows the pointers and looks for objects that have the styles it fetched from the table. When it finds an object with one of those styles, it creates a new ConstructionProject object. It writes the new ConstructionProject object and all the objects associated with it to a file. In Figure 6-6, besides the pointers to the project data tree, the TOC table also stores the pointers to Inlist and Outlist. When ReportVisitor writes the new ConstructionProject object and the associated objects into a file, it will store the pointer address in the TOC table. After all the requested new ConstructionProject objects are stored into their specific files and the addresses of the files are saved in the TOC table, the ReportVisitor will call the ReportManager to let the

ReportManager access the TOC table and retrieve the files. Next time, when new addresses replace the old addresses in the Inlist column of TOC table, the "kicked-out" addresses will be saved in the "pointers to Outlist" column. Once the capacity limitation of the TOC table is reached, the first saved address in the "pointer to Outlist" column will be deleted.

Figure 6-7 shows the relationships and applications between the classes of Visitor, RealVisitor, AbstractElement, RealElement, VisitManager, and RootNodeClass. The functionalities of these classes are listed below. The proposed visitor format in Figure 6-7 and the function descriptions, follow the format suggested by Grand (2002), however all the functions of these classes have been changed or extended. There is a rectangle enclosing the AbstractElement and RealElement classes indicating that these classes are out of the scope of work of this study. But with the embedded accept method and the AbstractVisitor's name, a bridge has been built up between the underlying project data structure and the information retrieval process. In this version of the Visitor pattern, the AbstractElement objects determine which elements of an object structure a Visitor object visits and the order in which it visits them. The Visitor classes do not need to make assumptions about the structure of an object structure in order to navigate through it, so they are reusable. It has the advantage of keeping the Visitor classes independent of the structure of the object structure. However, one drawback of this approach is when Object structures are very large, the execution time may become too long for a Visitor to visit every object when it needs to visit only a small subset of the objects in the structure. But it takes less effort to implement and maintain the code and minimizes the dependency that the RealVisitor objects have on the object structure. The functions of each part included in Figure 6-7 are listed as follows:

- **VisitorManager.** The instances of the VisitorManager classes manage the TOCVisitor and the ReportVisitor. These instances are responsible for manipulating an object structure and the objects that compose it. They use

RealVisitor objects to perform operations on the object structures of the project data tree.

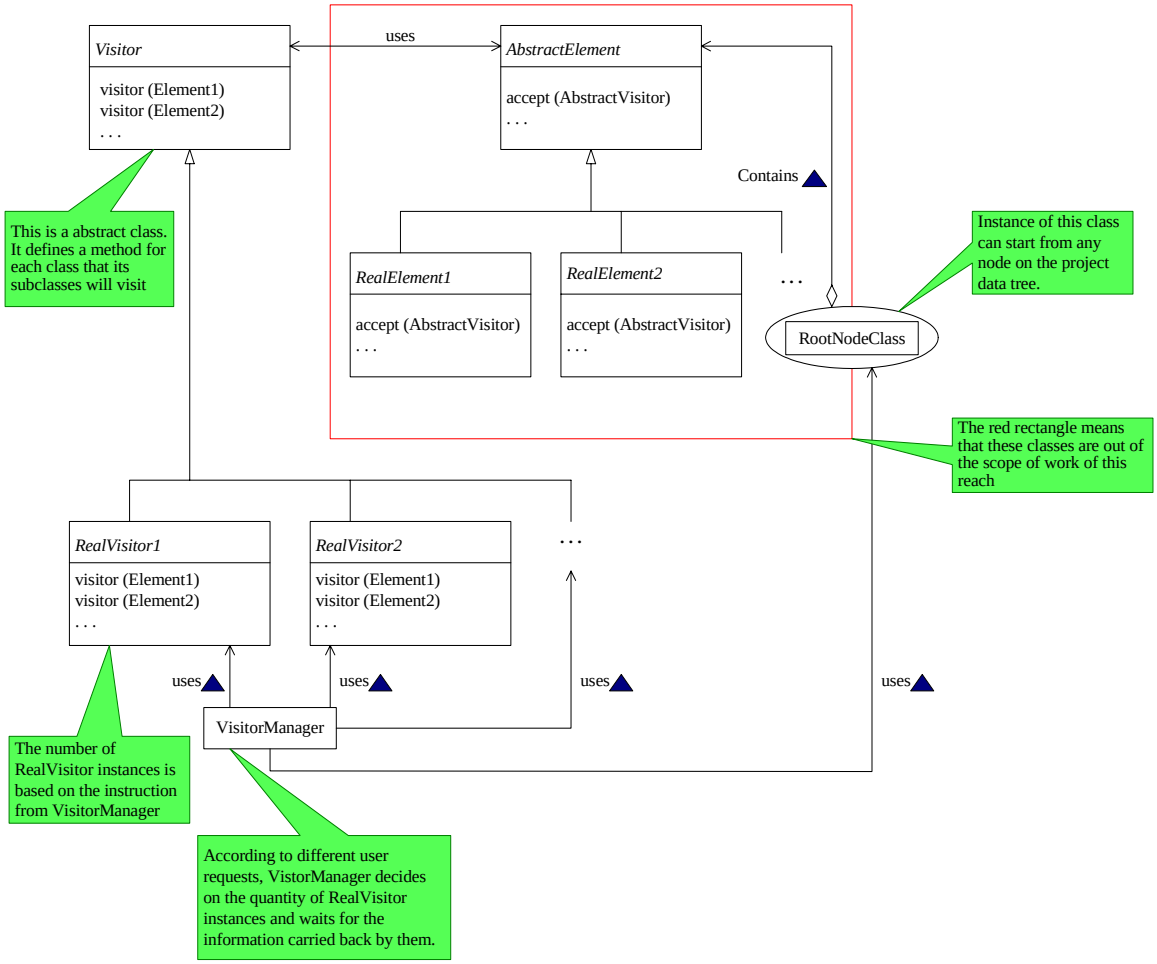


Figure 6-7 Proposed Visitor Pattern (Following the format suggested in (Grand 2002))

- **RootNodeClass.** Usually for one project database, one instance of the RootNodeClass is enough and it is the root object of the project tree structure. A Visitor object usually begins with an instance of the RootNodeClass and then moves on to other nodes in the object structure. The meaning of the instance of the RootNodeClass will be extended ReportVisitor calls it. ReportVisitor can start visiting the tree structure at any node and in that case, the starting node will be an instance of the RootNodeClass.

- **AbstractElement.** A class in this role is an abstract superclass of the objects that constitute the project data tree. It defines an abstract method as accept. It takes an AbstractVisitor object as its argument. This method works like a weld point and it joins the visitor part and the data part together. Subclasses of an AbstractElement class provide an implementation of `accept` that calls a method of the AbstractVisitor object and then passes the AbstractVisitor object to the accept

method of other AbstractElement objects. By doing this, the accept method helps the visitors travel along the project data tree.

- **RealElement1, RealElement2, and so on.** Computations are done on the objects of an object structure by passing an AbstractVisitor object to a RealElement object's accept method. The accept method passes the RealElement object to one of the AbstractVisitor object's methods so that it can include the RealElement object in its computation. When that is done, the RealElement object passes the AbstractVisitor object to the other RealElement object's accept method.

- **Visitor.** A class in this role is an abstract superclass of classes that perform computations on the elements of an object structure. It defines a method for each class that its subclasses will visit, so that their instances can pass themselves to Visitor objects to be included in their computations.

- **RealVisitor1, RealVisitor2, and so on.** Instances of classes in this role visit objects that constitute an object structure.

As a conclusion to the benefits achieved by using the visitor pattern, the following are the advantages listed in (Grand 2002): The Visitor pattern makes it easy to add new operations on an object structure. Because the ConcreteElement classes have no dependencies on the Visitor classes, adding a new Visitor class does not require making any changes to an AbstractElement class or any of its subclasses.

The Visitor pattern puts the logic for an operation in one cohesive ConcreteVisitor class. This is easier to maintain than operations that are spread out over multiple ConcreteElement classes.

A single Visitor object captures the state needed to perform an operation on an object structure. This is easier to maintain and more efficient than the way state information has to be passed as discrete values from one object to another.

Another consequence of the Visitor pattern is the additional work it takes to add new ConcreteElement classes. The ideal version of the Visitor pattern requires adding a new visit method to each ConcreteVisitor class for each ConcreteElement class added. A

direct consequence of the Visitor pattern is that ConcreteElement classes must provide access to enough of their state to allow Visitor objects to perform their computations.

6.3.1.2 Code Example

The following is code for some of the classes presented in the table-of-contents (TOC) design. First is code for the ProjectVisitor class that contains top-level logic for a project data tree. It is responsible for initiating operations that searching for information on the project data tree.

```
public class InformationSearcher {
   // The Project Data Tree currently being searched
   private ProjectDataTree project_tree;
...

   /**
    * Build a table of contents
    */
   private TOC fillInTOC() {
      return new TOCVisitor(project_ tree).fillInTOC();
}  // buildTOC()

   /**
* reports back to ReportManager process.
* reads the TreeNode from the TOC table
    */
   private ReportVisitor reportVisit (TreeNode level_node) {
     return (new ReportVisitor (project_ tree, level_node));
   }  // reportVisitor (TreeNode)

} // class InformationSearcher
```

The basic procedure for visitors is shown in the following algorithm:

```
/* Visitor Travel Procedure  */
 1) find the most specific actions  in the decomposed request such
                that for each element of decomposed request (dr) there exists an action
                that is subsumed by
2) for all  do
                a) for each parameter (p) of  the dr
                                identify the set of restrictions that subsume all
                                the restrictions associated to (p) in each of the
                                occurrences of  in (dr)
```

b) add to the Visitor Reporter the entry (I)
where is obtained from by associating to
each of its parameters the set of attributes
identified in step 2)
c) if the Visitor Reporter already contains an relative
to an instance B of an action and B is more
specific than the (I), then remove (I) (this step is done
to avoid redundancies in the Visitor Reporter).

The above InformationSearcher class tells the system which one is the object project data tree and then calls the visitors to start information retrieval. The ProjectVisitor class is an abstract superclass.

```
abstract class ProjectVisitor {
   private ProjectDataTree project_data_tree;
   private int start_node_index = 0;   // Index for navigating
   ProjectVisitor (ProjectDataTree project_data_tree) {
      this.project_data_tree = project_data_tree;
   } // constructor (ProjectDataTree)

   protected ProjectDataTree getTree() {  return project_data_tree; }

   /**
    * Return the next node that is a direct part of the tree
    */
   protected TreeNode getNextNode(TreeNode neededNode) {
      TreeNode myTree = project_data_tree;
      while (start_node_index < myTree.getChildCount()) {
         TreeNode next_node;
         next_node = myTree.getChild(start_node_index);
         start_node_index = start_node_index + 1;
         if (next_node.name == neededNode.name)
           return next_node;
      }
      return null;
   }  // getNextNode()
...
} // class ProjectVisitor
```

The following ReportVisitor class is responsible for visiting the node and its sub-nodes if any by following the pointers in the TOC table and maintains the TOC table.

```
class ReportVisitor extends ProjectVisitor {
```

```
   private TOCLevel[] levels;
   ReportVisitor(ProjectDataTree project_tree, TOCItem marked_item) {
// from the pointer marked_item, find the node in the project_tree
// bring all the items belongs to the node into a file
// make a pointer from the inList of TOCItem to the file
         ...
} // class ReportVisitor
```

The final listing is for the TOCVisitor class. The TOCVisitor class is responsible for building a table of contents. It navigates more deeply into a document's object structure, concerning itself with Document objects, Paragraph objects, and LineOfText objects. Its interest in LineOfText objects is that a table-of-contents entry will contain the text of the first LineOfText object in the paragraph that the table-of-contents entry corresponds to.

```
class TOCVisitor extends ProjectVisitor {
   private Hashtable myTOCTable = new Hashtable();
   TreeNode TOCVisitor (ProjectDataTree project_tree) {
       … …
       for (each sub-node of project data tree node) {
         if (node.name == visitor.name)
              return current_node;
         else if (node.subcontractor == visitor.subcontractor)
              return current_node;
         else if … …
   }
   TOC buildTOC(TreeNode current_node) {
      TOC toc = new TOC();
      … …
      addTOCItem ( current_node); //add the current_node on to the pointer list of TOC
      toc.number_of_tree_pointer = toc.number_of_tree_pointer + 1;
      return toc;
   }  // buildTOC()
} // class TOCVisitor
```

The VisitorManager should have the function of the Decomposition Hierarchy (DH), which is used for explaining how to execute complex actions. The InformationReporter is the Generalization Hierarchy (GH) function, which makes explicit

the relation between more specific and more general actions and is used for exploiting inheritance in their definition. For each user request, the VisitorManager identifies the actions and executes the following steps:

- Action identification: on the basis of the request uttered by the user, the VisitorManager will trigger a set of actions; each of them possibly represents the aspect of the requested task.

- Focusing: the set of real_visitors produced by the analysis of the previous sentences (such as the user's name, user's log in date, etc.) of the dialogue is updated to take into account the actions identified in step 1, real_visitors that can not be related to them are discarded.

- Follow the user instructions and downward expansion along the project data tree: the actions on the project data tree are repeatedly decomposed into more elementary ones until, as suggested by the user model, the user knows how to perform every action in the frontier of the project data tree or the visitor reaches the leaf node of the tree.

  The actions executed by InformationReporter are as follows:

- When there is ambiguity or the user is not sure of the next action, there is an evaluation of the relevance of the ambiguity among items in Table of Contents: if more than one TOC item is active, the ambiguity is evaluated, for establishing whether a single answer can be given to the user's utterance. If this is not the case, the visitor reporter will start a clarification dialogue, for identifying the user's intentions and solving the ambiguity.

- Selection of the contents of the answer: the contents of the answer to the user's utterance are chosen on the basis of the structure of the sentence itself and of the information selected.

  The VisitorManager should have the query-processing module. The query-processing module accepts user's query, extracts various feature vectors from the query, and submits the feature vectors to the visitors. During the feedback step, the query-processing module accepts relevant information to the original query by the users. To record and report information needed to manage a configuration efficiently, we can create a list with the approved configuration identification, the status of the proposed changes to the configuration, and the implementation status of the approved changes.

6.3.2 Navigational Models and ElementAmbassdors

Reinhardt (2003) proposed the ElementAmbassador and Navigator Models. With some modifications, this approach will allow the retrieval of the requested information from the data structures.

Reinhardt (2003) talked about integrated product and process models in his study and he demonstrated their usefulness in handling and storing detailed project management information during construction. Research areas for this topic include construction management and the usage of IT in construction management; research in human computer interaction; and relevant research in the design of data models for construction project management information. A lot of these models are large and difficult to use for engineering and project management tasks. Reinhardt (2003) proposed using the navigational model approach in his prototype software Site Data Collection System (SiDaCoS). SiDaCos has been validated for the project management tasks of progress monitoring and the creation and administration of punch lists, especially for the use on small mobile computing devices on a construction site. Besides punch lists, the approach may also support other project management tasks, such as cost control and document management. SiDaCoS could provide the needed representations that allow efficient interaction with project management data. The system can therefore make mobile computing devices useful tools for project management tasks and eliminate cumbersome and error prone paper-based data access and data collection solutions on construction sites.

A lot of construction tasks require the use of knowledge or experience to make judgments. These knowledge-based tasks require data access and data collection and vary greatly between construction projects. Reinhardt (2003) did cases studies to address data

collection tasks for progress monitoring and for the creation and maintenance of punch lists. Both tasks are project management related and require collecting data and information that is available on a construction site and needs to be processed further in an office (site office or stationary office). The case studies were made with projects that vary in size, type, scope and nature in order to capture a representative collection of aspects that may affect data access and data collection tasks. These tasks depend on a number of variables, such as the type of project, the scale of the project, construction methods, types of materials used, and specifics of the construction contract. For SiDaCoS research, data access and data collection processes are studies for the two project management tasks of (1) progress monitoring and (2) the creation and administration of punch lists (Reinhardt 2003).

According to Reinhardt (2003), the approach for providing the needed visual representations is to create an additional layer of data structures on top of the entities of a product and process model, which provide conceptual representations that correspond to information that a site superintendent is able to observe on a construction site. From these conceptual representations a system can create visual representations of different types, which a site superintendent can use in order to access information from a product and process model or integrate information in such a model. The functionality of the additional layer of data structures is to form the needed conceptual representations by disaggregating and aggregating pieces of information that are contained in entities of a product and process model. In order to provide this functionality, the proposed approach uses the concepts of ElementAmbassadors and Navigational Models.

The ElementAmbassadors are proxies for entities of a product and process model. These ElementAmbassadors can be disaggregated into more detailed ElementAmbassadors and therefore will enhance the degree of detail. Aalami's (1998) proposed a template-based approach for the efficient disaggregation of ElementAmbassadors. This approach uses predefined or user defined disaggregation templates in order to enhance the level of detail of concepts in a product and seven (7) process models. A navigational model is a hierarchical data structure that organizes ElementAmbassadors into sets. By addressing GroupNodes (the higher level nodes in the hierarchy) of a navigational model, the set of subsumed ElementAmbassadors is selected or deselected and data input or data access operations can be performed upon the set of selected ElementAmbassadors. The efficiency of the selection operations with a visual representation of a navigational model mainly depends on whether the user is able select ElementAmbassadors by addressing higher-level nodes in a navigational model.

In order to provide such specific navigational models, Reinhardt (2003) suggested a two step approach:

- Before a site superintendent goes to the construction site, s/he defines navigational models that provide the appropriate conceptual representations which s/he can use on the construction site in order to access information from a product and process model.
- On the construction site, the site superintendent can choose a navigational model from a repository that contains the concepts that he or she can use in order to access or collect data.

But the two-step approach is a compromise to SiDaCoS design and makes the system a semi-automatic system. In addition, this approach requires the site superintendent to be able to define navigational models, which will ask the people to understand the computer program design and data structure of the system. The hypothesis

made by Reinhardt (2003) is that a mobile computing system that implements the navigational model approach can provide the needed representations that support effective and efficient data access and data collection on construction sites. In order to validate this hypothesis, he further developed some research questions:

- What questions need to be answered on a construction site and what categories of information should to be gathered on a site to be processed in the site office or the stationary office for progress monitoring and the creation and administration of punch lists?
- Can navigational models effectively and efficiently support data access and data collection tasks on construction sites?
- Are the navigational model approach and the created software implementation effective and efficient for data access and data collection with mobile computers on construction sites and do they support progress monitoring and the maintenance of punch lists?
- Does the software that incorporates the design created based on previous research question support the two considered project management tasks effectively and efficiently?

Regarding to the types of information that need to be visually represented on a system and the decomposition of that information, Reinhardt (2003) indicated that the information that supports progress monitoring and the creation and administration of punch lists are building elements, activities and punch list items. Different levels of detail were used or were needed in order to capture the scope of an observation. Overall, a system that supports effective and efficient data access and data collection has to provide alternative, user defined decomposition structures for the project and the construction schedule. These decomposition structures should also support the interaction with elements of these decomposition structures at different levels of detail. Tree structure representations appear to be sufficient for visualizing activities, and punch lists should be represented in a list or table view. For building elements, a combination of tree

representations and 2D/3D representations will be needed. Site superintendents need to navigate through tree representations and 2D/3D representations in order to identify and address the elements that they want to select in order to enter data or access data from them.

The operations on the building elements, activities and punch list items are selection operations, data input and retrieval operations, and creation of customized representations. After identifying the building elements, the basic selection operations have to be able to Add, Replace, and Subtract. By using the selection mode control, site superintendents can choose the appropriate selection mode. The available selection modes are the following: (1) single selection mode; (2) multiple-selection mode; (3) single navigational model selection mode; and (4) multiple navigational model selection mode. In cases in which a site superintendent has to re-inspect punch list items, s/he has to retrieve the locations of a specified punch list item. This retrieval operation therefore involves the selection of a punch list item and the execution of the function that highlights the related building elements. Disaggregating and aggregating information that is contained in an integrated product and process model create customized conceptual representations in the form of navigational models. Figures 6-8, 6-9 and 6-10 show the User Interfaces of the SiDaCoS system. Figure 6-11 shows the comparison between different levels of detail regarding the selection (Reinhardt 2003).

The data model, which comprises the product and process model as well as the navigational model framework, is certainly the most complex component of SiDaCoS.

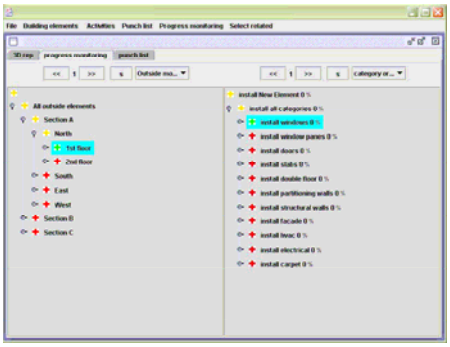


Figure 6-8 User interface of SiDaCoS, Tree view of project structure and activities (Reinhardt 2003: Figure 10)

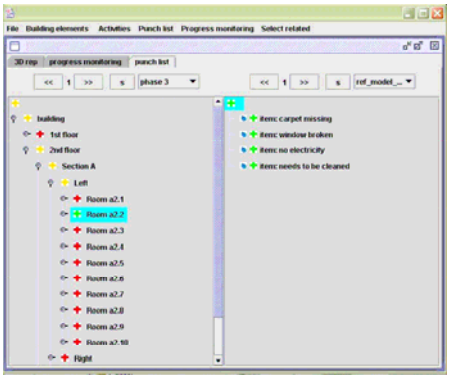


Figure 6-9 User interface of SiDaCoS, Tree view of project structure and punch list items (Reinhardt 2003: Figure 11)

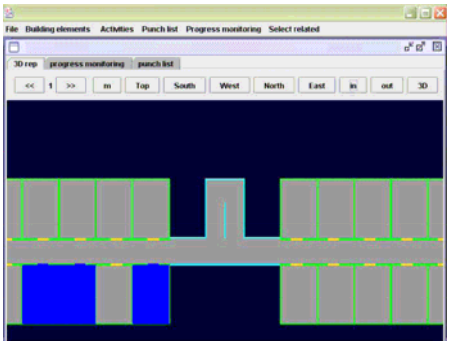


Figure 6-10 User interface of SiDaCoS, 2D view of building project (one floor of a building) (Reinhardt 2003: Figure 12)

The data model represents data (e.g., entities and ElementAmbassadors) and incorporates strong logic and algorithms, such as algorithms for the disaggregation of ElementAmbassadors or the population of navigational models. The data model should be fully encapsulated from the other components of the system. The way of passing

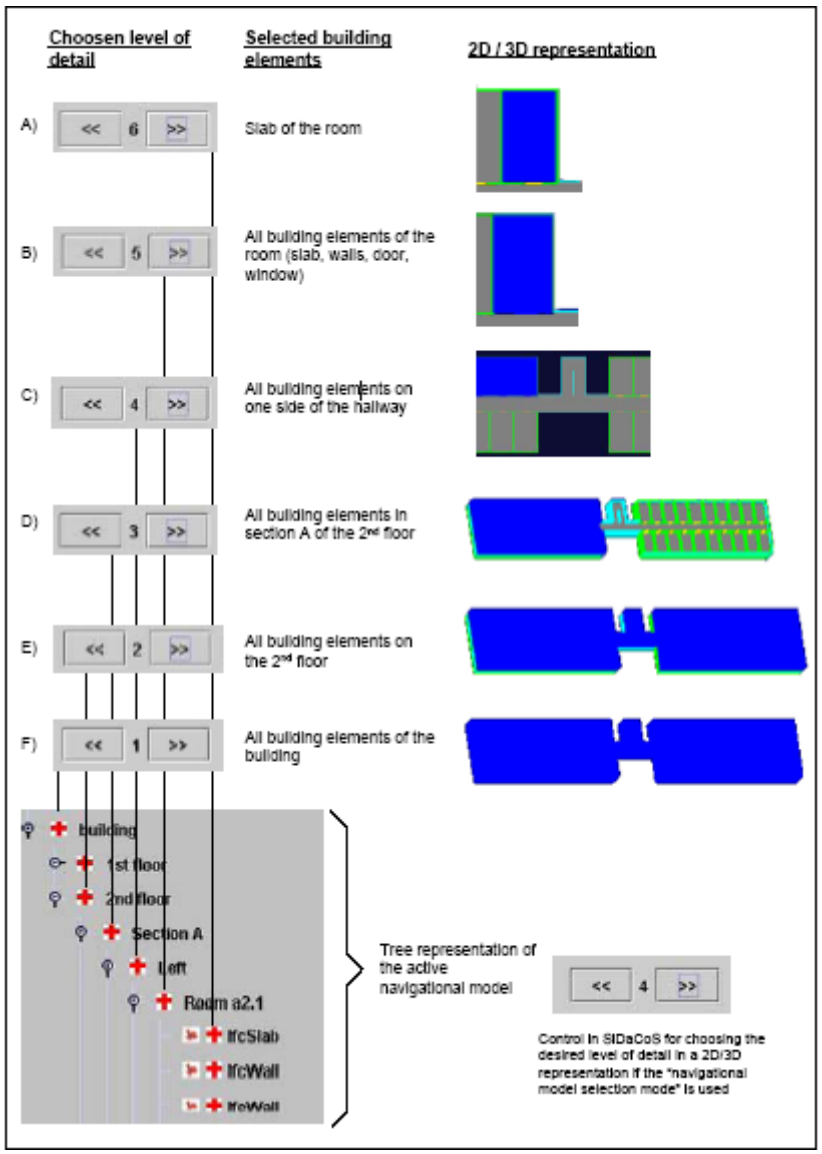


Figure 6-11 Selection operations on different levels of detail, (Using the navigational model selection mode. All selections are made by clicking in the same place (the same slab element; first line of the table) in the 2D/3D representation. The tree representation of the navigational model is not visible in this view, but the active navigational model controls which building elements are selected.) (Reinhardt 2003: Figure 15)

commands and data to SiDaCoS is to use commands in a text-based format. These commands are then translated in the command processor into command-objects that are passed to the appropriate component in the system. Since the navigational models only contain ElementAmbassadors of one type, the overall set of ElementAmbassadors in the system can be divided by type into sets. This reduces the number of comparisons that

have to be performed, and is also true for the Rete Algorithm (Forgy 1982) based retrieval mechanism. The generic element of a disaggregation operation is that relationships between the original (parent) ElementAmbassador and the newly created (child) ElementAmbassadors that have to be maintained in order to propagate data that is entered in the child ElementAmbassadors to the entity that is associated with the parent. Moreover, this relationship is needed in order for the child ElementAmbassadors to access attributes of their parent and the related entity (Reinhardt 2003).

A system that enables the site superintendent to enter status information for multiple building elements or even an entire section (higher level) would need two interconnected navigational models. Such models will have to allow selection in the product domain (select building elements) and in the process domain (select activities). The design of the navigational model framework is shown on Figure 6-12. Similar to the organization of information of different types in an integrated product and process model, information in the navigational model framework is organized in domains.

6.3.3 Comparison between the Configurable Visitor Design and the Navigational Model and ElementAmbassadors

Based on the descriptions of both the Configurable Visitor Design and the Navigational Model and ElementAmbassadors proposed by Reinhardt, a comparison can be made between these two design ideas. The first one is designed to fit tree type data structure and can traverse the whole structure. For the Visitor Pattern Design, the underlying data structure only needs to have a simple embedded accept method and the AbstractVisitor's name, a bridge will be built up between the underlying project data structure and the information retrieval process. In this version of the Visitor pattern, the AbstractElement objects determine which elements of an object structure a Visitor object

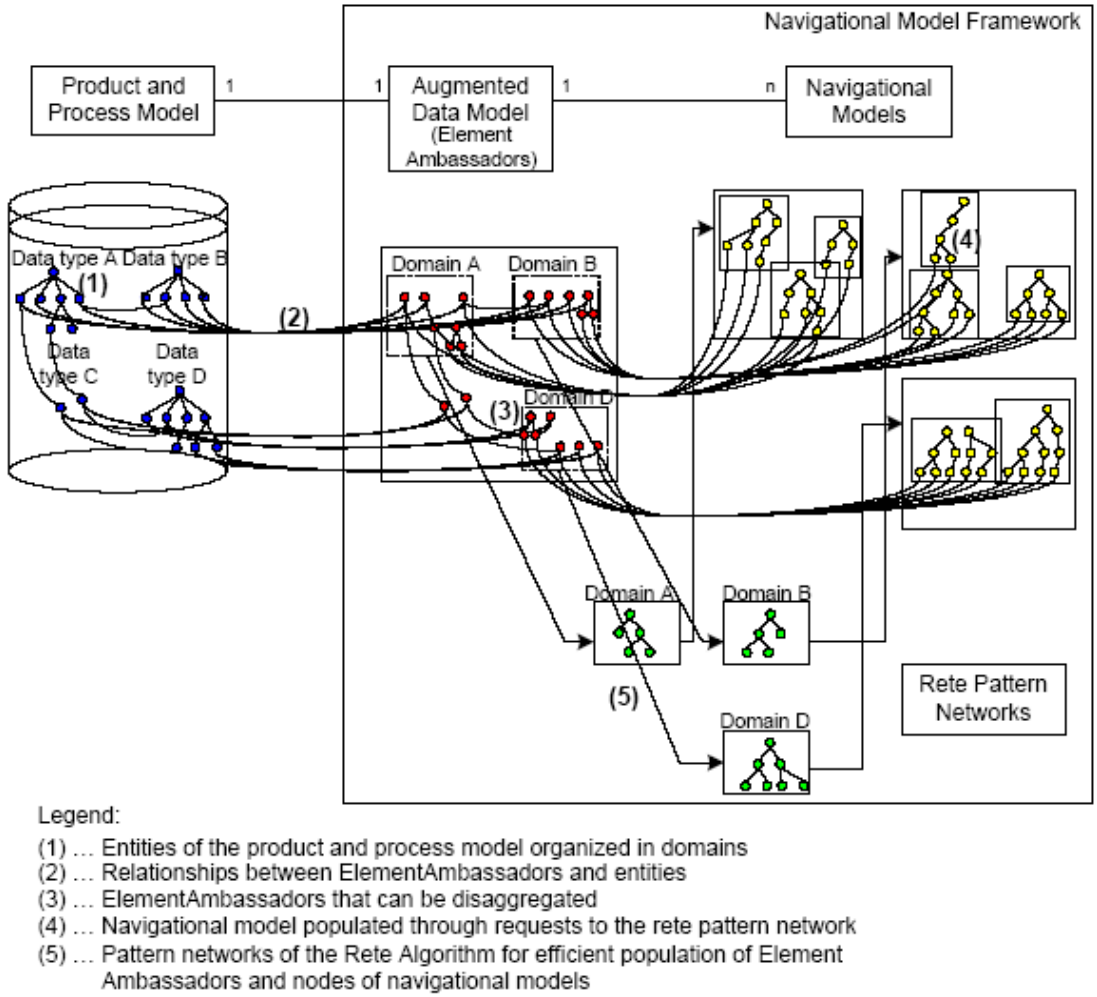


Figure 6-12 The design of the navigational model framework (Reinhardt 2003: Figure 24)

visits and the order in which it visits them. The Visitor classes do not need to make assumptions about the structure of an object structure in order to navigate through it, so they are reusable. It has the advantage of keeping Visitor classes independent of the structure of the object structure. However, one drawback of this approach is when the Object structures are very large and the execution time may become too long for a Visitor to visit every object when it needs to visit only a small subset of the objects in the structure. But it takes less effort to maintain the code because of the minimal of the dependency that RealVisitor objects have on the object structure. The Navigational Model proposed by Reinhardt depends on the structure of the ElementAmbassadors.

Moreover, depending on different types of data, the ElementAmbassadors have to build different structures to access them. As indicated by Reinhardt (2003, p. 70) each entity of a product and process model is associated with one ElementAmbassador. Thus for a system with multiple types of data, various ElementAmbassadors need to be developed to access them and multiple Navigational Models are needed to retrieve information. The Navigational Model in Reinhardt's research has the same role as the proposed Visitors in the Configurable Visitor Design of this study. Both of them connect the operations with the underlying data structure and go through the data structure to find the right information. But the difference between Navigational Model and the Visitors is that the visitors do depend on the data structures but the Navigational Model does. Detailed comparisons are listed in Table 6-1.

6.3.4 Connection between the Project Database and the Proposed System

For the design of the User Model Driven System, the primary research focus is on the extensible user model architecture. In order to make the system complete and validate the design, a structure is needed to connect the outcome of the extensible user model driven architecture with the underlying database of construction project data. The UMDA system acts as a middleware in between the User Interface and the construction project database. Obviously, different DBMS of the construction project database will provide different performance, which will affect the performance of the UMDA system. Both of the systems listed in Table 6-1 are good candidates for connecting the project database for the proposed extensible user model driven architecture. The Navigational Model and the ElementAmbassadors have been implemented simply because there is detailed documentation made available by Reinhardt (2003) and they require less programming time to develop. However, Reinhardt's system is not available for use at this time.

Table 6-1 Comparison between the proposed Configurable Visitor Design and Reinhardt's (2003) Navigational Model and ElementAmbassadors

	Configurable Visitor Design	Navigational Model and ElementAmbassadors
Designed to fit tree type data structure	Yes	Yes
Underlying data structure	General tree type data structure with accept method and the AbstractVisitor's name	Different data types are disaggregated and augmented by ElementAmbassadors and restructured by Navigational Models
Number of different types of data structures	One	Multiple, depending on how many types of data.
Dependency of operations on data structures	No. Visitor Manager generate visitors depending on the query type not on the underlying data structure	Yes. The Navigational Model must have different types of structures for different data models.
Computation Efficiency	Consume longer computation time because the visitors need to go through the structure to find the intended information	Consume larger disk and cache space because both ElementAmbassadors and Navigational Models need to be loaded into memory; the structures of both have duplicated information.
Easy to maintain	Very easy to maintain, because it has the following classes: Visitor, AbstractElement, RealElement, RealVisitor, VisitManager,& RootNodeClass. Each class is designed in encapsulation. Code changes in one class will not affect others.	Relative hard. Navigational Model is depending on ElementAmbassadors. In case of change in code, consistency must be maintained.
Data Redundancy	No redundancy.	Has data redundancy.
Construction Project types the system can support	All types of construction projects.	All types of construction projects. Primary design target is to help with progress monitoring and the creation and administration of punch lists.
Selection modes	Supports structured selection modes, e.g. single selection or multiple-selection according to different tree or 2D/3D views, and unstructured selection.	Support structured selection modes are the following: (1) single selection; (2) multiple-selection; (3) single navigational model selection; and (4) multiple navigational model selection.
Program intensiveness in design	Relative heavy in programming. Requires a lot of code.	The program load in developing the ElementAmbassadors and Navigational Models is moderate.

Adequate modifications need to be applied to the Navigational Models so that the models can accept the commands generated by the proposed Configurable Visitor system. The UMDA system will accommodate the existing structure of the ElementAmbassadors. The ElementAmbassadors are proxies for the entities of a product and process model that can access all the information contained in the entities, but also provide additional functionalities. ElementAmbassadors provide functions to support the selection of conceptual representations through multiple visual representations. The selection operation provided by ElementAmbassdors is the primary one we need for the UMDA system. Each ElementAmbassador can only be disaggregated into child-ElementAmbassadors of the same type. They can access information that is contained in entities and write back information that is passed to them by the entities of the product and process model. In cases where the ElementAmbassadors are disaggregated and the product and process model does not provide the needed data fields for the more detailed information, this information can be stored in attribute-value pairs in the ElementAmbassador. Moreover, specialized sub classes of ElementAmbassadors can provide specific fields and constructs in order to model this information. A sub class of an ElementAmbassador, for instance, that represents a wall can have specific data fields for the geometry of the disaggregated element.

The function of the navigational models is to organize the large amount of ElementAmbassadors into select sets of ElementAmbassadors. For the proposed User Model Driven Architecture, the navigation models need to be modified. Navigational models are tree structures with two types of nodes. GroupNodes are usually not leaf nodes and are labeled. Moreover, each GroupNode contains filtering criteria that the

ElementAmbassadors have to satisfy if they should be grouped under that particular node. With that modification, both the leaf nodes and GroupNodes in the navigational models will be able to accept the commands generated by the User Model Driven Architecture and to retrieve the required information.

The algorithm matching between the ElementAmbassadors and the navigational models is called the Rete Algorithm. The Rete Algorithm trades off computation with memory consumption by caching and reusing information rather than re-computing it in every cycle of the matching process. The entire data model (product and process model, augmented data mode and the navigational models) is held in the working memory. In order to make the data model persistent, all elements of the data model are written to the permanent storage media in one transaction. For the proposed User Model Driven Architecture, the Rete Algorithm works in between the ElementAmbassdors and Navigational Models. The rules in the Rete Algorithm decide which leaf node in Navigational Models points to which node in the ElementAmbassdors. There are other algorithms that can perform the same functions as the Rete Algorithm. But the Rete Algorithm has the advantage in saving computing time with the reduced cost of using memory. For most current PCs, the memory space limitation is not a big issue. For detailed information regarding the Rete Algorithm, please refer to Appendix B.

In general, only the Navigational Models need to be modified. To create a navigational model, the system asks for a structure of the navigational model that supports the desired tasks that have to be done on the construction site. This task is done by the Task Analysis Module of the proposed User Model Driven Architecture. The Task Analysis Module can also provide information for the filtering criteria and population

policies for each node in the navigational model. After the criteria have been set, the ElementAmbassadors can be distributed to the nodes of the navigational model.

In general, the Navigational Models set up the relationships for the ElementAmbassadors and uses a relational database for the retrieval of the ElementAmbassadors. Due to the optimized abilities of the relational database to perform searches on different dimensions, the retrieval mechanism is efficient. However, the major drawbacks of this solution are the large amount of redundant information that must be maintained and the extent and complexity of the procedures that are needed in order to keep the redundant data sets synchronized.

Reinhardt's (2003) SiDaCoS is prototype software that constitutes a reference implementation of the navigational model framework. In addition to the flexibility of representations provided through the navigational model framework (type of representation, level of detail, structure of the project decomposition), SiDaCoS is flexible with respect the product and process models that are used. SiDaCoS provides representations that allow the users to enter information that they are able to observe on a construction site into the IT-system without having to perform conversion operations, enter lengthy descriptions of observations, or perform inefficient, repetitive, low level selection operations.

6.4 Discussion of the User Model GUI Configuration

There are two options to configuring the User Model GUI, one is to configure the user-input interface so that different users have different layout of the screen to operate; the other is to configure the system-output interface which will give answers with different levels of detail to different people. The latter option has much more practical significance than the former one. If a system provides different user-input interfaces for

different people, the users will get confused. For example, two users may install the same system on their computers. They learn about how to operate the system together. If they find out that they have different interfaces and operations, they will doubt the stability of the system and hesitate to use it. As analyzed in Chapter 5, different user output interfaces will help the users save time in scanning through all the data trying to locate the useful information they seek. Without User-Model-Driven Architecture, the answers to users' queries may be represented in a semi-structured way and grouped under different databases. Within each group, the order of the results reflects the order of the records stored in the databases. As mentioned earlier, it is very time-consuming for the users to find the meaningful information by themselves.

One way to help people configure the system-output interface is to set up sorting conditions and arrange the order of appearance of the results from the most relevant to the least relevant. Without the User Model Driven Architecture, this method will end up using some static sorting rules and the order of the results will not change no matter who logged in and for what task. In the proposed User Model Driven Architecture, the GUI will be configured based on the user group and the system will present to the user with what s/he needs. The Interactive User Interface will also provide the user with the ability to access more detailed information.

6.5 Information Retrieval

In Information Retrieval, user models have been limited to lists of terms relevant to an information need. The list is usually very short for ad hoc querying and longer for information filtering tasks. Information systems that could benefit from having a user model should be able to adapt to individual users, to learn about their preferences and attitudes during the interaction (to construct a user profile), and memorize them for later

use. Moreover, these user profiles will represent a starting point for the creation of user communities based on shared interests or goals. Further, the system should be able to update its model as a user changes interests.

Information Retrieval (IR) is concerned with the study of systems for representing, organizing, retrieving and delivering information based on content. User modeling is the glue. The better we model users, the better we can satisfy their information needs.

The primary questions that need to be addressed are as follows:

- How can Information Retrieval techniques be applied to acquire and continuously adapt user models?
- How can user-adaptive IR systems be applied? Is it based on effective retrieval, user experience, reaction and satisfaction?

Based on common interests, and background, different use profiles can be set up and when a user logs in, the user will be assigned a particular profile and active configuration file.

6.6 Relationship of the Proposed Research with Human-Computer Interaction

Human-computer interaction (HCI) studies the interactions and the relationships between humans and computers. HCI is more than user interfaces; it is a multidisciplinary field covering many areas (Helander et. al. 1997). In the first ten to fifteen years of its history, HCI has focused on interfaces (particularly on the possibilities and design criteria for graphical user interfaces (GUIs) using windows, icons, menus, and pointing devices (WIMPs)) to create more usable systems. As interface problems were better understood, the primary HCI concerns started to shift beyond the interface. More recent HCI research objectives (Fischer 1993a) are concerned with tasks, with shared understanding, and with explanations, justifications, and argumentation about actions, and not just with interfaces. The new essential challenges are improving the way people

use computers to work, think, communicate, learn, critique, explain, argue, debate, observe, decide, calculate, simulate, and design (Fischer 2000).

As Suchman (1987) put it, human-computer collaboration can be approached from two different perspectives: an emulation approach and a complementing approach. The emulation approach is based on the metaphor that to improve human-computer collaboration is to endow computers with “human-like abilities”. The complementing approach is based on the fact that computers are not human and that human-centered design should exploit the asymmetry of human and computer by developing new interaction and collaboration possibilities. User modeling shows the potential to improve the collaborative nature of human-computer systems.

6.7 Relationship of the Proposed Research with Database Schema Design

Information architects determine the best path through the terrain, whereas interaction designers place the signs and draw the maps. For Web sites, and other information management products, there are three general categories of work when creating a user experience:

- Information architecture is the process of creating an underlying organization system for information the product is trying to convey
- Interaction design is the way that structure is presented to its users
- Identity design amplifies the product’s personality and attraction

The hypothesis that the user's intention is a subset of the system's knowledge base is clearly quite restrictive, and in general it would be more realistic to suppose that the user's and the system's world models intersect in some way. Moreover, it is possible that the user wants to execute some actions that are beyond the capability of the system. In this case, the system is supposed to be in some way cooperative in finding useful information.

Chapter 7 will discuss the conceptual design of the system architecture based on the analysis of system requirements in Chapter 6. Chapter 7 will also discuss how to search for the needed information for the users and to present the information in a way that the users will get to the requested information first without wasting time reading through all redundant data. The proposed User Model Driven Architecture (UMDA) will act as a middleware in between the user interface and the underlying data management system.

CHAPTER 7
CONCEPTUAL DESIGN OF USER MODEL DRIVEN ARCHITECTURE AND POSSIBLE APPLICATIONS

In Chapter 6, case studies were analyzed and the system requirements for construction management were generated from the study of the construction project data and users activities. In addition, some design ideas regarding the user interfaces which reflect the function and system requirements were presented. The following content will discuss the conceptual design of the User Model Driven Architecture (UMDA) system. The primary operation of the User Model Driven Architecture (UMDA) system is to search for the information needed by the users and to present the information in a way that the users will get to the requested information first without wasting time reading through all the redundant data. The proposed UMDA system will act as a middleware in between the user interface and the underlying data management system. As described in Chapter 1 and further explained in Chapter 2, the UMDA system is an adaptable system, which is controlled by users. To distinguish the UMDA system from a filter system, the existing filter systems and the importance of adaptive information retrieval will be discussed in Section 7.1. Another feature of the UMDA system is its ability to be configured. Section 7.2 discusses how to implement the adaptive and configurable features and their influences on the entire information retrieval system. In order to find a suitable connection between the proposed User Model Driven Architecture and the data management system two options were compared in Chapter 6: the Visitor Design Pattern and the ElementAmbassadors with Navigational Models. Both of these choices are

appropriate for the proposed UMDA and they have different advantages as well as shortcomings. The ElementAmbassadors and Navigational Models option was selected for implementation because it requires relatively simple programming.

7.1 Adaptive Information Retrieval

With all the system requirements set, the process of building the UMDA system based on the conceptual design can be started. As described by Sugiyama et al. (2004), researchers are realizing that search results should be adapted to users with different information needs. Sugiyama et al. (2004) first described hyperlink-based personalized web searches (i.e. Google, and the CLEVER project). To address several problems with these engines, for example, (1) the weight of a Web page has to be defined, and (2) the relative importance of the contents among hyperlinked Web pages is not considered. To personalize web sites, link topology and the structure and contents of Web pages are often used in their development. Two approaches are widely used: Link Personalization and Content Personalization. Link Personalization involves selecting the links that are more relevant to the user and changing the original navigation space by reducing or improving the relationships between Web pages. E-commerce applications (i.e. Amazon.com) use link personalization to recommend items based on the buying history of clients or some categorization of clients based on ratings and opinions. Users who give similar ratings to similar objects are presumed to have similar preferences, so when a user seeks recommendations about a certain product, the website will suggest those products that are most popular for his/her class or those that best correlate with the given product for that class. Another approach, content personalization (i.e. My Yahoo! or My Netscape), is done when pages present different information to different users. The difference between this and “Link Personalization” is subtle because part of the contents

(i.e., the link anchors[4]) presents different information when links are personalized. However, content personalization is referred to when substantial content or information in a Web page is personalized, unlike link anchors. An example is an interactive personalized newspaper on the WWW that allows for interactive personalization, browsing and layout control. Personalizing web sites can also be done by recommender systems. Recommender systems collect user feedback in the form of ratings for items in a given domain and exploit similarities and differences among profiles of several users in determining how to recommend an item. There are two prevalent approaches to constructing recommender systems: collaborative filtering-based and content-based recommendation. Collaborative filtering means that people collaborate to help one another perform filtering by recording their reactions to documents they read. A content-based approach provides recommendations by comparing representations of content contained in an item with representations of content that the user is interested in. In this approach, a model of user ratings is first developed. Algorithms in this category use probabilities and envision the collaborative filtering process by computing the expected value of a user prediction given the user's ratings of other items. The model building process is performed by three different machine learning algorithms: (1) Bayesian network, (2) clustering, and (3) rule-based models. Sugiyama's approach proposes that the search system should directly and exactly capture the changes in each user's preferences without any user effort in order to provide more relevant information for each

[4] Anchor is also known as span, region, button, or extent. It is an area within the content of a hypertext node (e.g. a web page), which is the source or destination of a link. A source anchor may be a word, phrase, image, or possibly the whole node. A destination anchor may be a whole node or some position within the node. Typically, clicking with the mouse on a source anchor causes the link to be followed and the anchor at the opposite end of the link to be displayed. Anchors are highlighted in some way (either always, or when the mouse is over them), or they may be marked by a special symbol. (Computing-Dictionary, 2004)

user. Sugiyama's approach is novel because it allows each user to perform a fine-grained search by capturing the changes in each user's preferences without any user effort (Sugiyama et al. 2004).

Most of these systems use the recommendation or the filtering approach to alleviate the information overload on the users. These systems provide personalized suggestions based on user preferences, either by explicit indication by the user, inferred (semi-) automatically from his/her profile, or from his/her navigation activity. The collaborative filtering process is one of current research trends in the recommendation systems, which can provide a certain level of expert advice in a certain domain. But in general, these personalized systems are not domain specific, which in turn reflects their Web-based nature.

However, the goal is to build a system for information retrieval in construction management. Whether or not this system is able to connect to the Internet is not the primary concern of the proposed design. Although this system is a "domain-specific" information retrieval system, with the degree of integration between different databases getting deeper, users of this system will easily be overloaded. To build a model recommending or filtering the retrieved information based solely on user profiles will not entirely satisfy the users' needs. As shown in the previous case studies, not only do different users have different needs, but also the same user performing different tasks will have different needs.

7.2 User Model Driven Architecture Discussion

The UMDA system will operate according to user needs. To build up this User Model Driven Architecture user models have to be developed. Based on the observation and analysis of the case studies done in Chapter 5, it has been determined that at the same

time, for the same project, with the same question, different users have different expectations and requirements for their search results. With the large amount of information and the multiple integrated databases, providing different users with the same list of all possible answers is not a very plausible idea. The user models have to reflect the information for the user and the information for the task that the user is working on. In order to fulfill these functions, the targeted system must be adaptable and configurable to user needs. The following sections will first compare the differences between customization and configuration. Then the requirements for the user model to configure the application layer of a system will be explained. Following that, the author will discuss the system's graphical user interfaces.

7.2.1 Comparison of Customization and Configuration

In the UMDA system, the core part of user modeling is stored in the User and Task Profiles databases. The Task Analysis Module will activate the User Profiles Database when the user logs in and it will activate the Task Profiles Database when the Task Process Queue is received.

Some vendors offer to "customize" the solution, while others speak of their solution as being "configurable". What is the difference between these two terms, and is one approach inherently preferable? Customizing software means, basically, writing new programming code. The advantage of custom software is that in theory you get exactly what meets your requirements. However, the pitfalls of custom software are many. For example:

- Customization adds cost, both for the initial development, but also for long-term support.
- Customization can add delays to the implementation, for code to be written and thoroughly tested. Even with careful project management, there is the possibility of

the project's slipping due to scope creep and other factors, and ultimately missing its deadline. If quality assurance is undercut in order for the project to come in on time, the result can be buggy software.

- Vendors that customize their solutions for individual customers commit significant resources away from the rest of their customer base.

The most serious, and often overlooked, disadvantage to customization is that the new code must be re-written each time the customer wants to upgrade to a new version of the vendor's solution. On the other hand, configuration is the opposite of customization. Configurable software has the flexibility to adapt and customize the system to fit a company's particular process through the extensive use of switches built-in the system. A configurable solution will, for example, allow a company to tailor the Careers section interface for the recruitment process by using different fonts, colors and layouts. The Configuration may occur at implementation, and during steady-state use through administrative control panels. Some of the key benefits of configurable solutions stem from the fact that all of the vendor's customers will be running identical programming codes.

- It is efficient for the vendor to support only one version of the software, which means better support for all customers.
- Quality assurance remains focused on one version, which means that the system is more stable and reliable.
- The ultimate benefit to the customer is tremendous flexibility to customize the system to internal processes, without ending up with a version that is so customized that it can no longer scale or upgrade to the next version seamlessly.

With the configuration strategy, though, requests by a customer for enhancement or additions of new functionality have to have broad appeal and applicability to the vendor's overall customer base. Configurable solutions allow for better match to needs without the disadvantages of customization. Large organizations especially have a great deal of

variability in their staffing processes. As recruiting becomes more sophisticated, enterprises are moving from mass recruiting to segmentation. Configurable software can support the need for segmentation at the enterprise level.

Section 2.6.2 compared the differences between adaptive and adaptable systems. Adaptive systems allow the dynamic adaptation by the system itself to the current task and the current user. Configurable systems are very similar to adaptive systems. One difference between adaptive systems and configurable systems is that adaptive systems require little (or no) effort by the user and no special knowledge by the user. The proposed configurable system will allow for user interactions. Section 7.4 will discuss the details of the UMDA system architecture.

7.2.2 User Model Configures the Application Layer

When a user submits a query to the system, the user may not know how many databases need to be accessed to retrieve information. The simplest way is to open all the databases and get information from all of them. It is relatively easy to implement on the software design side and a search result will include all possible answers. But on the user's side, few users will have the patience to read through all the records in the list in trying to find out the useful information for themselves. For example, when a system disassembles the query "Show me all possible work in next 2 weeks" into tokens and finds out none of the tokens can be replaced by the terminologies, the system will pass all the tokens to the Task Analysis Module and the Module will compare the tokens with the data in the task profiles. All the useful tokens will be sent out to the ElementAmbassador and the Modified Navigator with their attributes requirements, which specify the databases the user needs to open. This process represents the configuration of the application layer.

Chapter 6 discussed ElementAmbassadors and Modified Navigators in detail. The proposed UMDA system will modify the ElementAmbassadors and Navigator Models so that the requested information can be retrieved from the data structures.

7.2.3 User Model Configures Graphical User Interface

The Graphical User Interface (GUI) will be configured by the User Model. Without the User Model Driven Architecture, the entire answer set to the users' queries may be presented to that user. One way to organize the answer set is to follow the categories from each of the different databases. For the results grouped under different databases, within each group, the order of the results reflects the order of the records stored in the databases. This is a semi-structured way to present data to users. As discussed previously, it is very time consuming for the individual user to find the meaningful information.

Another way to present data to users is to set up sorting conditions and arrange the appearance order of the results from the most relevant to the least relevant. Without the User Model Driven Architecture, this method will end up using some static sorting rules and the display order of the results will not change no matter who logged in and for what task. From the analysis of the case studies in Chapter 5, it can be seen that the set of rules will not be fixed or static. The system must provide the corresponding set of rules if the user or the task changes.

The third way to present data uses the User Model Driven Architecture (UMDA) system. In the UMDA system, the GUI has been configured. The system will present to the user with what s/he needs. The Interactive User Interface will also provide the user with the ability to extend to more detailed information. The comparisons between these three ways of presenting data to users are listed in Table 7-1. Table 7-1 shows one of the important factors in GUI design. User usually likes to read the results in a certain order.

Table 7-1 Comparisons of data presentations

	Semi-structured way	Sorted results	UMDA GUI
Programming	Doesn't require too much programming	Needs to implement sort algorithms	Needs to implement UMDA and GUI
System Computation Time	Very fast, involves very little computation	Very long. The computation time spent on sorting algorithm will be tremendous.	Acceptable. The UMDA system needs to build up the structure and analyze the queries.
Length of final searched results	Very long. It lists all the related information and let user to select what s/he needs.	Very long. Although the results are sorted according to their pertinence, the amount of data does not change.	Relatively short. The useful information is presented to user.
Time spent by user to locate the useful information	Very long. In worst-case scenario, user needs to go through all the results and find the answer.	Various. Since it uses a static or fixed set of rules, the searched results are not always sorted according to user's needs. On average, its performance time is shorter than the semi-structured way. But the worst-case scenario still exists. User may have to read the entire results to find the answer.	Short. The rules to determine the order of the data are flexible and according to user model.

Thus designers could provide some ways of presenting data to users and according to their preferences, user could configure the user interface and decide the result order. Semi-structured way lists all the related information and let user to select what s/he needs. Sorted results could sort results according to their pertinence but it needs long computation time. In UMDA GUI, the rules to determine the order of data are flexible and are according to user model. The computation time of UMDA GUI is relatively short.

7.3 Explanation of the Proposed System Architecture

The next step for the conceptual design of a system is to establish the entire structure. Figure 7-1 shows the proposed UMDA system architecture. Before explaining each of the components of the system architecture the aspects needed to describe its components are defined as follows:

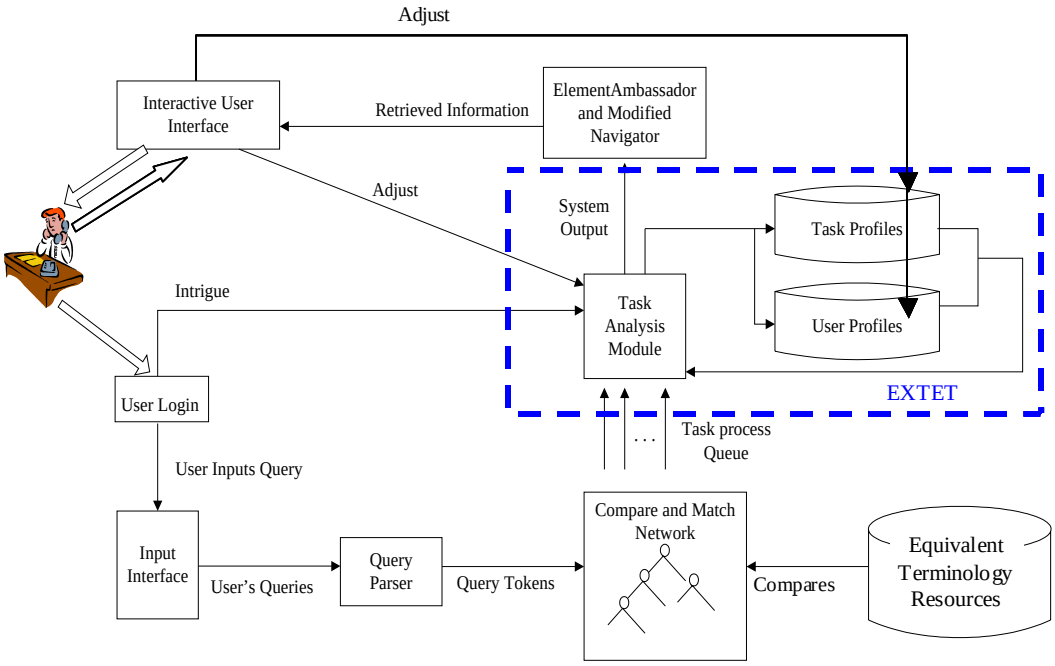


Figure 7-1 Explanation of the proposed system architecture

- General Description: describing the component;
- Requirements: Things that the component must allow the user to do;
- Constraints: Rules about what can and cannot be done. This includes the following:
  - Pre-conditions that must be true before the procedure is run;
  - Post-conditions that must be true once the procedure is run;
  - Invariants: these are always true, e.g. a purchase order must always have an order number.
- Scenarios: Sequential descriptions of the steps taken to carry out the procedure. May include multiple scenarios, to cater for exceptional circumstances and alternate processing paths;

- Scenario diagrams (optional): Sequence diagrams to depict the workflow (similar to Scenarios but graphically portrayed).

- Additional attributes such as implementation phase, version number, complexity rating, stereotype and status

The above definitions are for the formal specifications of a computer system. With those terms defined, the conceptual design of the User Model Driven Architecture (UMDA) will be as follows:

Function 1: User Login:

General Description: This represents the activity that a user (i.e. superintendent, project manager, etc.) performs in logging into the system.
Requirement: Log User into the system.
Constraints:

Pre-condition: None.
Invariants: User Name, User ID, Project Name
Post-condition: None

Scenarios:

(1) Log the user into the system
(2) Authorization check
(3) Save the user information and sent to Task Analysis Module

Additional attributes: None

Function 2: Input Interface:

General Description: General User Interface between the user and the system. Asks the user to input the queries or select functions to fulfill.
Requirement: Provide space for user input.
Constraints:

Pre-condition: None.
Invariants: User's query.
Post-condition: User's query not empty.

Scenarios:

(1) If user types in a query, send it to query parser.
(2) If the query is empty and user asks to search, respond with "What would you like to do?" message.

Additional attributes: None

Function 3: User's Queries:

General Description: The user's queries may not be formatted, in other words, they may not be recognizable by the system. For current system development, only text-based queries are acceptable by the system.

Function 4: Query Parser:

General Description: The parser works as a disassembler. It breaks down the user's query into single words (query tokens).
Requirement: Parse the query into tokens.
Constraints:
Pre-condition: query not empty
Invariants: None
Post-condition: tokens not empty
Scenarios:
(1) If the user query has only one word, send it to Compare and Match Network.
(2) If the user query is long than 256 characters, ask the user to limit the query length and search the answer for the rest of the query later.
(3) Words that appear repeatedly are treated as one word.
Additional attributes: None

Function 5: Equivalent Terminology Resources:
General Description: This part works as a dictionary. It is a database that stores all the A/E/C related terms and thesauruses. Since these are equivalent terms the system does not have to recycle through the Equivalent Terminology Resources module to change back to the precise terminology that the user had in their query.

Requirement: Add, delete, replace, search, and other basic database operations.
Constraints: Pre-condition: Database not empty
Invariants: None
Post-condition: Database not empty
Scenarios: (Reference all the basic database operations)
Additional attributes: version number and status

Function 6: Compare and Match Network:
General Description: All the query tokens will be compared with the terms and thesauruses from Terminology Resources by the Compare and Match Network. The final result is system recognizable and sent to Task Analysis Module.
Requirement: Substitute the wrong or inaccurate words in the user's query
Constraints: Pre-condition: user's query not empty && terminology database not empty
Invariants: None
Post-condition: generate token queue
Scenarios:
(1) If every token matches the terminologies in the database, then proceed with the token queue.
(2) If there are wrong words in the tokens, insert the suggested substitutions into the token queue.
(3) For characters that the Network unable to match, proceed with the original tokens.
Additional attributes: None.

Function 7: Task Profile Database:

General Description: This database stores all the task templates. It has the scalability to add in new task profiles. It is also extensible into more detailed tasks. In information-seeking dialogues, the user (i.e. construction project manager or supplier) often asks questions about specific aspects of the tasks s/he wants to perform. But most of the times, it is difficult for the software systems to unambiguously understand her/his overall intentions. The Task Profile Database will help the system to locate the information the user wants.
Requirement: Provide task profile to Task Analysis Module
Constraints:

Pre-condition: User logged into the system
Invariants: User is a company employee
Post-condition: task profile

Scenarios: Based on case studies, observations, and other literatures, build up the task profiles for construction process or building lifetime. When the user logs in, generate the task profile.
Additional attributes: None.

Function 8: User Profile Database:

General Description: This database stores all the profiles about user groups of construction people. It is extensible into single person profile database. As shown in Figure 7-2, the complexity of a query depends on two parameters: number of databases it needs to retrieve information from and how much detail the user wants to know. Using these two parameters as two axes, a matrix as shown in Figure 7-2 can be built. In practice, when a Superintendent asks a query to the system, s/he will need very detailed information from one or two databases, e.g. CAD drawings and Scheduling. An estimator may not need to know such detailed information as what the Superintendent requires, but s/he will still need to access CAD drawings and the subcontractor database. The company President usually cares about the project cost and schedule. The rest of the information will be redundant to him/her. When a superintendent wants to find out information from the integrated system, s/he needs to have highly detailed information from 3 to 4 databases. Too much information provided to any of these individuals may confuse them and delay their decision. Thus, while a Project Manager may have to access multiple databases and retrieve very detailed information, the Senior PM, acting as a consultant, may only need to know certain general information from these multiple databases. The granularity of the information needed is shown in Figure 7-2.
Requirement: Provide user profile to Task Analysis Module
Constraints:

Pre-condition: User logged into the system
Invariants: User is a company employee
Post-condition: Generate user profile

Scenarios: Divide users into different user groups and build up profile for each group. When the user logs in, generate the specific user profile.
Additional attributes: None.

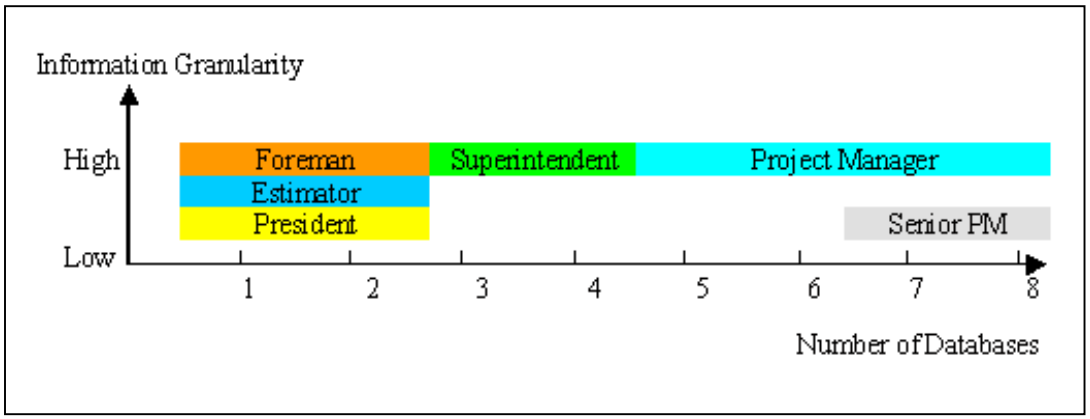


Figure 7-2 Complexity of a query depending on the Databases to be accessed

Function 9: Task Analysis Module:

General Description: The information collected when user logged in, the profiles from Task Profile Database and User Profile Database, and the match results from Compare and Match Network are all sent to Task Analysis Module.

Requirement: Task Analysis Module works like an information binder. All the tokens in token queue will be packed with the attributes generated from User Profiles and Task Profiles Databases.

Constraints:

Pre-condition: Token queue not empty && Task profile not empty && User profile not empty

Invariants: The user belongs to one of the user groups

Post-condition: Generate system output (token queue) with the attributes generated from the User Profiles and Task Profiles Databases.

Scenarios:

(1) After the Login process has checked the user's identification and approved the user login, the System will invoke the Task Analysis Module, which will store the user information and pass it to User Profiles Database.

(2) The token queue from the Compare and Match Network will be sent to Task Profiles Database to find out what are the databases that need to be searched.

(3) Pack the two attributes to each of the tokens and send them to the ElementAmbassador and Modified Navigator.

Additional attributes: None.

Function 10: ElementAmbassador and Modified Navigator:

General Description: For details see Reinhardt (2003). The Modified Navigator here is different from what is described in Reinhardt's work. It will have certain functions embedded in every group node (non-leaf node) generated. This way, the Reinhardt's model can be used to retrieve information.

Function 11: Interactive User Interface:

General Description: The Interactive User Interface will provide the user the ability to extend to more detailed information from other integrated databases.

In the UMDA system, user models will be implemented to configure the system environment. User interfaces will be configurable and different groups of users will have different interfaces with the system. For example, the system administrator will be able to maintain the system and databases. Their work will focus on the system functions and system stability. For the construction company staff, their interests are in the construction projects. The interfaces for them will be different from that for the system administrator. Besides user interfaces, system applications will also be configurable. Use the search operation as an example. The UMDA system will generate different commands for various types of users and the searched results will be presented to users according to their needs. In addition, different configurable operations or system applications will be applied to different users. For example, in construction companies, Project Managers are responsible for a lot of paper work. They need to write project memos, generate faxes and transmittals to subcontractors, architects, engineers, owners, and other people participated in the construction project, write and maintain purchase orders, propose change order requests, ask for information, and all other office work. Instead, Superintendents worry less about the paper work, except for their field reports. After the field report has been generated, it will seldom be changed. So a Project Manager will need the edit function of the system to make changes on the documents or keep track of the documents. A Superintendent will ask for review of the documents most of the time. In this case, the Project Manager and Superintendent have different requirements from applications provided by the system.

With all the system requirements set, the process of building the UMDA system based on the conceptual design can be started. Chapter 8 will discuss in detail the steps used to implement the conceptual design. Some sample codes and algorithms for the system will also be shown in Chapter 8.

CHAPTER 8
SYSTEM ARCHITECTURE IMPLEMENTATION

This chapter will discuss the implementation of the system architecture of the User Model Driven Architecture (UMDA) system. The following contents relate to Chapter 7, which is about the conceptual design of the UMDA system. Section 8.1 and 8.2 in Chapter 8 include the process of implementing the conceptual design of the UMDA system. Section 8.3 includes all the 6 parts of the UMDA system as shown in Figure 7-1 in Chapter 7. The contents in Section 8.3 is organized according to how the entire system workings. Section 8.4 discusses the interesting aspects of the implementation, and Section 8.5 is a summary of the previous sections. The organization of the contents in Chapter 7 and Chapter 8 shows that this study targets the construction IT field. It reflects how the ideas are generated and the steps involved in the process from surveys, to use cases, formal use cases, to information retrieved from user profiles and task profiles and then to configuring the entire system. The current organization connects actual construction industry practice with IT programming.

8.1 Building up Use Cases and User Models

The following content shows how, based on the previous case studies, a couple of user models can be built up. Then a formal user model will be used to reflect the construction material management process. The proposed user model in this research is based on the implementation of user research in software development, to be specific, the Unified Modeling Language (UML). UML is used to build up a formal user model in order to generate a fast and smooth transition from the user needs to the software

products. Furthermore, object-oriented languages, such as Java, can help build up a configurable navigator pattern to retrieve information. The proposed approach will help the user in retrieving information according to their needs, and in reducing the ambiguity in interpreting their intentions.

The goal of this project is to improve the level of satisfactory results obtained from a conventional information retrieval (IR) system, by modeling the user's information retrieval approach and leveraging the user model into structured knowledge. The users' needs will be analyzed and modeled by the "use case" modeling method (Mellor and Balcer 2002), and then the relationships in the use case will be used to address the IR system with a more relevant request. Use Cases are not intended to capture all of the system requirements. For example, in the Material Purchase Order process, when a PM calls a vendor for certain materials' prices, this activity will not be included in the system.

The first step in this research is to build up the use case of a specific area. The use case diagram is a model of the system's intended functions and its environment that supports the business process. This model serves as a contract between the customer and the developer. As mentioned before, the material management process was chosen for the use case example because it happens with high frequency in a lot of construction companies. Table 8-1 shows the main scenario for a material management use case. The PM is responsible for checking the storage of materials, placing the purchase order, and the maintenance of the material record database.

Table 8-1 reflects the PM's view of the material management process. A more detailed model of the whole purchasing process is shown in Table 8-2. Although the use

Table 8-1 Material management business use case

PM Doing Material Handling Order
1. PM (PM) requests a new Material Handling Order (MHO) number. 2. The system prepares a blank MHO form and pulls in a list of available materials in the warehouse from the Material Catalog System. 3. PM selects primary and alternate materials. 4. For each material, the system verifies that the PM has the necessary authorization and adds the material to the MHO form and tags the material with the project #. 5. When the PM indicates the MHO form is complete, the system saves the form. Extensions: (The first number indicates the corresponding item above) 1a. Material Catalog System does not respond. The system notifies the PM and terminates the use case. 1b. PM already has a MHO form: System brings up the current version of the PM's MHO form for editing instead of creating a new one. 4a. That material is not enough for the quantity the PM specified. System notifies the PM and asks whether to continue or exit. 4a-1. If the PM wants to continue System adds the material to the schedule using the existing amount 4a-2. If the PM does not want to continue, the System disables selection of that material and notifies the PM.

case presented in Tables 8-1 and 8-2 is not a full-fledged business material management scenario, the idea behind doing this scenario is to start with modeling the users performing their information retrieval task and then incorporating the user model into software development.

After the formal use cases have been built up, the next step is to retrieve information from these formal use cases. Tables 8-1 and 8-2 are all natural-language requirements specifications (i.e. by use case descriptions). Table 8-3 is still a formal use case for modeling the material management purchasing process. In Table 8-3 all the actors have been highlighted, in this case, the PM and the Superintendent. Besides that, the activities and their objects have also been highlighted. With the highlighted items, items down to the details can be traced in the task module as well as what activities

Table 8-2 Formal use case for modeling material management purchasing process users

Purchase Material or Products
Primary Actor: PM
Goal in Context: PM buys material through the system, gets it. Does not include paying for it.
Scope: Business – The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests PM: Wants what he/she ordered and an easy way to do that. Senior PM: Wants to control cost but allow needed purchases. Vendor: Wants to get paid for any goods delivered.
Precondition: None
Minimal Guarantees: Every order sent out has been approved by a valid authorizer – Senior PM or President of the company.
Trigger: PM decides to buy something because s/he finds the requirements of the materials on the drawings or gets the requests from project superintendent for the materials.
Main Success Scenario:
1 Superintendent / PM (Requestor): initiate a request.
2 Senior PM (Approver): Check money in the budget, check price of products, complete request for submission.
3 PM (Buyer): Check contents of storage, find best vendor for products. Validate Approver's signature. Complete request for ordering. Initiate Purchase Order with Vendor.
4 Vendor: Deliver products to jobsite. Get receipt for delivery (Most of the times, Vendor prepared the receipt and requests the jobsite superintendent to sign it. This is out of scope of system under design).
5 Jobsite superintendent (Receiver): Sign the receipt. Send a copy of receipt to PM.
6 Superintendent / PM (Requestor): Mark request delivered.
Extensions
• 1a. Requestor does not know vendor or price: leave those parts blank and continue
• 1b. Prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing. (Delete from the system) Reducing the price leaves it intact in process. Raising the price sends it back to Approver.
• 2a. Senior PM does not know vendor or price: Leave blank and let PM filling in or call back.
• 2b. Senior PM is not Requestor's manager: Still okay, as long as Senior PM approves.
• 2c. Approver declines: Send back to Requestor for change or deletion.
3a. PM finds goods in storage: Send those up, reduce request by that amount and carry on.

Table 8-2. Continued

• 3b. PM fills in Vendor and price, which were missing: Request gets re-sent to Senior PM.
• 3c. Request involves multiple vendors: PM generates multiple POs.
• 3d. PM merges multiple requests: Same process, but mark PO with the requests being merged.
• 4a. Vendor does not deliver on time: System does alert of non-delivery.
• 5a. Partial delivery: Receiver marks partial delivery on PO and continues.
• 5b. Partial delivery of multiple-request PO: Receiver assigns quantities to requests and continues.
• 5c. Products are incorrect or improper quality: Requestor refuses delivered goods.
Technology and Data Variations List: None
Priority: Various
Release: Several
Response Time: Various
Channel to Primary Actor: Internet browser, mail system, or equivalent
Channels to Secondary Actors: Fax, phone, paper proposal
Open Issues: When is a canceled request deleted from the system? What authorization is needed to cancel a request? Who can alter a request's contents?

the user (PM or Superintendent) needs to do and what kind of detail the user needs to have for the information can be determined. Next the procedure of how the formal use case will be transformed into the ***EX***tensible ***T***ask ***E***lement ***T***ree (EXTET) structure will be discussed.

8.2 EXtensible Task Element Tree (EXTET)

Table 8-3 has been simplified for analysis purposes. Figures 8-1 thru 8-3 are based on two user groups: PMs (PM) and Superintendents. An ***EX***tensible ***T***ask ***E***lement ***T***ree (EXTET) for the above material handling process will be built. The EXTET is a collection of lightweight utilities for generating system commands and processing. EXTET is centered around a data structure for representing the formal use case for modeling the construction management process. As its name implies, this data structure is

Table 8-3 Highlighted formal use case for modeling material management purchasing process users

Purchase Material or Products
Primary Actor: PM
Goal in Context: PM buys material through the system, gets it. Does not include paying for it.
Scope: Business – The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests PM: Wants what s/he ordered and an easy way to do that. Wants to control cost but allow needed purchases. Vendor: Wants to get paid for any goods delivered.
Precondition: None
Minimal Guarantees: Every order sent out has been approved by the PM.
Trigger: PM decides to buy something because s/he finds the requirements for the materials on the drawings or gets the requests from project Superintendent for the materials.
Main Success Scenario (To simplify the analysis, the system will only execute the SEARCH function.):
1.Superintendent / PM (Requestor): Check the schedule and initiate a request.
2.PM (Approver): Checks for available money in the budget, check price of products, complete request for submission.
3.PM (Buyer): Checks contents of storage, find the best vendor for products. Validate Approver's signature. Complete request for ordering. Initiate Purchase Order with Vendor.
4.Vendor: Deliver products to jobsite. Get receipt for delivery (Most of the times, the Vendor prepares the receipt and requests the jobsite Superintendent to sign. This is outside of the scope of the system under design).
5.Jobsite Superintendent (Receiver): Checks the receipt against the detailed Material Take-offs and signs the receipt if they match. Send a copy of the receipt to the PM.
6.Superintendent / PM (Requestor): Mark request delivered.
Extensions (The system will handle the extensions, invokes exceptions, or reminds the User to execute another command)
• 1a. Requestor does not know vendor or price: leave those parts blank and continue
• 1b. Prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing. (Delete from the system) Reduced price leaves it intact in process. Increased price sends it back to Approver.
• 2a. PM does not know vendor or price: fill in or call back.
• 3a. PM finds goods in storage: Send those up, reduces request by that amount and carries on.
• 3b. Request involves multiple vendors: PM generates multiple POs.

Table 8-3. Continued

• 3c. **PM** merges multiple requests: Same process, but mark PO with the requests being merged.
• 4a. Vendor does not deliver on time: System issues an alert of non-delivery.
• 5a. Partial delivery: Receiver marks partial delivery on PO and continues.
• 5b. Partial delivery of multiple-request PO: Receiver assigns quantities to requests and continues.
• 5c. Products are incorrect or improper quality: Requestor refuses delivered goods.
Technology and Data Variations List: None
Priority: Various
Response Time: Instant
Channel to Primary Actor: Internet browser, mail system, or equivalent
Channels to Secondary Actors: Fax, phone, paper proposal
Open Issues: When is a canceled request deleted from the system? What authorization is needed to cancel a request? Who can alter the contents of a request?

a hierarchy of objects, each of which represents an element in the formal use case. The following analysis is strictly on elements and illustrates how a system command will be generated to retrieve data from the databases.

The procedure of how the formal use case will be transformed into the EXTET structure will not be addressed in this study. This procedure is code-intensive and the final program should include: (1) Element Definition; (2) Element Path; (3) Tree Structure Definition; (4) Tree Builder (which includes reader to read from the formal use case and writer to write to buffer or directly to the tree structure); (4) Other Tools. The EXTET structure after implementation of the formal use case in Table 8-3 is shown in Figure 8-1.

The previously described Case Study #1 in Chapter 5 is an example of material handling and management for a commercial construction project. In Case Study #1, both the PM and the Superintendent ask the system the same query: “Door #2 Delivery”. As

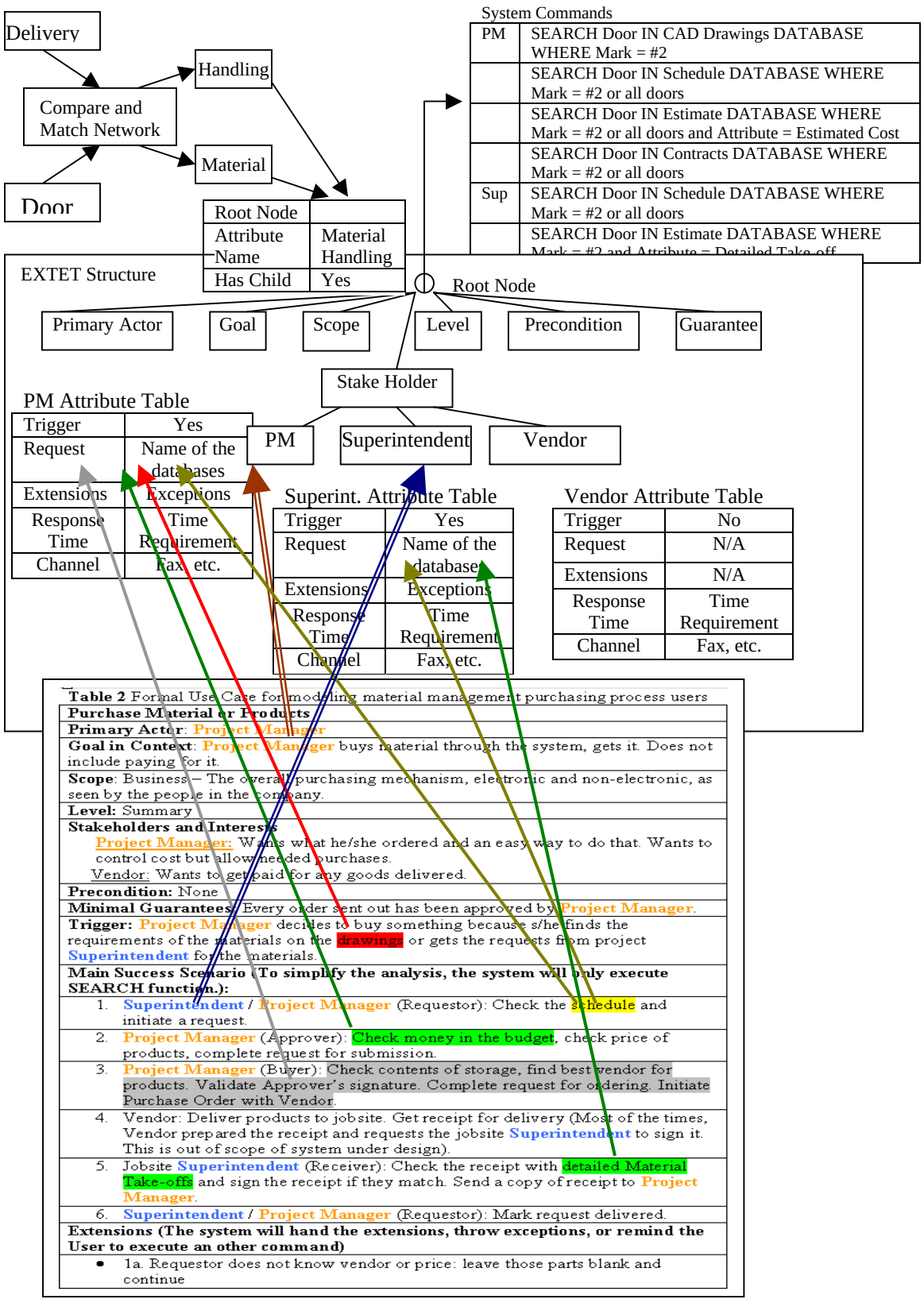


Figure 8-1 Extensible Task Element Tree (EXTET) Structure

illustrated in Figure 7-1 in Chapter 7, if this query is an input into the proposed system, the Compare and Match Network will match this query with the material handling function, and it will ask the system to call the Material Handling EXTET structure. The upper right hand corner of Figure 8-1 shows the output of the system commands generated by the EXTET structure. It can be read as follows:

- If a PM (PM) logs into the system and inputs the query: "Door #2 Delivery", the system commands generated by EXTET would be as follows:
    - SEARCH Door IN CAD Drawings DATABASE WHERE Mark = #2
    - SEARCH Door IN Schedule DATABASE WHERE Mark = #2 or all doors
    - SEARCH Door IN Estimate DATABASE WHERE Mark = #2 or all doors and Attribute = Estimated Cost
    - SEARCH Door IN Contracts DATABASE WHERE Mark = #2 or all doors
- If a Superintendent (Sup.) logs into the system and inputs the query: "Door #2 Delivery", the system commands generated by EXTET would be as follows:
    - SEARCH Door IN Schedule DATABASE WHERE Mark = #2 or all doors
    - SEARCH Door IN Estimate DATABASE WHERE Mark = #2 and Attribute = Detailed Take-off

As shown in Figure 7-1 in Chapter 7, all the system output (system commands) will be sent to the ElementAmbassador and the Modified Navigator to retrieve information from the integrated databases. The system commands in this example are detailed, specific, and ready for the system to execute. This study is focused on how a user model driven architecture configures the GUI and the Application Layer. As discussed before and as shown in Figure 8-1, for the same query, different user groups will get different extended answers based on their need. In addition, since the EXTET Structure is based on the formal use cases of different tasks, for the same user, if s/he queries the system under different tasks, the answers will be different according to the task and the user preferences. Here, the meaning of the word "Extensible" in the proposed EXTET

Structure is elaborated upon. Suppose a PM asks the "Door #2 Delivery" query to the system. Without EXTET or any filter, there are two ways to retrieve information and to present it back to the user:

- Go to Scheduling Database, find the item regarding doors, and give that answer to the user. These are the operation under the assumption that the user wants to find information regarding Door #2 and Delivery.
- Or the system could find all the information regarding Door #2, plus the information regarding Delivery, and give all the information back to the user. These are the operation under the assumption that the user wants to find information regarding Door #2 or Delivery.

With the ***EX***tensible ***T***ask ***E***lement ***T***ree structure, what the user gets is not limited to the Scheduling database, but it also under reasonable extended scope. The user's needs get satisfied based on their preference and the task requirements.

After the EXTET system has generated all the system commands, the next step is to send these commands to the ElementAmbassador and the Modified Navigator for specific information. The system proposed by Reinhardt (2003) is composed of the ElementAmbassador and the Navigator for integrated product and process models. With adequate modifications, it would be feasible to use that system as an Application Layer for information retrieval.

There are four databases shown in Figure 8-2.They are for CAD Drawings, for P3 Scheduling, for Excel Estimating, and a XML Contracts Databases. The Navigational Models in Reinhardt's system are modified for the purposes of this research, so that each of the group nodes is embedded with a couple of more attributes. As such the group nodes can provide detailed information as well as meta-data about an item. Figure 8-3 shows that Navigational Models were added with some attributes in each of the leaf

nodes and non-leaf nodes. This modification to Navigational models will help further preprocessing data processed from Element Ambassador and make it easier to demo data.

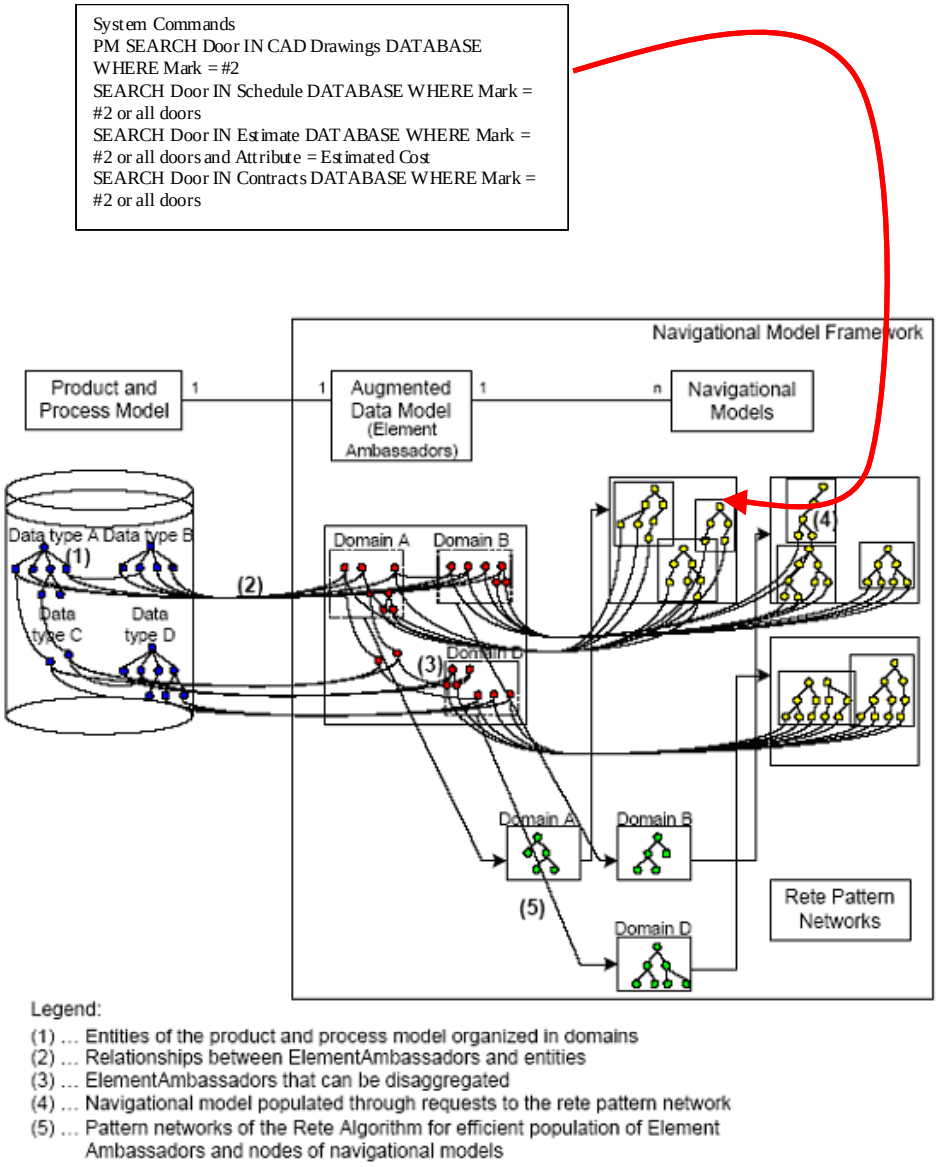


Figure 8-2 Element Ambassadors and the Modified Navigational Models (Reinhardt 2003)

8.3 User Model Driven Architecture (UMDA) Implementation

In Chapter 7, the conceptual design of the User Model Driven Architecture (UMDA) has been discussed and the possible applications and functions included in the UMDA design were analyzed. The intent behind the implementation of the UMDA

design is to configure the User Interface and the application layer of the entire system. The system will present to the user with what s/he needs. The Interactive User Interface will also provide the user with the ability to extend to more detailed information. In order to implement the conceptual design of the UMDA, the functions defined in Section 7.3 in Chapter 7 (such as user log-in, input interface, user's queries, and query parser, etc.) have to be implemented and fulfilled. The following content will discuss the UMDA Implementation starting with the user input.

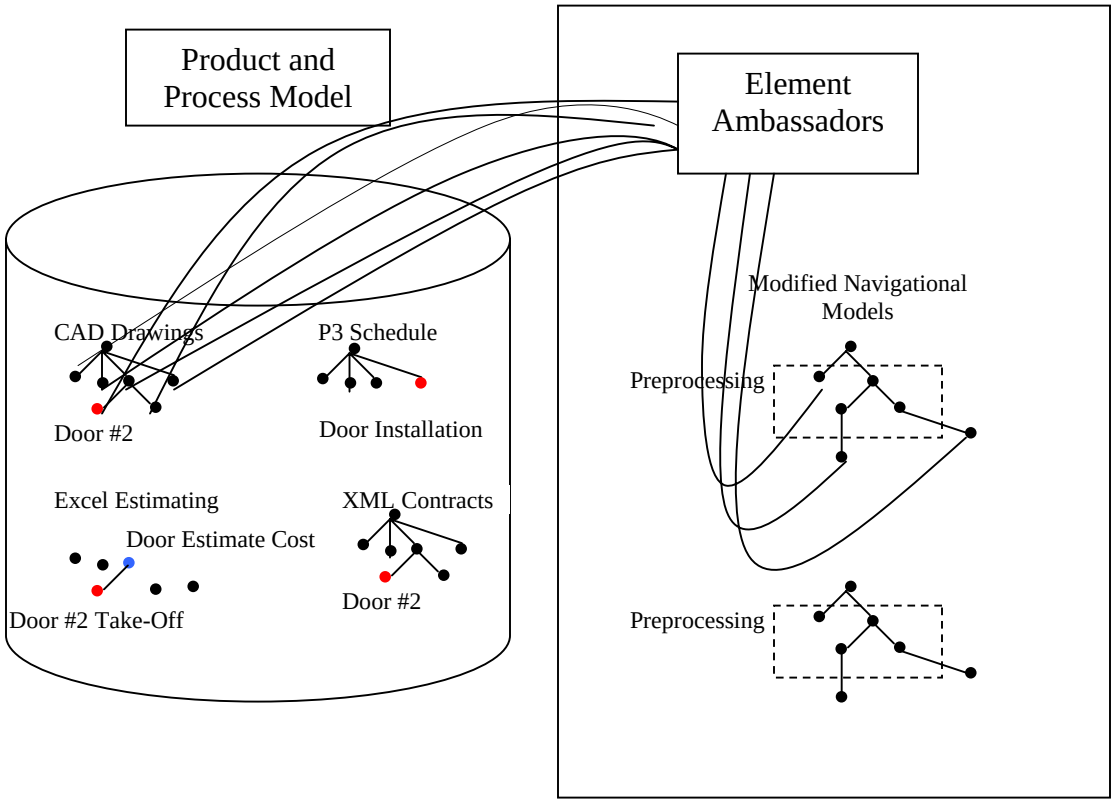


Figure 8-3 Modifications on Navigational Models

This section includes six (6) parts of the UMDA system as shown in Figure 8-4. Figure 8-4 adds six (6) blocks to the Figure 7-1 in Chapter 7 to reflect the UMDA system work process. The contents in this section are also organized according to the system work process. The order of the six (6) blocks is as follows: once the user logs into the

UMDA system, the system will first collect user information. Then it will parse the user query. The third step is to analyze the user query. The next step is to send the analyzed query into the use case databases. After that, the system will execute the EXTET structure and generate system commands. The last step is to use these system commands to navigate the construction project databases. The details of the six (6) blocks will be discussed below.

8.3.1 Collect User Information

Collecting user information is the first step in the 6 blocks shown in Figure 8-4. As mentioned before, the UMDA system is able to collect the necessary user information and search the information a user needs according to his/her user group and/or intended task. The user groups designed in the current UMDA system are the construction project

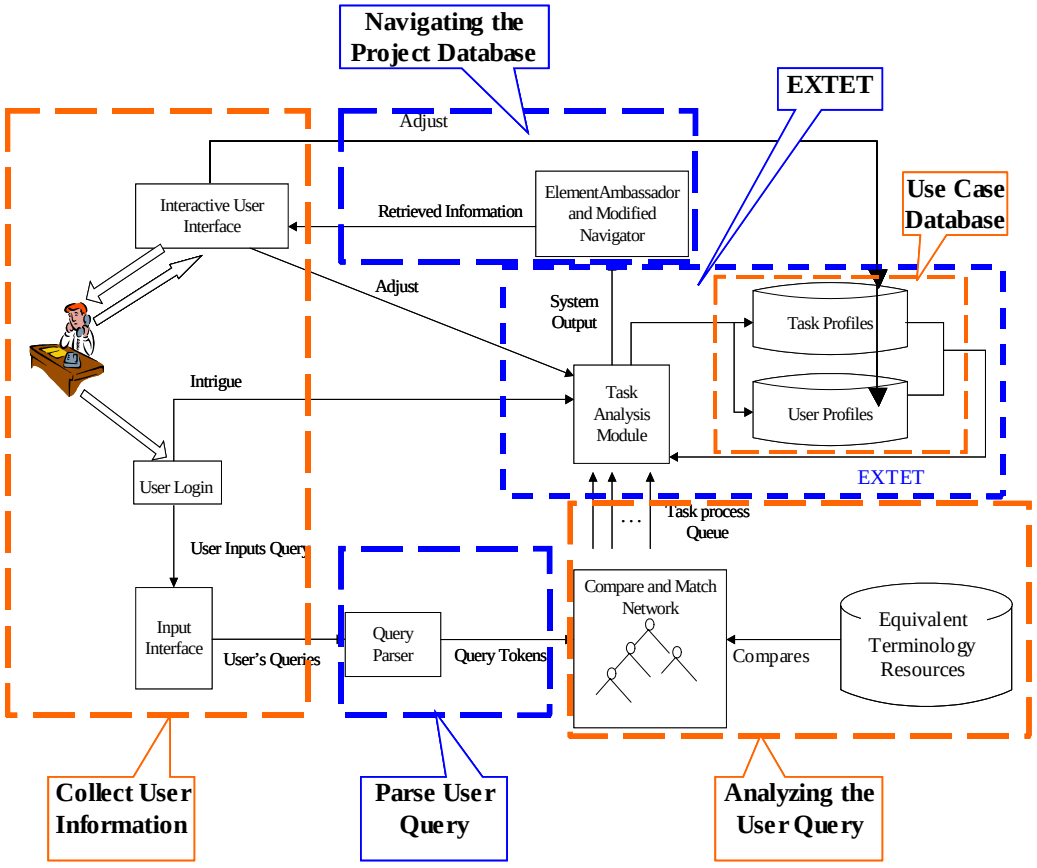


Figure 8-4 The UMDA system working process

participants. The project participants include those from outside of the construction company and the company staff. There is one more group of users, the system administrator, who is responsible of using and maintaining the system. The user groups implemented in the UMDA system from outside of the construction company (outside users) include Owner, Architect, Engineer, Subcontractor, Vendor, and Building Authorities. The company staff includes PM, Superintendent, Estimator, President, and Office Manager. Messages, pictures, and any other content available in the system can only be changed by the person who initially uploaded them. Outside users can also inform the PM or System Administrator to remove any posting about which the outside users have made formal complaints.

Figure 8-5 shows the interface for collecting user information from outside users. If a subcontractor logs into the system, a tool will automatically collect the users' log data in the UMDA system environment. The user information collected includes company name, telephone number, fax number, address, email, website, photos, project description.

User data can be stored in the XML format (i.e. Figure 8-6) or in a database (i.e. a user information table as shown Figure 8-7. In the future, if the UMDA system is upgraded into a web-based system, the XML format for user data storage will be a good choice. For the proposed UMDA system, the primary target of implementation is to demonstrate the user model driven system itself. An Entity/Relationship (E/R) model will be able to meet the requirement. Outside user groups are categorized and listed in a jComboBox with the name "Role". "jComboBox" is a Java Swing component and it performs the drop-down list function.

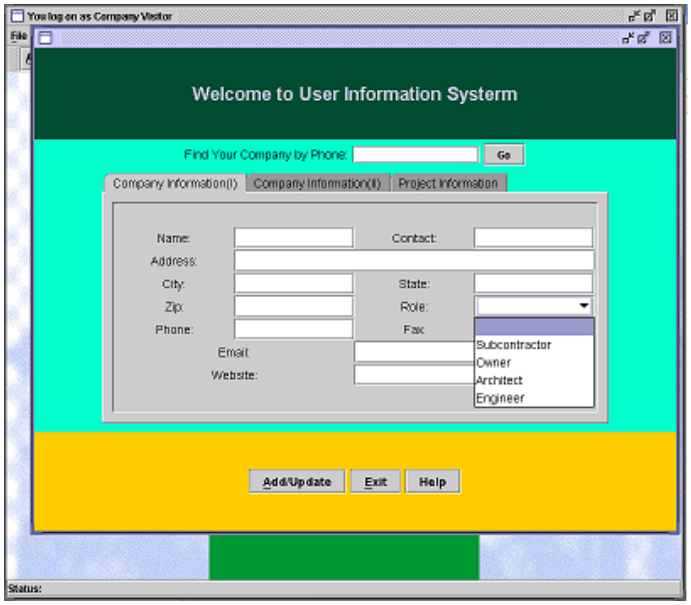


Figure 8-5 Interface for collecting user information from outside users

```
<Record>
        <Company_Information_I>
                <Action Type="...">
                        <Name>...</Name>
                        <Contact>...</Contact>
                        <Address>…</Address>
                        …
                        <!—RoleType= Owner, Architect, Engineer, Subcontractor,
                        Vendor, and Building Authorities -->
                        <Role RoleType=…>…</Role>
                </Action>
        </ Company_Information_I>
        <Company_Information_II>
                <Action Type="...">
                        <Company_Picture>...</Company_Picture>
                        <Company_Description>...</Company_Description>
                </Action>
        </ Company_Information_II>
        <Project_Information>
                <Action Type="...">
                        <Project_Picture>...</Project_Picture>
                        <Project_Description>...</Project_Description>
                </Action>
        </ Project_Information>
</Record>
```

Figure 8-6 Skeleton of XML user log data

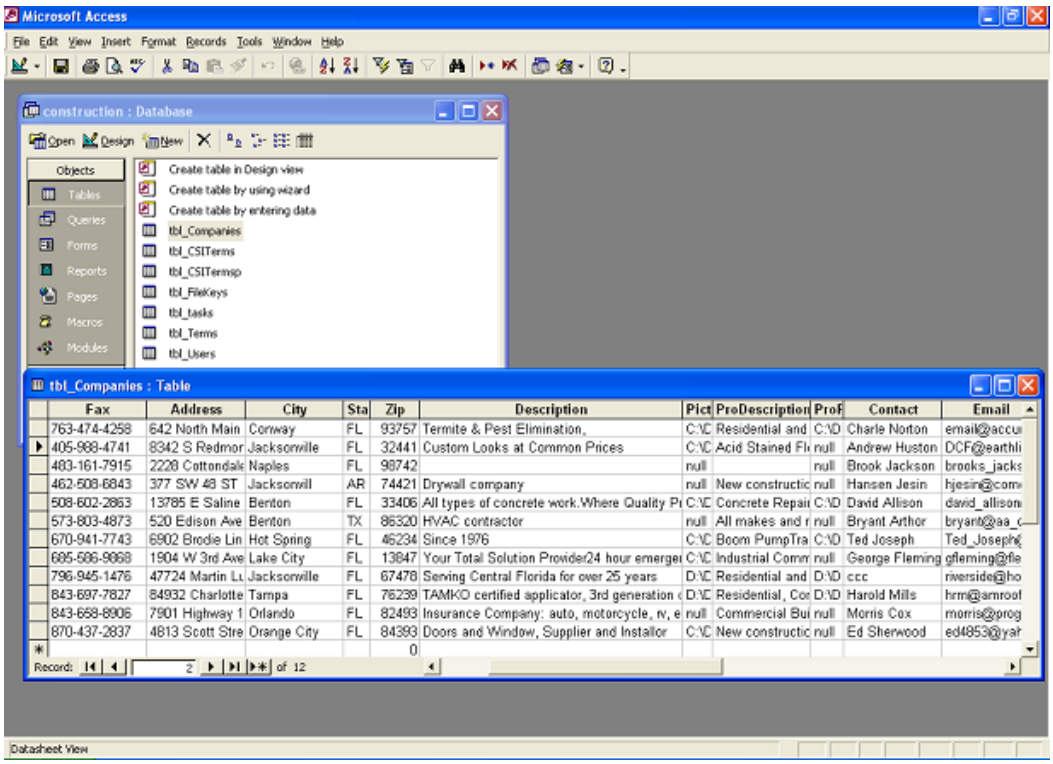


Figure 8-7 User information table in an MS Access Database

As shown in Figure 8-5, there are two other jTabbedPanes (Java Tabbed Pane area) next to the one with the name "Company Information (I)". These two jTabbedPanes are labeled "Company Information (II)" and "Project Information." In the tabbed area labeled "Company Information (II)", users can fill in the description of their company and post their company pictures. In the tabbed area provided by "Project Information," users can submit to the system the projects they did and the project photos. If this is the first time the outside user logs into the system, the user needs to fill the user information. For most users, their telephone numbers could be used as primary key in the database. The Add/Update button performs two functions. For the first-time user, this button will add the user information into the system. If user made changes to his/her record, the button will update the information in the system.

The company staff needs to log in through the second button with the name "Company Staff Login". If the user clicks the button, a login window will pop up and ask the user to input their user name and password. This is a regular user status check process. After the user successfully logs into the system, the system interface will change. The command bar on top of the window will show the changed interface. Some example, the commands include File, VisitorManagement, Dictionaries, Search, and Help. If the user chooses to search for some queries in the project database, s/he will click the Search command. A search query input window will be displayed and the user will be asked to input the query. Figure 8-8 shows the search query input window and the changed interface.

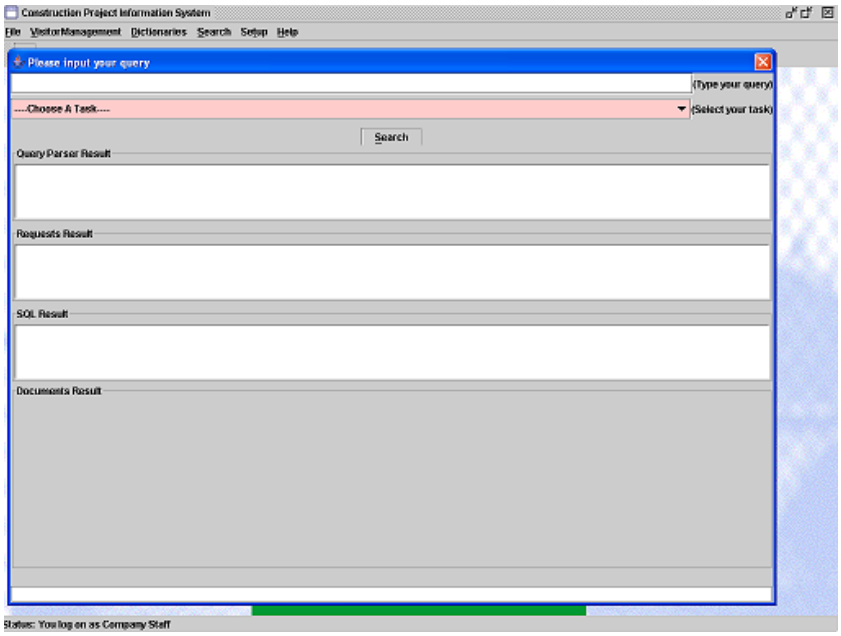


Figure 8-8 Search query input window and the changed interface

User Interface design includes a lot of theoretical, empirical, and methodological issues and relates to User Science and to System Design. Many organizations have been doing research on Human Computer Interaction (HCI) to improve User Interface design (HCIbib 2004). In order to provide the needed user interfaces, the functions and

input/output design need to be defined by system views, which means from the system design point-of-view, defining the user interface in the proposed system. The output of the system can be defined in two stages:

- Stage One: the system command stage. The proposed system has the abilities to search information from multiple databases. In order to let user understand the databases that the system retrieved information from, the system will display the name of the databases and the paths the proposed system uses to navigate to the data.

- Stage Two: the project information display stage. The proposed system finds out the required data according to the system command. Then it will display the data to users. The view components of the proposed system utilize the view functions provided by some software. For example, to read AutoCAD drawings, the system uses the view functions in the Autodesk DWF Viewer. Figure 8-9 shows an example of the view components in the proposed UMDA system. In Figure 8-9 the meta-info of the CAD file is interpreted. If the drawings database has the data only from project drawings, it will not be able to answer queries about some general attributes of the construction project (i.e. carpentry). With metadata added to the original drawings database, information could be retrieved as shown on the drawings (i.e. detail of window headers); and we can also answer queries about the meta-info of the CAD files (i.e. total number of windows).

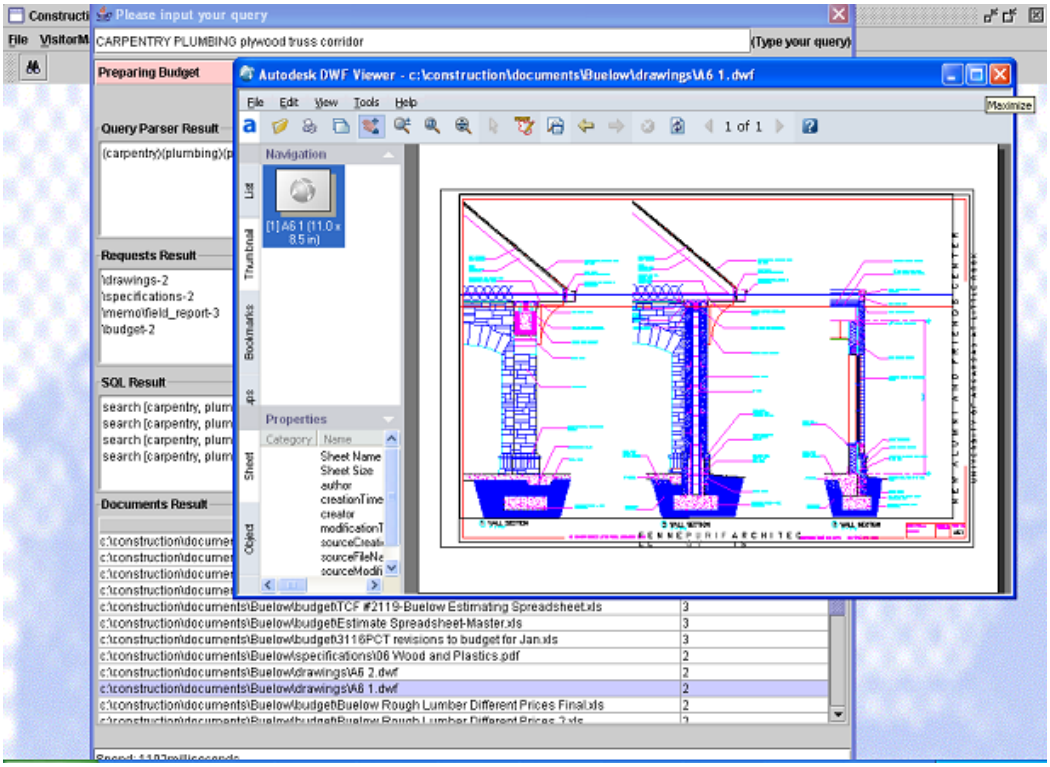


Figure 8-9 Example of the view components in the proposed UMDA system

8.3.2 Parse User Query

After a user logs into the UMDA system and the system collects the information about the user, the user can search for the project information stored in the construction project database. The next step is to parse the user's query. When the search query input window is displayed, the user will be able to retrieve information from the system with the help of the UMDA system. The goal for parsing user queries in the proposed system is to find some methods to parse comma- or space- separated text in a Text field and to make sure that the words in multiple fields are ready for use by the proposed system. Several methods have been used by software companies in order to parse space-separated text fields, and to display the resulting text strings in multiple text fields. For example, one method developed by Microsoft for the MS Access DBMS uses an expression in a query that includes three functions: the Instr() function to search for the comma in the Text field, and the Left$() and Right$() functions to extract the two parts of the Text field. To parse a Text field that contains two words separated by a comma, the system will create the following query:

```
Query: QueryTest
---------------------------------------------------------------
Field: FirstName: Right$([Empl],Len([Empl]) - InStr(1,[Empl],",") -1)
  Show: True
Field: LastName: Left$([Empl],InStr(1,[Empl],",") -1)
  Show: True
```

In the proposed system, Java language is implemented to parse the user queries. The users can use familiar terms, such as construction related words, to search for data and documents. The query parser includes the following functions:

- Connect to database management system.
- Read each of the characters in the user query and save them into a token string.
- Read the token string and analyze the characters in the string.

The last step is further discussed in the next section. The flowchart in Figure 8-10 illustrates the use of the UML charts to show the algorithm for finding out the intention of the user query. Figure 8-10 also shows the sequence of functions and parsing and analysis algorithms used for the user queries.

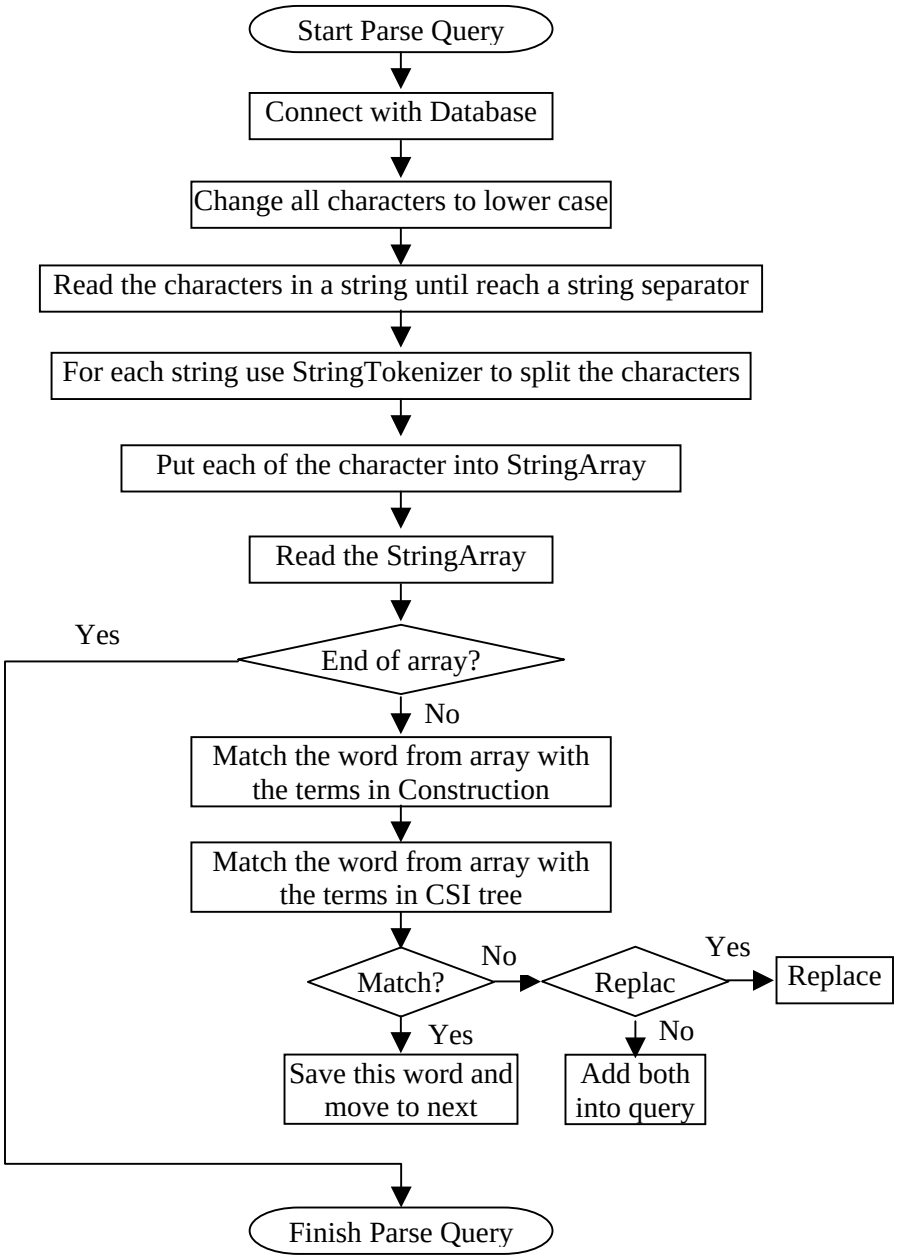


Figure 8-10 Algorithm for the Java Class QueryParser

The algorithm shown in Figure 8-10 for the Java class QueryParser executes both the function of parsing and analyzing user queries. When the UMDA starts parsing user's query, the system first connects with the construction terminology database and changes all characters in the query into lower case. Then the system reads the characters in the query string until it reaches a string separator (i.e. space). For each string in the query, the system uses StringTokenizer to split the characters. After the system splits the characters, the query string will become a few tokens (words), which are ready to be processed. The next step the system will do is to put each of the tokens into a data structure called StringArray. Then the system will read from the array and match each of the tokens with the construction terms.

8.3.3 Analyzing the User Query

After the user's query has been converted into tokens, the UMDA system is ready to analyze the query. As mentioned earlier, Figure 8-10 shows the algorithm for both the function of parsing and analyzing user queries. To analyze a user query, the ideal situation is that the user knows exactly what s/he wants and knows precisely the terms a database management system would accept. Most of the time, users are not sure about the words they need to use to find the information they need. For an ordinary user, especially a construction worker, s/he has limited knowledge about the content of a construction project database. It would be hard for users to guess the right term. Sometimes, users may misspell the word(s) they want to use in the query. When users need to analyze large columns of detailed business information, IT organizations usually predefine and tune the user queries and run those queries in batch mode during off peak hours. This requires significant IT resources and removes the benefits provided by ad hoc interactive queries. To a construction project database user, the ability to analyze (slice and dice) the data and

to drill down through data to investigate particular details, situations and records are very important.

As a machine, the computer communicates with human through interfaces and input/output. Enabling the computer to understand the meaning of human language involves deploying many technologies (including knowledge engineering, learning algorithm, artificial intelligence, etc.). The UMDA system is focused on the users in the construction process. This research assumes that for the proposed system, there are two situations that could happen when a user type in a query:

- The user types the right query s/he wants; there is no misspelling in the query; and the construction project database system has the information for the query;
- The user query needs to be improved for the following reasons:
    - The user does not have any database background knowledge and could not type in a valid query.
    - The user could not find the exact query to reflect the information s/he wants to search.
    - There are some misspellings in the query.
    - The construction project database system does not have the terms used in the query.

The first case is the best-case scenario. In that case, the proposed system will simply parse the query and send every token to the underlying Task Analysis Module as shown in the Analyzing the User Query block in Figure 8-4. For the second case, the system will implement some user-friendly designs. The author proposes the inclusion of the following knowledge engineering functions in the system:

- Use space, comma, and other punctuation marks to find the mark of separation within the sentence of the user query.
- For those users who do not have any database background knowledge and who could not type in valid queries, the proposed system uses a tree structure to organize and present the 16 divisions of construction knowledge in MasterFormat developed by CSI (The Construction Specifications Institute). Users can use the

information found in this CSI MasterFormat window to specify and finalize their queries.

- If there are some misspellings in the query, the proposed system will use construction thesaurus terminology to find the word(s) that may match the user's intention and display to the user. The user then can choose to select the word(s) the system suggested or go with the word(s) s/he typed before. If the user uses the word(s) suggested by the system, the system will replace the typed-in word(s) with the suggested word(s). If the user insists to use the typed-in word(s), the system will keep all the typed-in word(s) and the suggested word(s) and send them to the construction project database.

- It might happen that the construction project database system does not have the terms used in the query. With the help from the CSI MasterFormat and the construction thesaurus terminologies, the probability of zero result searches for some queries will have been decreased. EXTET, which will be discussed in the next section, can also help users find the results for their needs. The design intent of the proposed system is to help users find information according to their needs. Merely extending the search scope and displaying lots of results for the user will cause a system design that neglects the user needs. The proposed system design not only helps users to extend their inquiries so that they will not omit some useful data, but it also filters out the unnecessary information according to user group and user task.

Figure 8-10 shows the algorithm for the Java Class QueryParser. It includes the UML charts to show the algorithm to find out the intention of user query, to compare the typed-in word with the thesaurus word, and to find out the CSI MasterFormat suggestions. Appendix C includes the sample Java code for the QueryParser program.

In case a user misspells some words in the query, the proposed UMDA implements a thesaurus dictionary or domain specific dictionary to give the user replacement suggestions. There are a couple of ways to compare and suggest words to users. The "Bit-String Longest-Common-Subsequence Algorithm" (Allison and Dix 1986) operates in terms of bit or bit-string operations. It offers a speedup of the order of the word-length on a conventional computer. The longest-common-subsequence (LCS) problem is to find the maximum possible length of a common subsequence of two strings, 'a' of length |a| and 'b' of length |b|. This algorithm requires O(|a|*|b|) operations on single bits or

O(ceiling(|a|/w)*|b|) operations on w-bit computer words or O(|b|) operations on |a|-bit bit-strings, for a fixed finite alphabet. It has the same complexity as simple LCS algorithms, if w is greater than any additional multiplicative cost, this algorithm will be faster. If |a|<=w the algorithm is effectively linear in |b|. An alphabet larger than |b| can be effectively reduced to |b| by sorting 'b' in time O(|b|*log(|b|)) and using index positions in the sorted string (Allison and Dix 1986). The LCS algorithm is shown in Figure 8-11, and the sample code is shown in Figure 8-12.

```
rowi = x And ((x - (rowi-1 << 1)) != x)
  where x = rowi-1 Or bi-string

  Or And !=  bit-string operations
  <<         logical left-shift bit-string; right-hand bit set
  -          |a|-bit integer subtraction
```

Figure 8-11 LCS algorithm (Allison and Dix 1986)

The longest-common-subsequence (LCS) algorithm has deficiency in matching two words with the difference in the first letter(s). For example, if use the algorithm to compare “concrete” and “ocncrete”, the system will evaluate them as totally different words and stop comparison right after comparing the first letter of each of the two words. There is another match algorithm that uses the edit distance. Edit distance measures textual similarity. Formally, given two strings S_1 and S_2, their edit distance, denoted Δ_ε (S_1, S_2), is the *minimum* number of edit operations (insertions, deletions, and substitutions) of single characters that are needed to transform S_1 to S_2. For instance, Δ_ε (“concrete”, “ocncrete”) = 2. It is known that the complexity of computing the edit distance between strings S_1 and S_2 is O (| S_1 | X | S_2 |), where | S_1 | and | S_2 | are the lengths of S_1 and S_2, respectively (Yianilos and Kanzelberger 1997).

```
// Compute Edit Distance a.k.a. Levenshtein distance
  public static int LD (String s, String t) {
  int d[][]; // matrix
  int n; // length of s
  int m; // length of t
  int i; // iterates through s
  int j; // iterates through t
  char s_i; // ith character of s
  char t_j; // jth character of t
  int cost; // cost
    // Step 1
    n = s.length ();
    m = t.length ();
    if (n == 0) {return m;}
    if (m == 0) {return n;}
    d = new int[n+1][m+1];
    // Step 2
    for (i = 0; i <= n; i++) {
      d[i][0] = i;
    }
    for (j = 0; j <= m; j++) {
      d[0][j] = j;
    }
    // Step 3
    for (i = 1; i <= n; i++) {
      s_i = s.charAt (i - 1);
      // Step 4
      for (j = 1; j <= m; j++) {
        t_j = t.charAt (j - 1);
        // Step 5
        if (s_i == t_j) {cost = 0;}
        else {cost = 1;
        }
        // Step 6
        //d[i][j] = Minimum (d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1] + cost);
        d[i][j] = Math.min(d[i-1][j]+1 , Math.min(d[i][j-1]+1 , d[i-1][j-1] + cost));
      }
    }
    // Step 7
    return d[n][m];
  }
```

Figure 8-12 Example code for the implementation of edit distances

Given a string s, an integer q, and the set of *q-grams of s*, denoted G(s), is obtained by sliding a window of length q over the characters of string s. For example, given a

string “concrete” and an integer 2, G(s) is obtained by sliding a window of length 2 over the string “concrete”.

$G(s_1)$ = G(“concrete”)={‘co’, ‘on’, ‘nc’, ‘cr’, ‘re’, ‘et’, ‘te’}.
$G(s_2)$ = G (“ocncrete”)={‘oc’, ‘cn’, ‘nc’, ‘cr’, ‘re’, ‘et’, ‘te’}

The *relative q-gram distance* between two strings S_1 and S_2, denoted $\Delta_q(s_1, s_2)$, is defined as (Yianilos and Kanzelberger 1997):

$$\Delta_q(s_1, s_2) = 1 - \frac{|G(s_1) \cap G(s_2)|}{|G(s_1) \cup G(s_2)|} \tag{8-1}$$

For example, Δ_q (“concrete”, “ocncrete”)

$$= 1 - \frac{|\{'nc','cr','re','et','te'\}|}{|\{'co',''on','nc','cr','re','et','te','oc','cn'\}|}$$
$$= 1 - 5/9$$
$$= 0.44$$

The smaller the relative q-gram distance between two strings is, the more similar they are (Yianilos and Kanzelberger 1997).Use the above example, if s_1 = “concrete” and s_2 = “concrete”, Δ_q (“concrete”, “concrete”)

$$= 1 - \frac{|\{'co',''on','nc','cr','re','et','te','oc','cn'\}|}{|\{'co',''on','nc','cr','re','et','te','oc','cn'\}|}$$
$$= 1 - 9/9$$
$$= 0 < 0.44$$

So the value of relative q-gram distance can reflect the similarity of two strings. The edit distance algorithm can avoid the shortcoming of the LCS algorithm. In addition, the edit distance algorithm has the ability to find out how much similarity the two compared strings have. If a user accidentally misspells the first and the last letter of a word, the edit distance algorithm can still compare and match the word with some suggestions. In the proposed UMDA system, a string distance algorithm is implemented. The edit distances between the string typed by user and the words in the construction dictionary will be compared. As explained above, if the edit distance Δ_ε (S_1, S_2) between

two strings S_1 and S_2 is 0, that means the two strings are an exact match. For any edit distance greater than 0, the two strings are different. The purpose of using edit distance in the proposed UMDA system is to give some suggestions to the users. If the edit distance between two strings is too large, which means there are a lot of different characters in the two strings, the suggestion may not be valid. On the other hand, if the edit distance is too short, there may not be any match in the construction dictionary. So, there should be a threshold distance. Considering that for long words, user will intend to make mistakes, the author used different threshold distances for strings with different lengths. Using Dt denoting the threshold distance and Sl denoting the string length, the author implemented the following rules in the UMDA system:

- if Sl < 6, then Dt = 2;
- if 6 <= Sl < 8, then Dt = 3;
- if Sl >= 8, then Dt = 4.

The usage of the CSI dictionary can be divided into two kinds. One is to use it as a reference. When a user click the CSI dictionary on the command bar, a new window will show up and display the entire CSI tree structure to the user. Then the user can go through all the terms in the structure to find the word s/he needs. The other way is to use it as a dictionary database. Similar to the construction dictionary database, the CSI database also implements the edit distance to find suggestions to give the user. What makes the CSI database different from the construction dictionary is its ability to expand user's queries. The CSI MasterFormat has three levels of titles. The construction terminologies are organized into these three levels, which in a computer programmer's view are in a tree structure. If the string in the user's query matches the terms in the first or second level, the system will help the user to expand the query to the second or third

level. For example, “Metal” could match the term on the first level of the CSI tree. The UMDA system will expand it to the second level and suggest “structural metal framing”, “metal joists”, and “metal deck” to the user. These suggestions will help users find the right words they want to use to retrieve information.

8.3.4 Use Case Database

This part will implement one of the key contributions in this research – the use case database that is formalized from construction activities. In Chapters 2 and 7 the details of the construction use case database proposed in this research have been discussed. The implementation of the Use Case Database includes the data and the management system. The use cases have been developed by using UML. Chapter 5 described the UML models for some construction use cases and some formal use cases for construction management processes. Chapter 6 discussed the requirements on these UML models and the formal use cases. The formal use cases will be changed into XML format by the UMDA system. XML is a standardized text format designed specifically for transmitting structured data to Web applications. In addition, XML's goals of being easy to learn, use, and implement will have clear benefits for users, making it easier to define and validate document types, and to transmit and share them.

The Use Cases stored in the proposed system are in the XML format and illustrate some construction management processes. The proposed system has the following XML documents for task profiles:

- Change_Order
- Close_Out_Administration
- Competitive_Bidding
- Estiblish_Cost_Model_and_Develop_Cost_Extimates
- Identify_Trade_Contracts
- Maintain_Project_Site_Document

- Material_Management
- Perform_Constructability_Review
- Perform_Quality_Control_Inspection
- Perform_Value_Engineering
- Preparing_Budget
- Pre_Qualification
- Project_Scheduling
- Provide_Project_Staffing
- Safety
- Shop_Drawing_Review
- Submittal_Process

For user profiles, the proposed system has the following XML documents (Appendix D has detailed XML documents stored in the UMDA system):

- Architect
- Building Department Official
- Engineer
- Estimator
- Office Manager
- Owner
- President
- Project Manager
- Subcontractor
- Superintendent

In Appendix D, the user profiles and task profiles are completely separated, which means that they are not related to each other. Figure 8-13 shows the details of the change order task profile. The task profile and the callouts on it show that the task profile does not have user information in it. Since the main function the UMDA system performs is to search according to user needs, the change order task profile shown in Figure 8-13 is focused on the request to perform the request of user's query. This change order task profile is generated from a change order use case. The format of the change order use case is similar to the formal use case shown in Table 8-3. But the XML task profile is not

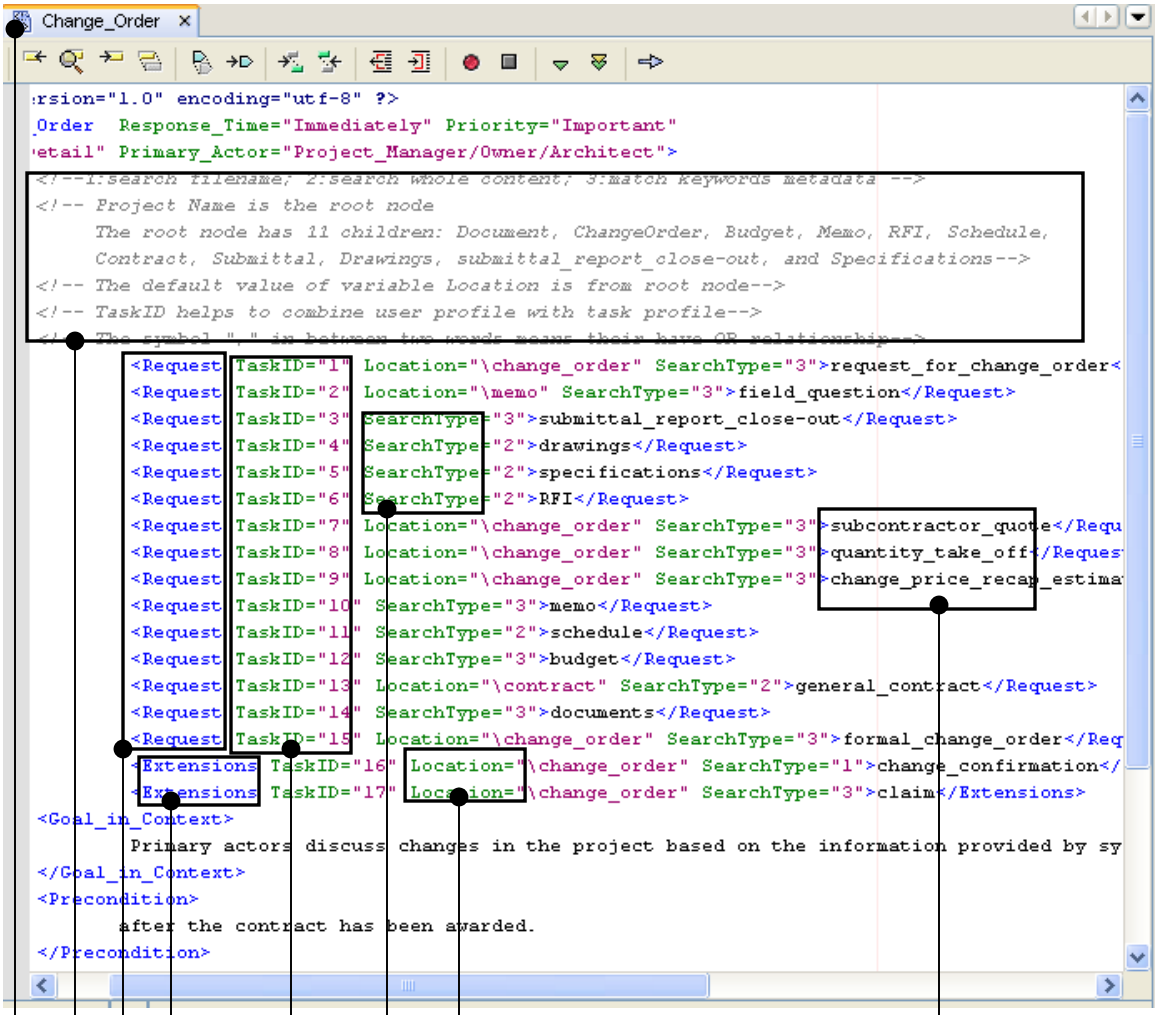


Figure 8-13 Details of the change order task profile

simply copied from its formal use case. After carefully analysis, only the items relating to the task are organized into the task profile. The other information, which relates to user habits, user preferences, or user specific aspects, is organized into the user profile. The example of user profile is shown in Figure 8-14. Figure 8-14 is the user profile for project managers. The user profile and the callouts on it also show that the user profile does not have task information in it.

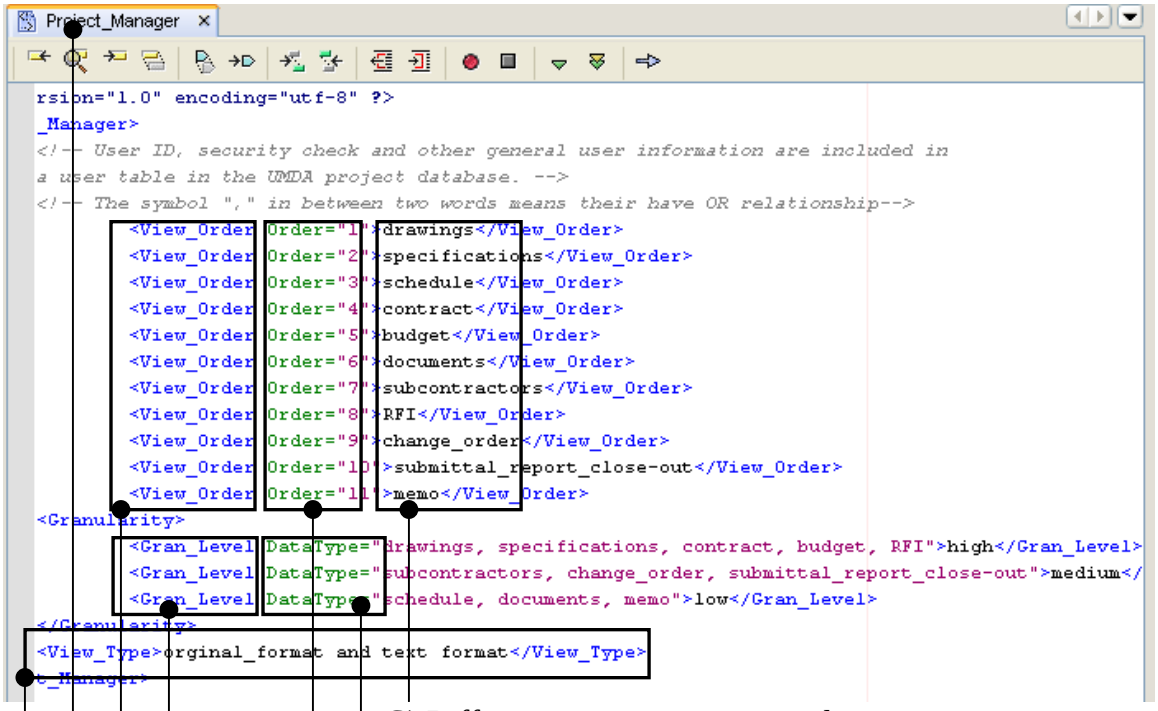


Figure 8-14 Details of the Project Manager user profile

Figure 8-13 is an example of a task profile and Figure 8-14 is an example of a user profile. The two groups of profiles are separated. The disadvantage of separating these two groups of profiles is that it requires the involvement of human experts and consumes time to recognize each object and its attributes in each of the profiles.

But this separation of user profiles and task profiles has many more advantages when compared to its required tradeoffs. In addition, by separating the two different groups of profiles, scalability can be achieved. In the field of user model or use case implementations in computer systems, stereotype approach is straightforward and still utilized. The concept of stereotype approach has been existed for ages since 1980s (Tockey 2004). The stereotype approach refers to the method that directly converts the formal use case into XML without further processing it. One feature of the stereotype approach is that it does not separate user preferences from user's tasks. Compared with the stereotype approach, the proposed separated approach has several advantages.

The first advantage is its scalability. When change happens, especially when more use cases are added to the XML use case database, the entire system should be able to function properly without major structural change. The scalability of the XML use case database depends on the scalability of the fundamental architecture of the entire UMDA system. With every function being completely independent, the amount of resources necessary grows linearly with the amount of requests received. In a system that does not scale poorly, the amount of resources necessary would increase at a higher rate than the number of requests. Each part of the UMDA system is independent from each other. The independency of the user profiles and task profiles also makes the use case database scalable. When a new user profile and/or task profile is added to the database, the only

change the system needs to make is in its combination structure, which is the EXTET structure, which will be explained in the next section. The separation of the two groups of profiles ensures that the database and the management structure are scalable to new records.

The second contribution of the separated XML use case database is the reduction of the total number of records stored in the database. Figure 8-15 shows the comparison between the stereotype approach and the separated profile approach of the XML use case database. Suppose there are three (3) user profiles and 10 task profiles in the separated use case database. The total number of items stored in the use case database is 13. For the stereotype approach, the user information and task information are not separated. So to iterate all of the cases and keep the complete records, the use case database needs to store:

Number of user information records X Number of task information records

= 3 X 10 = 30 (items)

Thus, the total number of records stored in the stereotype approach database is 30.

The third advantage of the separated profile approach is that it is easy to maintain. The XML data structure of the use cases, allow one to understand the file structure or database scheme and makes one better able to modify it. In many legacy systems using flat files the data structure cannot be modified without breaking the applications that rely on the existing data structure. With the separated profile approach and its EXTET structure, it is comparatively easy to provide links to other XML data elements, and this will dramatically improve with the introduction of link and pointer, techniques that enable pointing into a document from the outside without changing it.

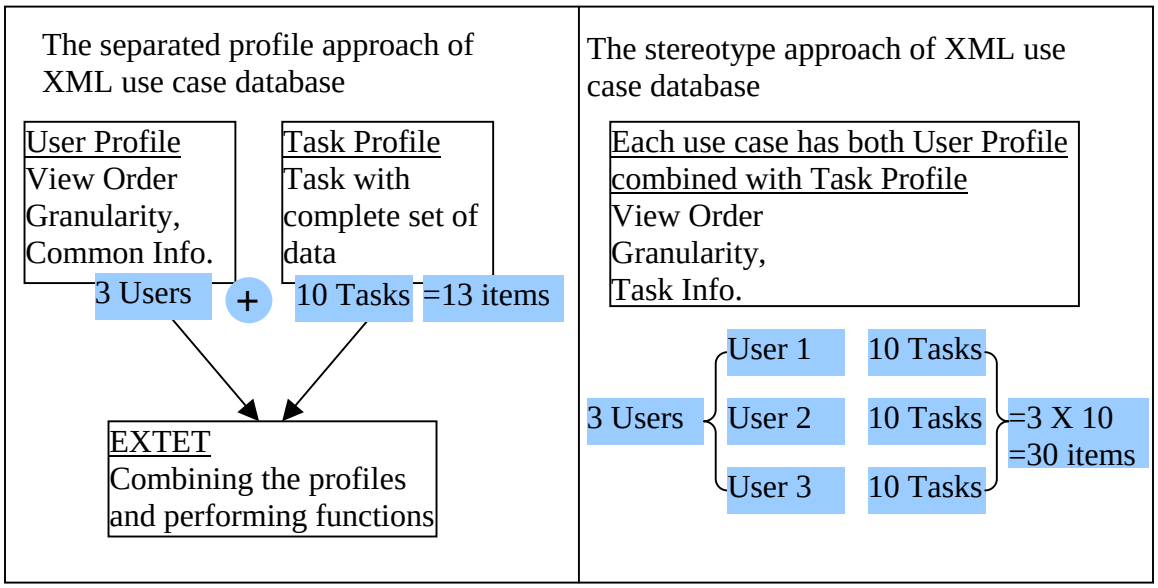


Figure 8-15 Comparison of stereotype and separated profile approaches

If users of the proposed system need to edit or add more use cases to the Use Case Database, they will also be able to revise or add the XML documents to the system. By doing so, different companies can tailor the system to adapt to their business conditions and requirements.

Instead of hardwiring the existing user model, the UMDA system has to separate to two classes (and potentially more classes) of profiles and combine them. The XML format use case database is encoded or in plain text. The next step is to build a tree out of the information enclosed in the formal use cases while resolving default values. This resolution process can include the retrieval of specific terms (e.g., the retrieval of a Project Manager's preferences and attributes of certain tasks). The tree is called the Extensible Task Element Tree (EXTET). It is the manager of all profile objects and it also implements the methods to generate system commands to retrieve information from the underlying construction project database.

8.3.5 Implement Extensible Task Element Tree (EXTET)

After the separation of user profiles and task profiles, the next step is to combine them make them functional. This combination step is performed by the Extensible Task Element Tree (EXTET) structure. The structure can also generate system commands to navigate through the construction project database.

As stated in Chapter 1, the problem the research is focused on is the retrieval of construction information based on the user's specific needs. Chapter 5 described three distinctive use case examples that frequently occur in construction to show the complexity of construction information retrieval. The variety of construction projects and different user needs and their work tasks make the problem difficult to resolve. As mentioned in Chapter 2, current research has not developed an efficient or effective way to solve this problem. General User Modeling System (GUMS) was proposed in 1986 but has never been implemented. The IBM Object View Interaction Design (OVID) has user involvement in system design. But user only involves in system functional analysis and system performance evaluation. The Lumiere project at Microsoft uses Bayesian reasoning for automated assistance. It observes the user's work habits and the environment parameters, then analyzes those parameters using expert knowledge and Bayesian reasoning, before finally making suggestions to the user. The individual user is the observing object of the system. User needs are not paid enough attention. Besides, the Lumiere project is not scalable enough to adapt to changes. Chapter 2 and Chapter 6 also reviewed some construction software systems. User needs consideration can be found in some systems, but this consideration is still limited to functional analysis.

The previous section reflected one of the significant contributions of this UMDA system. It described the implementation of the use case database and the separation of the

different types of profiles. The use case database and the separation made the UMDA system scalable and provided a solution to the difficult problem in construction information retrieval.

The separation of the user and task profiles is critical to make the UMDA system scalable. Compared with other user models that are currently implemented in some systems, this scalability has helped in improving the efficiency and flexibility of the entire system. The next key contribution in this research, the Extensible Task Element Tree (EXTET). The EXTET structure is an important component of the UMDA system. The separated user and task profiles provide scalability to the system. To make the system work and function according to what the formal use cases depict, the UMDA needs to combine the two individual parts together. This combination work is performed by the EXTET structure. The profile database transfers data in a user model (as described in Chapter 7) into formatted XML files. The EXTET structure retrieves information from XML files and organizes it on the tree structure. This transferring and organizing accomplishes the implementation of the functions and requirements of user models.

The algorithm for the Extensible Task Element Tree (EXTET) is shown in Figure 8-16. The function of EXTET is to generate the system commands for user queries based on the collected user information and user task profiles and to combine the separated user and task profiles. When the UMDA system receives a user query, it will parse the user query and change the query string into individual tokens. The information embedded in the tokens includes the query for construction project data. It may also have the information about the job the user wants to perform or uses to specify the scope of the construction project data. Those steps are performed by the following 3 blocks in the

UMDA system: Collect User Information module, Parse User Query Module, and Analyzing the User Query module. Then the tokens are sent to the EXTET structure. When the EXTET structure receives the tokens, the EXTET will start user/task profile analysis. Since both user profiles and task profiles are XML formatted files, the EXTET structure will import Java XML parser. Java XML parser is an application provided by Sun Company. It acts as a parser to analyze XML files. The tokens received by the EXTET structure may have tokens about the task the user specified. If there is a token about the construction task the user wants to perform, the EXTET structure will open both the task profile for that token and the user profile to generate system commands. If no token is about the construction task to be performed, the EXTET structure will simply open the user profile for the user and generate system commands. When the EXTET structure needs to open both user profile and task profile, the structure will combine the two profiles together.

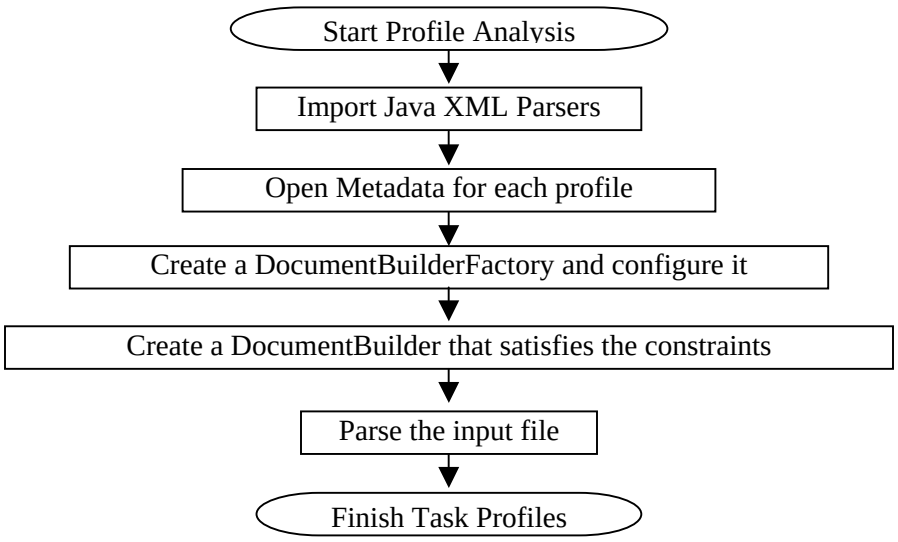


Figure 8-16 Algorithm of EXTET

As shown in Figure 8-16, the EXTET structure will open the metadata file of the user profiles and task profiles. The metadata file of the profiles has the ability to combine the user profile and task profile together. The details of the metadata combination file are shown in Figure 8-17. Figure 8-17shows how the system continuously uses the Change Order task profile as shown in Figure 8-13 and the Project Manager user profile as shown in Figure 8-14. User profiles and task profiles are combined together by task ID. Task profiles have task ID listed in it, but user profiles do not. As shown in Figure 8-13 explanation E. “Task ID helps to combine with user profile”, the Task ID is the key that links the user profile and the task profile together. Which task item or sub-task will be assigned what number is not important. The distinctive part of these sub-tasks is that they are itemized and have their specific identification numbers. As shown in Figure 8-17, the EXTET structure uses a user-task combination table to combine the two groups of profiles together. The first column in the combination table is about the different user type information, for example, project manager and superintendent. The second column in the table lists the task names, for example, change order. The third column contains the task ID numbers. For each type of users and each type of tasks, there are certain task-ID numbers, which in turn represent sub-tasks the UMDA system should perform. Figure 8-17 also has a partial screen-shot of the change order task profile to show where the task identification numbers are. To combine these two different groups of profiles, some expert knowledge needs to be implemented to build up this table. For example, if a PM wants to retrieve information regarding a Change Order, the PM needs to perform some tasks. Those tasks are identified by their ID numbers. In the current implementation of the EXTET structure, the structure can automatically retrieve user profiles and task

profiles from formal use cases. To diagnose the sub-tasks for each user under each task situation in the user-task combination table, human effort and expert knowledge will be

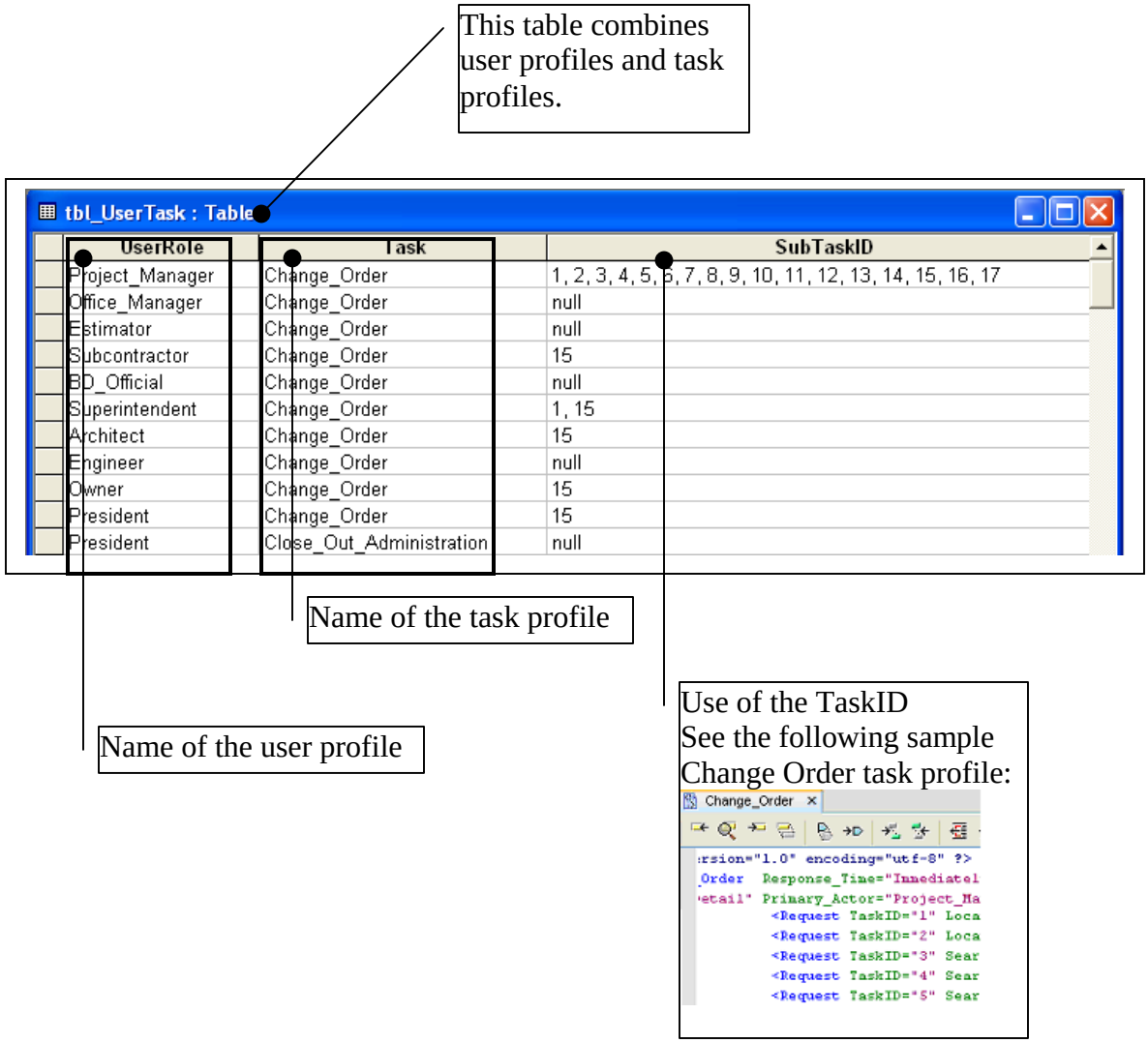


Figure 8-17 Metadata combination of profiles

needed. This compact link simplifies the work to build up and maintain the profile database. Detailed information about the ID numbers and their related records can be found in Appendix D.

After the profiles have been combined and parsed, the next step is to generate system commands. At this point, the EXTET structure has already collected the specific

user and task profiles. The XML profiles have the information about user preferences and task related data, for example, the location of the files. Based on the sample system commands shown in Figure 8-1, the command to search for a door's drawing information will be as follows:

SEARCH Door IN CAD Drawings DATABASE WHERE Mark = #2

In the above command,

- "SEARCH" is a standard SQL term.
- "Door" is from the user query tokens.
- "IN" is a SQL term.
- "CAD Drawings" specifies the exact location to start search. From user query tokens, the EXTET structure finds out the token(s) about the task profiles. Then it retrieves the exact location from the XML profiles.
- "DATABASE" and "WHERE" are all SQL terms.
- "Mark =" is from the XML profiles.
- "#2" is from the user query tokens.

As explained in Figure 8-2, the EXTET generated system commands will be sent to the Navigational Model Framework. The navigator structure will use the system commands to travel along the ElementAmbassador and find the entities associated with construction project information. The UMDA system can use system commands to find the exact group nodes. Figure 8-18 shows a partial segment of the program code for the EXTET functions.

Figure 8-1 shows the EXTET structure, while Figure 8-16 shows the algorithm of the structure and Figure 8-18 lists the partial code of the program for the EXTET structure. All these figures reflect the extensibility feature of the structure. The extensibility feature is reflected by the fact that the EXTET is a tree structure. If the

system needs to add more entities to the tree nodes, it can be done with minor revisions to the EXTET code. In addition, the profiles in the current use case database are separated

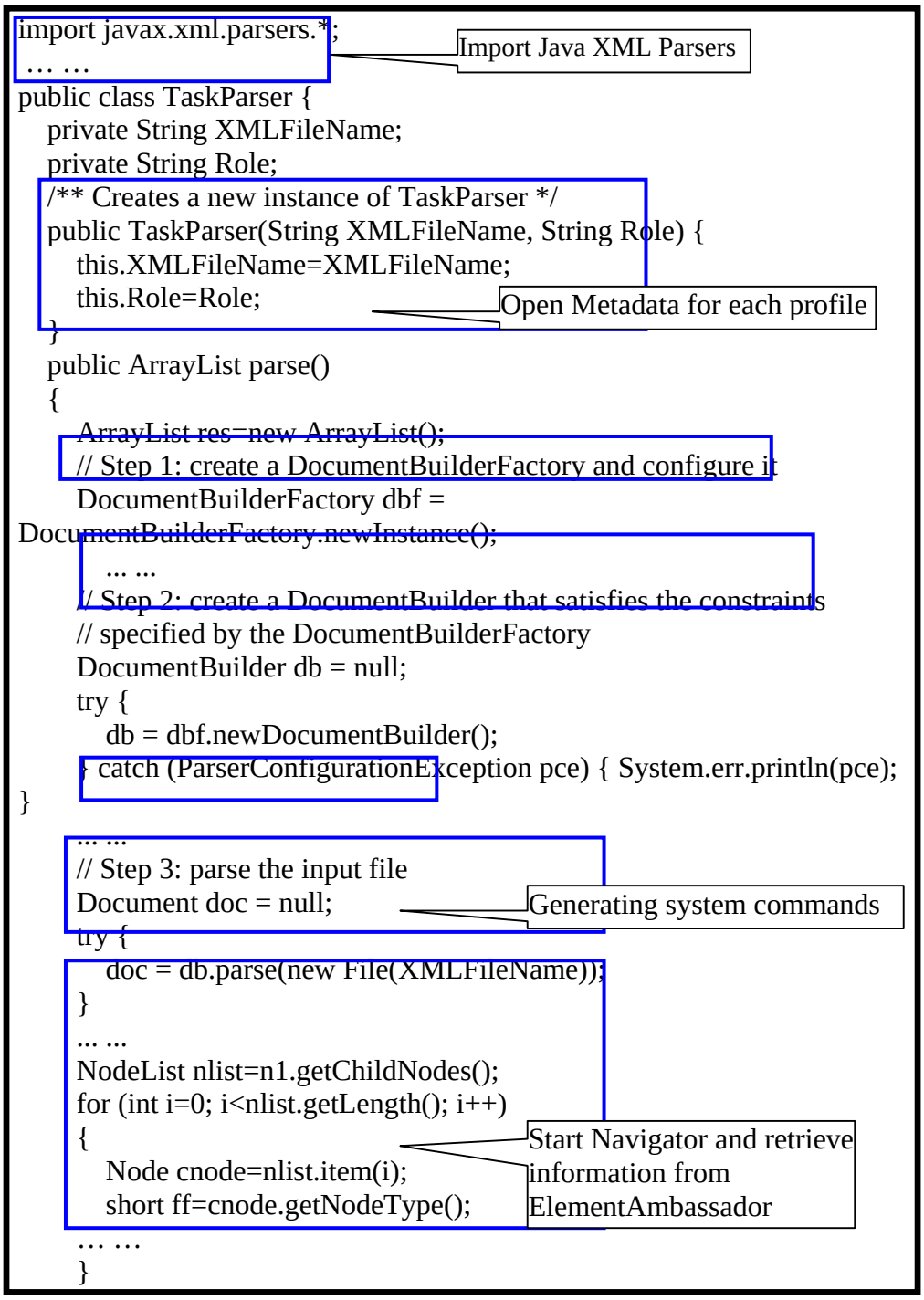


Figure 8-18 Partial of program code for the EXTET functions

into two groups. If in future research, more groups of profiles are separated or added, all the entities with new information can be loaded into the EXTET structure. The combination of the separated profiles also provides the extensible feature of the system.

The combination is not simply adding two profiles together. Figure 8-17 shows how expert knowledge is added to the table. This table is easy to extend and adapt to new knowledge.

Since the XML format was used to organize the information in the task profile database, the data in each of the task profiles can be loaded into a tree structure. So depending on how many task profiles exist, the UMDA system needs to populate the same number of EXTET structures. In addition, the UMDA system is an open system. Users can define more task profiles to adjust the search engine and as such the population of EXTET is not a fixed number.

There are multiple ways to find out the task a user wants to perform. For instance, the system could have a task term database. Once a user types a word that matches the task term in the database, the system will use the corresponding task profile to generate a system command. This method involves a lot of uncertainty. The UMDA system uses a drop-down list and requires the users to select a task they want to perform. The drop-down list works well when the tasks specified in the UMDA system are not so many. If the system needs to store and use a lot of task profiles, the system has to use other ways to select the task profile. The execution on the task profile depends on the rules specified in XML documents. Please refer to Appendix D for more information.

This section has described the concepts and implementation of the EXTET system. The section also related its approach to some existing research findings. Compared with

the stereotype approach, which is not scalable, the separated profiles and the corresponding EXTET structure are technically more effective and efficient. The design choices have been made to make the system scalable and extensible.

8.3.6 Navigating the Project Databases

The previous sections have shown a complex problem in construction and the conceptual process used in the UMDA system to solve it. This section will explain how the generated system commands navigate the underlying construction project database and retrieve information according to user needs.

In software systems, each part has an interface, and software parts are linked together to form a software system. These software parts are referred to by different names, such as subsystems, modules, or components. They must be identified and they must have a version number. They must have compatible interfaces (White 2000). However, because software systems are always changing, it is very hard to manage the configuration of software systems. The purpose of software configuration is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle. Software configuration management involves identifying configuration items for the software project, controlling these configuration items and changes to them, and recording and reporting status and change activity for these configuration items. In order to build a configurable environment, the system has to be able to adapt to changes. More importantly, the system shall have the appropriate interfaces between each of the system components and have the capacity to put together proper components to perform a task.

The UMDA system has the ability to configure software. After the proposed system executes the EXTET structure and generates some system commands, it is time for the

system to go to the underlying construction project database and to use the commands to retrieve the required data. As discussed in Chapter 6, the author chose to use the ElementAmbassador and Navigational model (Reinhardt 2003). Figure 8-19 explains the algorithm the system uses to build up the search engine. The first step is to connect to the existing construction project database. Then the UMDA system uses the system commands generated from the EXTET structure to configure operations. The UMDA system will use the navigator structure to locate the group nodes specified in the system commands. After the required group nodes in the ElementAmbassador have all been determined, the UMDA system will perform the match functions to find the exact data. The last step is to present the search results to the user. The presentation will be configured according to the information provided by the user profiles in the use case database. At this point, the UMDA system will have configured both the application layer and the GUI layer. The system will have retrieved and presented information to users according to their needs. Section C.2 in Appendix C shows the partial view of the program code navigating the project database.

The selection of the construction project database is specified by the rules included in XML documents. Each object in a XML document has two attributes: Location and SearchType. Location shows where to find the requested information. SearchType tells the UMDA system how to find a piece of information. In Chapter 7, the fact that different users will need different types of details on the same topic was discussed. Figure 7-2 in Chapter 7 showed the complexity of a query depending on the databases to be accessed. The SearchType in those XML documents reflects the complexity or the needed information granularity. There are three search types implemented in the UMDA system:

1. Search by filename;
2. Search whole content of the file;
3. Match keywords of the metadata file

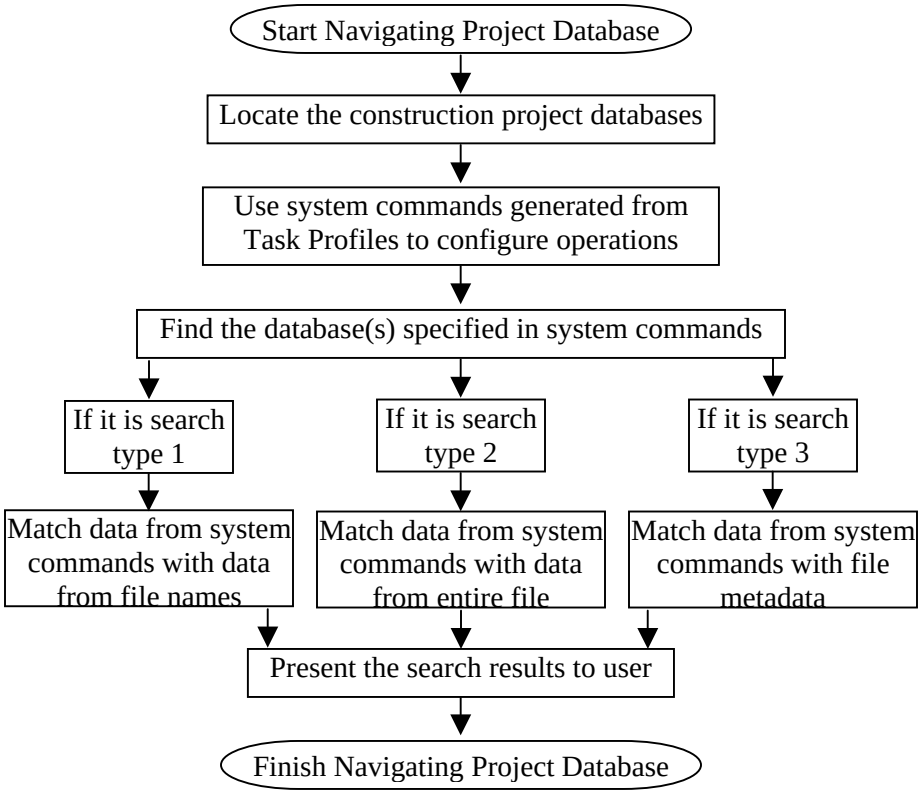


Figure 8-19 Algorithm of the system search engine

At this stage, the author has performed the process from use cases, to system functions, to the UMDA system and its EXTET structure. Next a complete example will be used to show the execution of the UMDA system. The example is the material management task for door #2 in the sample project. Figure 8-20 shows the entry screen where a user will be identified. It also shows the user's query and the refinement of the query. Figure 8-21 shows the query results derived from the construction project database. An example of the drawing searched is shown in Figure 8-22.

The next section is about the interesting aspects encountered by the author in designing and implementing the proposed system.

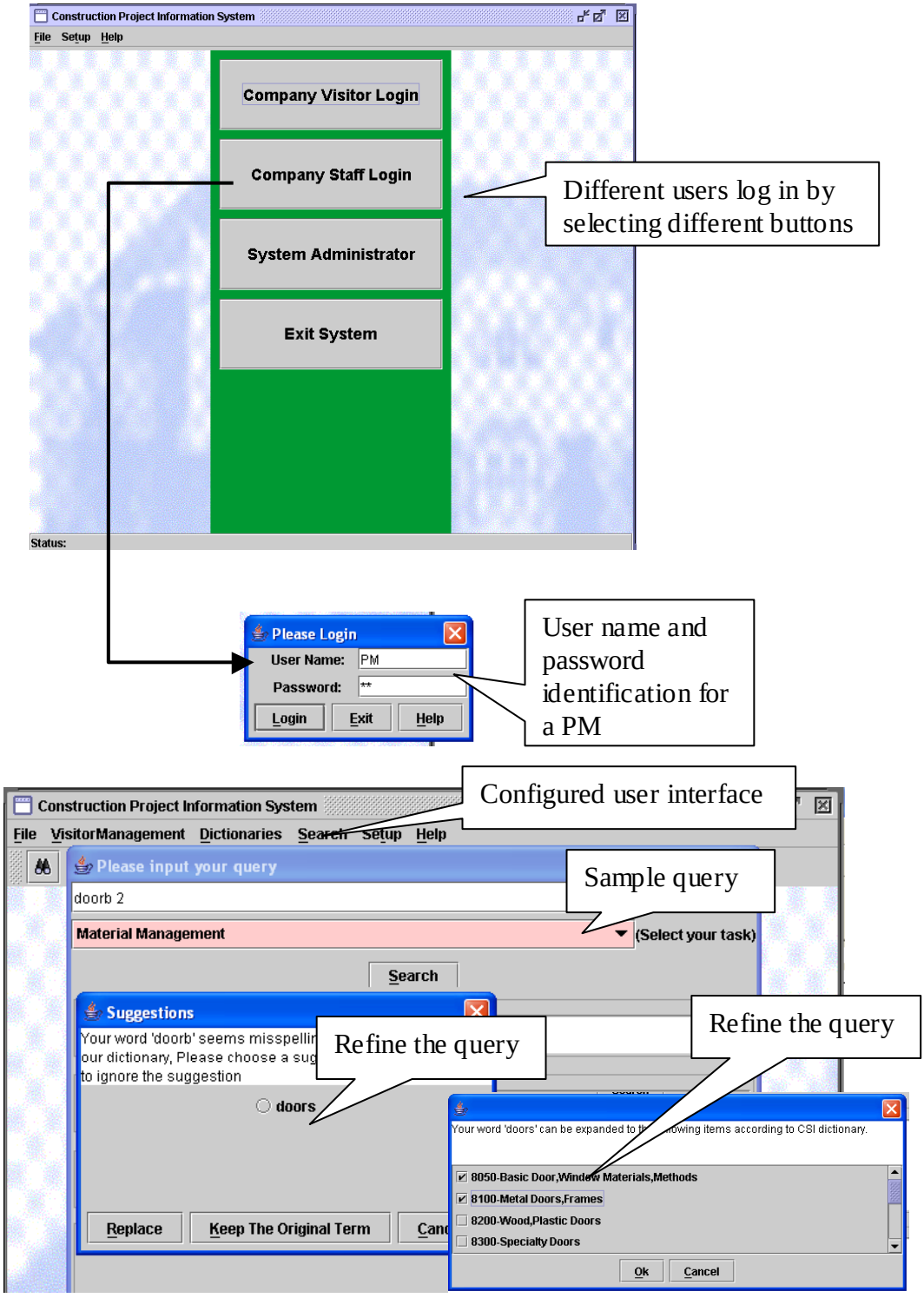


Figure 8-20 Entry screen, query, and refine the query

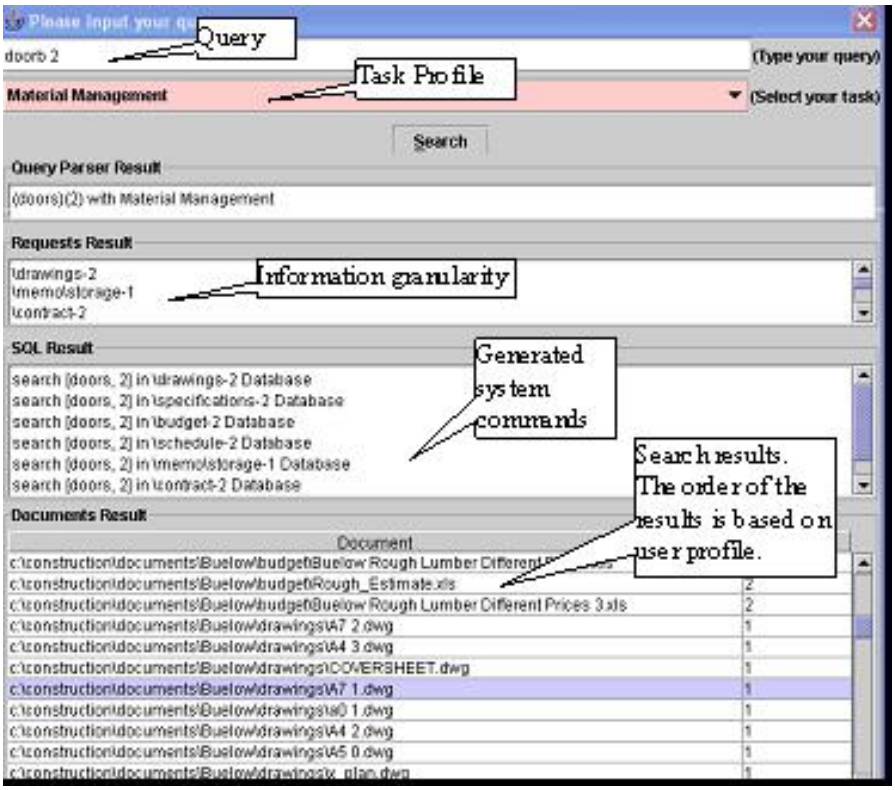


Figure 8-21 Sample query results

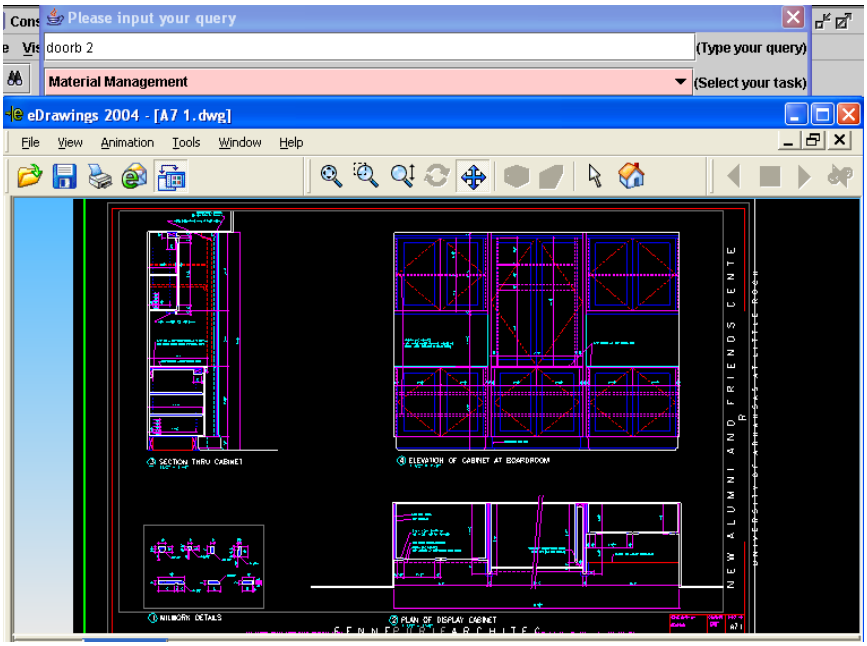


Figure 8-22 Example of searched result

8.4 Interesting Aspects of the Implementation

The following includes some interesting aspects encountered by the author in designing and implementing the proposed system. One of the research questions of this dissertation is to find out how to utilize the existing functionalities from AutoCAD, Primavera Project Planner (P3)/Microsoft Project, Timberline/Excel, and other existing software systems that are popular in the construction industry. The UMDA system uses the following software systems to facilitate design and programming:

8.4.1 Autodesk DWF Viewer

The Autodesk DWF Viewer is used to display and control the presentation of drawings. The Autodesk DWF Viewer has been developed by Autodesk, Inc., which is also the developer of AutoCAD. Since AutoCAD is one of the popular design software packages used in AEC industry, using this Autodesk DWF Viewer can fulfill the system functional requirements. Since construction companies do not typically design projects and since the revision rights on the architectural drawings are reserved for the architects, construction companies will most frequently use the Viewer functions.

8.4.2 Microsoft Office software

- Microsoft Word has been utilized in the UMDA system. By using Word, users are able to open a file, compose it, edit it, and save the word documents, and use all the functions provided by Word.
- Microsoft Excel is also available in the UMDA system. Similar to the Word program, users can use Excel functions to open a file, edit it, and save spreadsheet documents.
- Microsoft Project is used to display and revise schedule files. It has the same implementation as Word.
- Other Microsoft Office Tools. They will have the same implementations as Word.

8.4.3 Acrobat Reader.

Some files in the project databases are PDF files. Acrobat Reader will be used to display and control the presentation of these files. After files have been converted to PDF format and have been saved in the project databases, they do not need to be changed. Acrobat Reader will be adequate to display the files. In addition, it is a freeware at present and the user can download it from Adobe's website.

8.4.4 Supporting Software

In order to let the Java system recognize the words in Word or PDF files, the UMDA system implements a software system called "Convert Doc (CD)". To read from the Excel files, the UMDA system uses another software package called "Convert XLS (XLS)". The CD and XLS systems will help the UMDA system read strings from Word, PDF and Excel files.

8.5 Summary of the System Implementation

Implementation of design ideas for a real software system challenges and extends the designer's background knowledge and practical skills. From the first step of analyzing user requirements, to building up use cases, to designing the system architecture, to the last step in making sure the system works just as designed, the prototype of the UMDA system will go through the entire user-centered design process.

The process of building up use cases involves the design of UML models, formalizing use cases, and implementing XML documents to store user and task profiles. The EXTET (Extensible Task Element Tree) system has been proposed to manage and retrieve information from the task profiles. The implementation steps of the UMDA system include collecting user information, parsing user queries, analyzing user queries, composing use case databases, building EXTET, and navigating the project databases.

The UMDA system is a prototype software package that constitutes a reference implementation of the navigational model framework. In addition to the flexibility of representations provided through the existing software components (level of detail, structure of the project decomposition, etc.), the UMDA system provides flexibility along other dimensions. Based on its handling of the user and task profiles, the UMDA system is flexible with respect to the product and process models of construction management.

The intent of the UMDA system design is to find a system that can serve the users' information retrieval needs and the system should be scalable and extensible. A literature search was conducted to look for solutions to this technical problem. The review of existing literature has revealed that there are no good answers to solve this problem because of the complexity of the user models needed as well as the number of possible variations. Accordingly, Chapters 7 and 8 describe a much more compact system that has the desired scalability and extensibility. The examples in this chapter show that the implemented UMDA system has a compact user model for a project manager as well as compact user model/task profile to retrieve information for change order processing. The analysis on the algorithms and sample codes illustrate the feasibility and logic of the UMDA system. Some screen shots and sample results have also been presented to show that the UMDA system is capable of retrieving information for users according to their needs. Accordingly, the author has shown a more compact and scalable solution to this complex problem and has demonstrated that the system works. That solution also addresses the information problem discussed at the beginning of this study.

Chapter 9 discusses the UMDA design process; the proposed system will be validated and a design will be presented to configure the user interfaces and the application layer based on user modeling techniques.

CHAPTER 9
VALIDATION

In this chapter, the UMDA design process and the proposed system will be validated and a design will be presented to configure the user interfaces and the application layer based on user modeling techniques. The system behavior includes searching and retrieving information according to use and their task profiles. In the following sections, detailed data and analysis on the information retrieval results will be presented.

9.1 Validation Cases

9.1.1 Usability

The proposed system is a user–model-driven architecture. The intent is to help users configure the user interface and applications so that they can find the information they need and execute the functions that best suit their work and tasks. The usability of the UMDA system will be assessed by its ability to accept some query words that are close to natural language and are within the construction domain.

As shown in Table 9-1, each row represents a user query. It also has the information on the type of the user who asked the query and the type of the task the user wanted to perform. The query strings in the UMDA system use keywords that are close to the language construction people use.

9.1.2 Sensitivity Analysis

The sensitivity of the UMDA system is a measure of whether the system can find out the mistakes or misspelling in the user's query automatically. The implemented

Table 9-1 Examples of user queries

Query No.	User Type	Task Type	Query String	System Parsed String
1	PM	1	2 floor hand rail	“2” “floor” “hand” “rail” “hand rail”
2	PM	3	Entrance door specification	“entrance” “door” “specification” “entrance door”
3	PM	5	Drywall subcontractor	“drywall subcontractor” “drywall”, “subcontractor”
4	PM	7	Truss delivery	“truss delivery” “truss” “delivery”
5	PM	9	Inspection rebar	“inspection rebar” “inspection” “rebar”
6	PM	11	Quantity takeoff rough lumber	“Quantity takeoff rough lumber” “quantity” “takeoff” “rough lumber”
7	PM	12	Subcontractor name list	“subcontractor” “name” “list”
8	Super.	7	Material delivery dates	“Material delivery dates” “Material” “delivery” “dates”
9	Super	10	Ceiling tile value engineering	“ceiling tile cost” “ceiling” “tile” “cost”
10	Super	11	Total project cost	“total” “project” “cost” “total project cost”
11	Super	12	Feasibility report	“reasibility” “report”
12	Super	14	Hard hat	“hard hat” “hard” “hat”
13	Estim.	11	Electrical panel	“electrical, panel” “electrical Panel”
14	Presid.	2	Glass subcontractor	“glass subcontractor” “glass” “subcontractor”
15	Office Mgr.	3	Project cost	“Project cost” “Project” “cost”

Explanation of User Types:

PM—Project Manager Super.—Superintendent Estim.—Estimator
Presid—President Office Mgr. – Office Manager
Explanation of Task Type:
1--Change_Order; 2--Close_Out_Administration 3--Competitive_Bidding
4--Estiblish_Cost_Model_and_Develop_Cost_Extimates 5--Identify_Trade_Contracts
6--Maintain_Project_Site_Document 7--Material_Management
8--Perform_Constructability_Review 9--Perform_Quality_Control_Inspection
10-Perform_Value_Engineering 11-Preparing_Budget
12-Pre_Qualification 12-Project_Scheduling
13-Provide_Project_Staffing 14-Safety
15-Shop_Drawing_Review 16-Submittal_Process

UMDA system will give immediate suggestions to user's query. Chapter 8 has presented the algorithms and the sample code used to illustrate the functions of the query parser and the algorithms used. The UMDA system implements the Edit Distance approach to compare the similarity of two words. Figure 9-1 shows screen shots of the UMDA system asking the user if they would like to replace the query word with the suggested word. Figure 9-1 (A) shows that a user misspelled the word "prywood". The UMDA system suggested that the word "plywood" be used. Figure 9-1(B) shows that the user accepted the suggestion and the system put the word "plywood" into Query Parser Result field. Figure 9-1(C) shows how for queries with multiple words, the UMDA system will confirm each of the suggestions with the user and it includes two misspellings: one is for insulation; the other is for shingles.

Table 9-2 lists the time the UMDA system takes to find the suggestions presented to users. It also lists the time elasped for the system to specify or extend the search scope by using the CSI tree structure. The computer system the author used to test performance has the following configuration: Microsoft Windows XP Professional, Version 2002, Service Pack 2, with 2.00 GHz Intel Pentium Processor and 1 GB of RAM and a 30-GB HD. This is a regular configuration for most current computer systems. The intent behind using the performance metrics is not to judge the speed of query processing but just to demonstrate that the UMDA approach works and computes in a reasonable timeframe.

Using standard statistical methods (Ott and Longnecker 2001) to calculate the mean and standard deviation, the following values are obtained:

Mean of the Performance Time of System Suggestions = $\bar{X} = \frac{(\sum x)}{n} = 92 / 15 = 6.133$

Standard Deviation of the Time = SD = $\sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$ = 2.4746

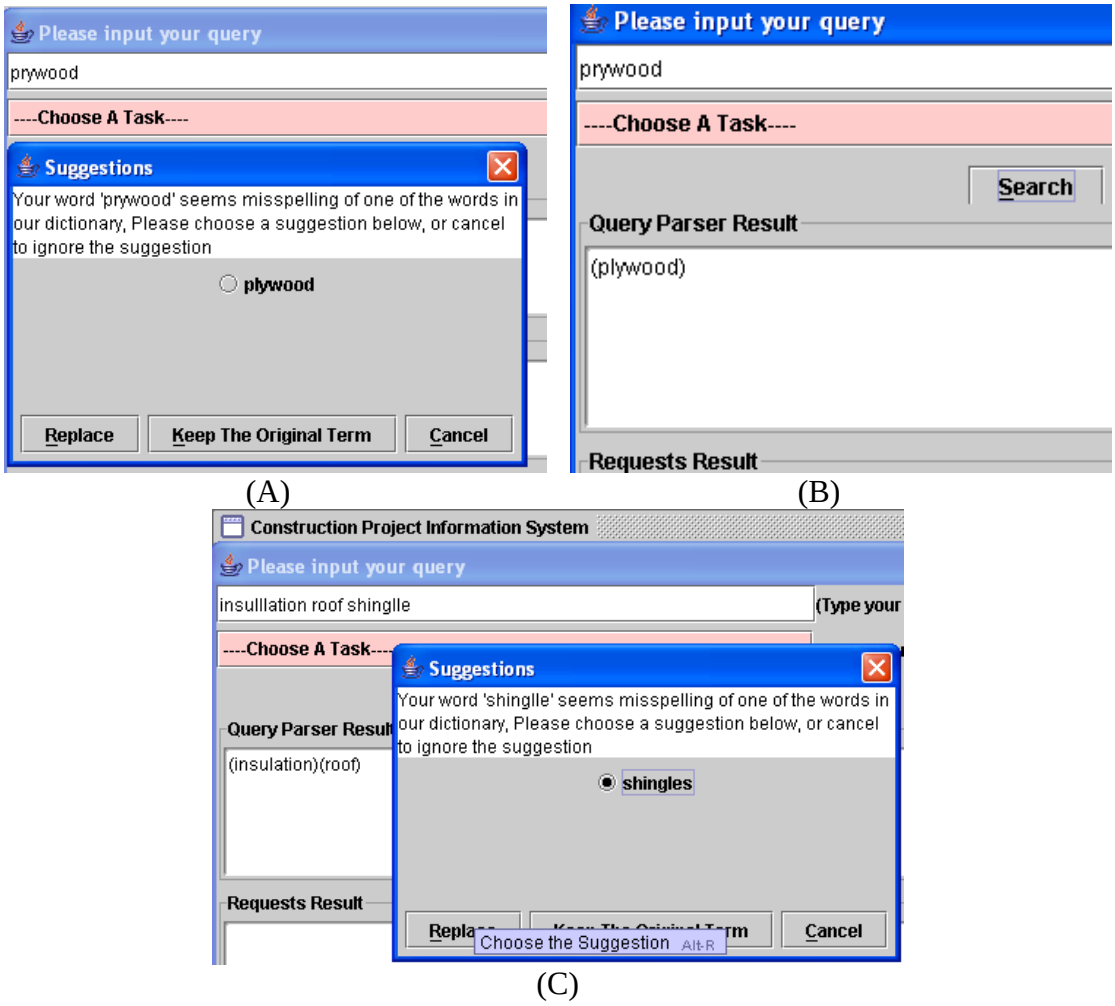


Figure 9-1 Screen shots of UMDA system suggesting a word replacement. A) Shows that a user misspelled the word “prywood”. The UMDA system suggested that the word “plywood” be used. B) Shows that the user accepted the suggestion and the system put the word “plywood” into Query Parser Result field. C) Shows that the UMDA system will confirm each of the suggestions with the user.

According to (Ott and Longnecker 2001), it is appropriate to use probabilities related to standard deviations for a Normal distribution and to assume the distribution is reasonably symmetrical about the mean. According to the Standard Normal distribution

Table (Ott and Longnecker 2001), at the 95% confidence level, the range of times elapsed is given by:

$$6.133 - 2 \text{ X } 2.475 = 1.183 \text{ ms to } 6.133 + 2 \text{ X } 2.475 = 11.083 \text{ ms}$$

Table 9-2 Time elapsed for the UMDA system to find suggestions

Query String	Misspelling Correction	CSI Extending	User Accepted?	Time Elapsed
Ocncrete	concrete	(Basic Concrete Materials,Methods or Concrete Forms,Accessories or Concrete Reinforcement or Cast-In-Place Concrete)	Yes	12 ms
electricaal	electrical	(Basic Electrical Materials,Methods or Wiring Methods or Electrical Power or Transmission,Distribution)	Yes	6 ms
mechinacal	mechanical	(Basic Mechanical Materials,Methods or Building Service Piping or Process Piping or Fire Protection Piping)	Yes	9 ms
aggregated	aggregate	N/A	Yes	5 ms
Girlder	girder	N/A	Yes	5 ms
Rddiator	radiator	N/A	Yes	5 ms
windows	N/A	(Basic Door,Window Materials,Methods or Metal Doors,Frames or Wood,Plastic Doors or Specialty Doors)	Yes	10 ms
tunneling	N/A	(Foundation,Load-Bearing Elements)	Yes	7 ms
Finishes	N/A	(Basic Finish Materials,Methods or Metal Support Assemblies or Plaster,Gypsum Board or Tile)	Yes	7 ms
infiltration	N/A	N/A	No	4 ms
Lintel	N/A	N/A	No	4 ms
Drywall	N/A	N/A	No	3 ms
Theaet	No Match	No Suggestion	User kept the word	5 ms
9dfja	No Match	No Suggestion	User kept the word	5 ms
Ldjy	No Match	No Suggestion	User canceled	5 ms

Hence, at 95% confidence level, the Performance Time of System Suggestions is within the range of (1.183 ms to 11.083 ms). The timeframe is within an acceptable range. Since the goal of this research is construction information technology research, improving the system execution time a few seconds or milliseconds is not the design purpose. However, the system should not be so slow that it would take hours or days to find results, which is the reason why preceding sensitivity analysis exercise has been conducted.

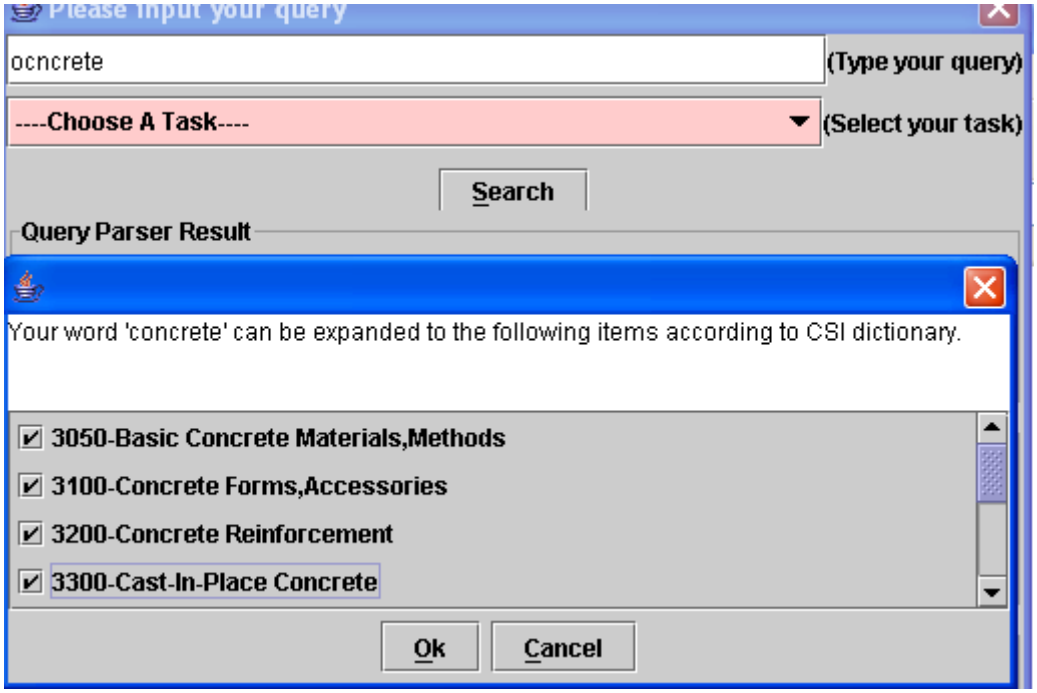


Figure 9-2 Example of the UMDA system extending the user search scope

9.1.3 System Execution Time

With the implementation and execution of the UMDA system, will the information retrieval performance be affected? To answer this question, the author suggests using some computer system and some computer project data or files to test the UMDA system. Appendix E shows the detailed analysis and conclusion of the system performance in information retrieval.

The execution time of the system with UMDA design shows that the proposed system has acceptable performance in terms of execution time. Strict speaking, the assessment shown in Appendix E is not a performance evaluation. To validate the performance of the UMDA system, too many factors would need to be taken into consideration. This is outside the scope of this research. This research is on construction information technology and pure improvement in system performance time is not the design target. The purpose of the evaluation of information retrieval performance is to show that the system has a reasonable execution time.

Standard information retrieval precision recall techniques (i.e. compare the list of all possible answers retrieved manually with the list of all possible answers retrieved by EXTET) were used. Then it was determined how many of the items in the list of all possible answers retrieved by EXTET are not related to the question (i.e. are trash). The goal is to retrieve all relevant records and no unwanted records. Appendix F has the literature review and analysis on the UMDA model validation process. Table E-1 and E-2 in Appendix E have detailed explanations of the design of the Test Modules (TM). Table 9-3 shows the number of SQL queries generated by the UMDA system, the number of results retrieved for the requirement of the query, the number of relevant records retrieved, the number of relevant records not retrieved, and the number of irrelevant records retrieved.

Figure 9-3 shows the relationship between the recall and precision measurements for the Test Modules. According to (Jizba 2000; Tsuruoka and Tsujii 2004), based on the results shown in Figure 9-3, the UMDA design is acceptable in terms of recall measurements. The precision measurements of the UMDA design show that the

implementation reaches the design goal. A precision-recall break-even point is a hypothetical point at which precision and recall are the same. As shown in Figure 9-3, the break-even point is at P=82.05% and R=68.09%. Before the break-even point, recall value is greater than precision value; after the point, recall value is less than precision

Table 9-3 Using Precision/Recall to validate system performance

TM No.	SQL	Ma	A	B	A+B	Recall= A/(A+B)	C	A+C	Precision= A/(A+C)
TM1_1	6	62	54	45	99	0.5455	8	62	0.8710
TM2_1	14	23	10	4	14	0.7143	13	23	0.4348
TM3_1	5	53	39	32	71	0.5493	14	53	0.7358
TM4_1	17	78	78	30	108	0.7222	0	78	1.0000
TM5_1	10	39	32	15	47	0.6809	7	39	0.8205
TM6_1	8	32	23	19	42	0.5476	9	32	0.7188
TM7_1	24	9	9	8	17	0.5294	0	9	1.0000
TM8_1	29	181	167	129	296	0.5642	14	181	0.9227
TM9_1	32	386	367	283	650	0.5646	19	386	0.9508
TM10_1	20	90	33	10	43	0.7674	57	90	0.3667
TM11_1	27	467	438	276	714	0.6134	29	467	0.9379
TM12_1	23	432	368	158	526	0.6996	64	432	0.8519
TM13_1	43	269	235	176	411	0.5718	34	269	0.8736
TM14_1	21	30	27	15	42	0.6429	3	30	0.9000
TM15_1	16	66	53	9	95	0.5579	13	66	0.8030
TM16_1	17	63	45	20	65	0.6923	18	63	0.7143

SQL – the number of SQL queries= the UMDA system commands generated from EXTET structure.
Ma – the number of results retrieved for the requirement of the query
A – the number of relevant records retrieved
B – the number of relevant records not retrieved
C – the number of irrelevant records retrieved. Here, (A+C) = Ma

value. Figure 9-3 also shows the regression line of the points and the trends of the points. In most circumstances, precision and recall have an inverse relationship. When precision increases, recall decreases; and vice versa. According to (Soboroff 2004; Ibushi, et al. 2004; Cho and Richards 2004), an average precision of 25% is acceptable in most cases; and an average precision at 50% is good. For recall values, if the value is greater than 50%, it is acceptable. Table 9-3 shows that the average-recall of the tested queries is

62.27%; the average-precision of the tested queries is 80.64%, which is good for precision in the searched results. Some researchers (Herbrich, et al. 2004; Tsuruoka and Tsujii 2004) also used Harmonic Mean (F Measure) of recall and precision and E Measure (parameterized F Measure) to measure performance that takes into account both recall and precision. Using P for precision and R for recall, the calculations for the F-Measure and the E-Measure are as follows:

$$F = \frac{2 * P * R}{P + R} = \frac{2}{1/R + 1/P} \qquad \text{(F-3)}$$

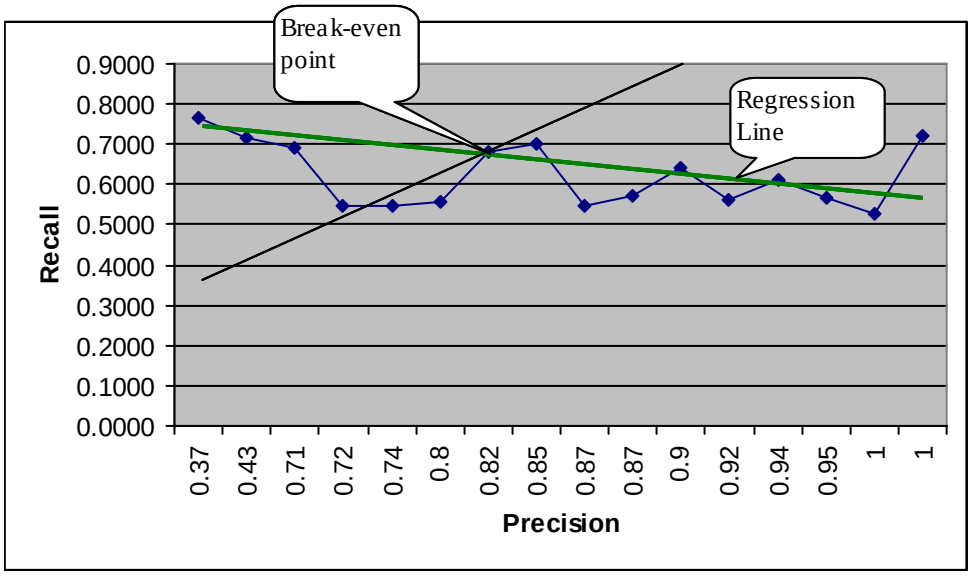


Figure 9-3 Recall-Precision relationship in the test modules

$$E = \frac{(1+\beta^2) * P * R}{\beta^2 * P + R} = \frac{(1+\beta^2)}{\frac{\beta^2}{R} + \frac{1}{P}} \qquad \text{(F-4)}$$

The value of β controls trade-off.

If β = 1: Equally weigh precision and recall (E = F).

If β > 1: Weigh precision more.

If β < 1: Weigh recall more.

The F-Measure is independent from β. The average F-Measure is 0.3513 and for β = 2, the average E-Measure is 1.34.

The above statistical results and analysis follow the general process used in recall-precision analysis. As can be observed in Table 9-3, some measurements are relatively low compare to other results, for example, the recall measurement in Test Module 7_1 (TM7_1) and the precision measurement in TM10_1. To find the reason(s) for the difference in those test results in Table 9-3 and the low measurements in some cases, it is necessary to perform detailed analysis. By analyzing the test modules developed for Table 9-3, the author determined that those test modules are complex and need to be simplified. To analyze the recall/precision measurements, the search queries need to be carefully selected. The goal of a search in the UMDA system is a comprehensive retrieval of information. The searcher may include synonyms, related terms, broad or general terms in each query. One problem with the comprehensive retrieval goal is that broader terms may be included in the retrieval of material, which may not match the narrower search topic. Because synonyms may not be exact synonyms the probability of retrieving irrelevant material increases. As a consequence precision may suffer. As noted earlier, records must be considered either relevant or irrelevant when calculating precision and recall. Obviously records can exist which are marginally relevant, somewhat irrelevant, very relevant, or completely irrelevant. This problem is complicated by individual perception. What is relevant to one person may not be relevant to another (Jizba 2000).

That is why in the next step to validate the UMDA system, a refined evaluation process has been designed. In the refined process, 10 different users (with different construction backgrounds, experiences, educations, etc.) were invited to test the system.

These 10 users were randomly divided into 2 groups. One group (Group A) was provided with paper-based materials about a project. The paper-based materials (including drawings, specifications, budget estimating, scheduling, etc.) had exactly the same info as that stored in the UMDA construction project database. The other group (Group B) was given access to the UMDA system and the construction project database. The author acted as observer. The 10 users in both Group A and Group B were provided with the same question list. The 5 Group-A users were using paper-based materials and manually searching for information. They were asked to follow their regular steps in searching for information. The 5 Group-B users received a 10-minute demonstration of the functions available in the UMDA system.

For confidentiality purposes, the 10 users were designated as A1, A2, A3, A4, and A5 to represent the 5 users in Group A and B1, B2, B3, B4, and B5 to represent the 5 users in Group B. Each user was asked the following 5 questions:

Question 1: Door #2 delivery

Question 2: Window competitive bidding

Question 3: Plumbing fixture change order

Question 4: Steel structure shop drawing review

Question 5: Stucco work scheduling

Tables 9-4 and 9-5 show the observations of the validation processes by the 10 users and lead to the following conclusions:

- Comparison of the user who is familiar with a project with the user who is not indicated that the user with familiarity has better precision and recall results.
- Comparison of the results shows that the UMDA system has precision and recall results that are very close to those achieved with manual searches.

Table 9-4 shows the precision/recall indices of the performance of the manual-search group. The manual-search group is used as a control group for Group B, which uses the UMDA system to search for information. For analysis purposes, 80% was used in both precision and recall indices as the “gold standard” for this validation. The “gold standard” reflects the level of performance needed to for a classification as good. The low precision or recall results in Table 9-5 show that the system did not pull the correct answers for those cases. In the above sample questions, Question 1 relates to Case Study 1 material handling and management in Chapter 5.

Tables 9-4 and 9-5 can be used to compare Questions 1 and 2 and with different persons testing the same query. One person used the manual searching method; the other used the UMDA system to retrieve data from the project database. The results recorded in Table 9-4 show that the person who used manual searching achieved 100% in both recall and precision measurements. There are 12 relevant records found from the hard copies of the project data. The time spent in searching for the data is 8 minutes 43 seconds. The person in this test is familiar with the project.

In Table 9-5, the UMDA system was used to search for the project data. The UMDA system found 17 records. After analysis, 12 out of the 17 records were found to be relevant. It is known that there are 12 records in the project database that are relevant to Question 1. The retrieved 12 relevant records match with the 12 records in the database that should be retrieved. The 5 irrelevant results in the 17 retrieved records are all Excel spreadsheets. After tracing back to the task profile for material management for Question 1, the author found out that that task profile specifies using filename- matching to search. Those 5 irrelevant results are contained in files with ambiguous filenames. The UMDA

system considers those 5 filenames relevant to Question 1. The test for Question 1 shows that the use cases are not complete and specific enough. With the further development of user research in the construction industry, use cases with more accurate description of the

Table 9-4 Observations of Group A (manual-searching) validation

	User A1	User A2	User A3	User A4	User A5
Age	30-40	30-40	50-60	20-30	40-50
Education	College	College	High School	High School	College
Experience	3 years	5 years	35 years	2 years	20 years
Background	PM	PM	Superintendent	Assistant Superintendent	Senior PM
Query	1	2	3	4	5
Familiarity with the project?	Yes	No	No	No	No
Time spent	8 minutes: 43 seconds	32 minutes: 05 seconds	12 minutes: 32 seconds	15 minutes: 45 seconds	3 minutes: 17 seconds
Records found	12	18	11	15	7
Relevant records in the documents	Observers discussed with User A1 and went through the entire documents together. There are 12 relevant records in the docs.	Observers discussed the records with User A1 and A2 and checked the entire documents. There are 23 relevant records in the docs.	Due to time limitation, A3 left without discussion. Observers discussed with User A1 and found 27 relevant records in the docs.	A4 insisted that there are15 relevant records in the entire document. Observers and A1 found 19.	Observers, A1, and A5 agreed that there are 15 relevant records in the docs.
Relevant records in the retrieved documents	Observers and A1 agreed that there are 12 relevant records in retrieved docs.	A1 suggested 16 relevant. Observers and A2 suggested 17 relevant.	Observers and A1 suggested 11 records relevant.	Observers suggested 12 relevant. A1 and A4 suggested 14 relevant.	Observers, A1, and A5 agreed 7 records relevant.
Precision	100%	88.9%	100%	80%	100%
Recall	100%	78.3%	40.7%	78.9%	46.7%

Table 9-5 Observations of the Group B (UMDA system) validation

	User B1	User B2	User B3	User B4	User B5
Age	20-30	30-40	50-60	50-60	30-40
Education	High School	High School	College	High School	College
Experience	2 years	15 years	31 years	8 years	12 years
Background	Assist PM	PM	Superintendent	Superintendent	PM
Query	1	2	3	4	5
Familiar with the project?	Yes	No	No	No	No
Time spent	1 minutes: 43 seconds	3 minutes: 05 seconds	1 minutes: 32 seconds	3 minutes: 45 seconds	2 minutes: 17 seconds
Records found	17	29	25	20	14
Relevant records in the documents	Observers discussed with User A1 and went through the entire documents together. There are 12 relevant records in the docs.	Observers discussed the records with Users A1 and A2 and checked the entire documents. There are 23 relevant records in the docs.	Due to time limitation, A3 left without discussion. Observers discussed with User A1 and found 27 relevant records in the docs.	A4 insisted that there are15 relevant records in the entire document. Observers and A1 found 19.	Observers, A1, and A5 agreed that there are 15 relevant records in the docs.
Relevant records in the retrieved documents	12	22	24	17	13
Precision	70.5%	75.9%	96%	85%	92.9%
Recall	100%	95.7%	88.9%	89.5%	86.7%

real construction work/management process will emerge. The UMDA system will be able to generate more precise system commands from the more accurate use cases. Question 3 relates to the Case Study 2 Request For Information process in Chapter 5. Due to the limitations of the construction project database that was used to test the UMDA system, certain user queries may not have completed answer results or no results at all. For the

above 5 queries, the UMDA system retrieved enough data for the precision/recall analysis.

A few of the reasons for these failures may be:

- The use cases are not complete enough. For example, the Request For Information process should be added to the profile database. This will improve the power/accuracy of the results for sample question 3. With the further development of user research in the construction industry, more precise use cases will become manifest. This will help to resolve this problem.

- Sample question 4 is a complicated query. It takes into consideration building codes, engineer design, and other criteria to build up the task profile. The low precision number shows the complexity of the work.

- Word mismatch may also cause bias in the searched results. Future user model systems can build more intelligence into them and have the systems "understand" the meaning of query.

Table 9-3 shows the retrieved results and recall/precision measurements based on 16 general-purpose queries. The advantage of using Table 9-3 is that allows a quick build up of the approximate idea of the recall/precision indices. It also leads to more specific design and analysis on the system performance. The disadvantage of Table 9-3 is that the queries are too complex to derive meaningful conclusions. Tables 9-4 and 9-5 were built based on the observation on the results shown in Table 9-3. The limitation of Tables 9-4 and 9-5 is that the number of test users and test queries are small. The achievement of using those two tables is that the queries are refined. The results shown in Table 9-4 and

9-5 are straightforward and can be traced back to different parts of the UMDA system. The results can help in finding out which part of the UMDA system needs to be improved.

9.2 Conclusion

Chapter 9 has discussed the evaluation strategy of the implemented UMDA system. The focus of the strategy is on the whether the UMDA system provides users with the information they need; whether the system keeps consistency of the information; and whether the system is efficient in retrieving and displaying the information to users. Based on this strategy, five validation cases were designed. These validation cases tested the UMDA system from the aspects of user involvement and satisfaction, usability, sensitivity, search scope, and information retrieval performance.

The execution time was recorded and the related statistical indices were calculated in Appendix E. The execution time of the system with UMDA design shows that the proposed system has acceptable performance in terms of execution time. The UMDA system implemented user profiles and task profiles to configure the GUI layer and the application layer of the information retrieval system. To validate the performance of the UMDA system regarding its quality of search results, standard information retrieval precision/recall techniques were used. The UMDA design is acceptable in terms of recall measurements. The precision measurements of the UMDA design show that the implementation reaches the design goal. The goal of a search in the UMDA system is comprehensive retrieval. As further described in Appendix F, some users were invited to test the system. The feedbacks from these users indicated that who was familiar with a project has better precision and recall results than those who were not. A comparison of

the results also showed that the UMDA system has precision and recall results that are very close to those achieved with manual searches.

The validation process has demonstrated that the UMDA system provides, in all the considered cases, the needed representations that are usable and that it supports effective data access and data collection of project management information. It also demonstrated that the user interaction style of the UMDA system allows in many cases for efficient user interaction.

Chapter 10 discusses the conclusions made based on this discussion and recommendations will be made for future research.

CHAPTER 10
CONCLUSIONS AND RECOMMENDATIONS

10.1 Conclusions

The scientific focus of this research lies in the investigation and advancement of user modeling and the use of case study methodologies for the development of domain specific services. Those are achieved by observing and experiencing the practical work of the users in the construction industry and testing theoretical developments from software products and formal use cases. The research questions investigated in this study are as follows:

- What are the constituent components of user models for construction project management?

  1.1 What user model can be built based on the detailed case studies and literature reviews?
  1.2 How do we group users into different classes and what will the features of each of the user classes be?
  1.3 How does the system judge the user preferences based on the user interaction with the system?
  1.4 How do we present the search results to the users?

- How does user modeling direct query processing and result fusion?

  2.1 How do we utilize the existing functionalities from AutoCAD, P3/Microsoft Project, Timberline/Excel, and other existing software systems that are popular in the construction industry?
  2.2 How do we utilize the design ideas from current user modeling projects reviewed in this research?
  2.3 Using the user model as a direction, can the functionalities of the system be developed?

These two questions are further developed into a couple of detailed questions. For Question 1.1, based on the case studies and literature reviews, a formal user model as

shown in Tables 8-2 and 8-4 in Chapter 8 has been developed. For different groups of use cases, different formal user models need to be built. These formal user models reflect the ordinary work procedures of the staff working in construction companies. The formal user models are very easy to understand and maintain and they are domain specific, which means for the people working in the construction industry, the content in the models will be about construction.

For Question 1.2, users are grouped into different user classes, according to their different job responsibilities. For example, this research is about construction management, the user classes include project managers, superintendents, vice president, and estimators. Different user classes have different preferences. In the case studies in Chapter 5, it was shown that at the same time, for the same project, with the same question, different users have different expectations and requirements for their search results. This study focuses on two classes of users: Project managers and Superintendents. From the case studies and the formal user models, it was determined that these two classes have a lot of differences, especially in their needs for information retrieval.

Question 1.3 is about the user preferences and how the system will know about those preferences. When a user logs into the system, the system will check his or her ID and password. From the person's ID the system will be able to find out which user class this person belongs to. Then, when the user types a query and requests information from the system, the system will be able to compare and match the task the user wants to perform by parsing the query. In case the user does not specify any task information, the system will access the project database and send back as much information as it can. If

the user submits the task information to the system, the system will invoke the Task Analysis Module to find out what kind of information this user needs for that task.

Construction users are familiar with drawings so 2D/3D graphical presentations will be appropriate for the information on the drawings. For all other information, for example, contractor information, a tree structure (directories, folders, and files) will be a good option to organize documents. For scheduling data, bar charts will be the right choice. More examples of this can be found in the case studies in Chapter 5.

Question 2.1 is about the connection between the proposed system and the database management system. As discussed in Chapter 6, ElementAmbassadors and Navigational Models will be used to search for and retrieve information. In addition, the existing functionalities from AutoCAD, P3/Microsoft Project, Timberline/Excel, and other existing software systems that are popular in construction industry will be used. In order to utilize them certain function calls provided by Java programming language will be used

The design ideas derived from current user modeling projects are mainly about the design process. The primary target of using user modeling in design is to get users involved in the design procedure and to get early feedback and evaluation of the product. That is a very important design principle. The proposed system uses user modeling as part of the system. Case studies and use cases are transformed into formal user models. Then user profiles and task profiles could be generated from these formal user models and loaded to Task Analysis Module of the UMDA system to help retrieve project information.

The Answer to Question 2.3 “Using the user model as a direction; can the functionalities of the system be developed?” is yes. The former has been used to directly build up the functions of the system. For more details, please refer to Figures 8-3 and 8-4 in Chapter 8.

As a construction project progresses, more and more information is added to the system. Data storage will become huge. The biggest challenge is to make it easy to access and manage data. This is accomplished through the automation of all the tasks of data organization, accesses, and protection.

Chapter 7 and Chapter 8 discussed the conceptual design and implemented the UMDA system. The UMDA system was used to retrieve information for queries (i.e. as shown in Appendix E). The execution time was recorded and the related statistical indices were calculated. The execution time of the system with UMDA design shows that the proposed system has acceptable performance in terms of execution time. The UMDA system implemented user profiles and task profiles to configure the GUI layer and the application layer of the information retrieval system. To validate the performance of the UMDA system regarding its quality of search results, standard information retrieval precision/recall techniques were used. Using the Test Modules developed in Appendix E, the author analyzed the relationship between the recall and precision measurements in the Test Modules. The UMDA design is acceptable in terms of recall measurements. The precision measurements of the UMDA design show that the implementation reaches the design goal.

The goal of a search in the UMDA system is comprehensive retrieval. As further described in Appendix F, some users were invited to test the system. The feedbacks from

these users indicated that who was familiar with a project has better precision and recall results than those who were not. Comparison of the results also showed that the UMDA system has precision and recall results that are very close to those achieved with manual searches.

User model acquisition is a difficult problem. In Machine Learning, the information available to a user modeling system is often limited, and it is hard to infer assumptions about the user that are strong enough to justify non-trivial conclusions. In Information Retrieval, user models have been limited to lists of terms relevant to an information need. In this research, the UMDA system was built based on interviews with prospective users, case studies found in literature and personal industry experience. Through the analysis of the use case of the construction material handling process, the feasibility of involving users in the life cycle of software system development was explored. The final Java classes were built up based on the use case → user model → Java code evolvement. The users are able to evaluate the temporary result system and based on their feedback, the use case and user model can be changed and so is the software system.

10.2 Recommendations for Future Research

The next step in this research will include doing a more extensive user evaluation of the prototype system and adjusting the system based on their comments. In the future, research on an executable user model with semantics for actions should also be explored. One possible improvement to the proposed system is to use a graphical user interface. This will make it easier for users to locate the needed information. Another possible future work is to analyze the construction change order agent and to determine if it can be more intelligent? An intelligent Agent could also help find out the user's intention and speed up the information retrieval process. Methods and tools for system security,

identification, authentication, access control and encryption need to be applied for protecting user models.

The design process of the user/task combination table also suggests more work should be done to facilitate the generation of the combination table. In the current design of the EXTET structure of the UMDA system, the separated user profiles and task profiles can be generated automatically, but the combination table is done manually. The reason is that this combination involves expert knowledge and thus requires human involvement to specify and adjust the table. The current UMDA system implemented 17 task profiles and 10 user profiles. But for real industry software, the system would have more profiles. Imaging having 500 task profiles and 20 user profiles, to combine these profiles purely by hand would be a cumbersome work. For large profile database, which is a trend of future development, some techniques need to be found to ease the combination job. The following suggestions will provide the intuitions to start resolving the problem:

- Develop a learning tool in the combination mechanism to facilitate the process. By using neural network or other artificial intelligence (AI) tools, the system could be trained to learn while human experts combining profiles. Human involvement is unavoidable. But the system can learn to make suggestions or recommendations to human experts. This would help to save time in first generating the combination table.

- Encode more knowledge in the user profiles. For example, Project Managers tend to read more types of data, such as drawings, estimating spreadsheets, and contracts. This type of information could be built into the attributes in the Project Manager's user profile. But surveillance needs to be taken to make sure that the complete separation of user profiles and task profile will not be violated.

Some of the topics that need to be addressed by future research are:

(1). Automatic classification

Large document collections can be handled successfully by means of automatic classification. This will also boost commercial interest and along with it the support for further development.

(2) Search strategies

Probabilistic search strategies have been used in the fields of pattern recognition and automatic medical diagnosis. They have great potential for information retrieval applications.

(3) Network Support and Scalability

The design of the proposed User Model Driven Structure is a single user system. It does not need to consider the problems caused by networks (such as network security, multiple users' login, file consistency, bandwidth, etc.). But since a lot of projects are located in places remote from the headquarters or the local offices of a construction company, Network Support has to be a research topic for the proposed system.

(4) Content analysis

Content analysis requires the system to be able to understand the semantic meanings of the user instructions. The proposed User Model Driven Architecture (UMDA) is based on keyword search. It does not have the ability to understand the strength of relationships between keywords or between the keyword descriptions of documents. The content analysis attempt has to use natural language to represent documents inside a computer. There is reason for optimism now that a lot more is known about the syntax and semantics of language.

Other topics for future research include:

- Simplifying the queries to allow just one keyword or combination of keywords input. This research relates to content analysis. It requires the software system to "understand" the relationship between keywords.
- Populating the nodes with all the equivalent words at system creation time. This will save time in system operation. In addition, when user types a query, the system will be able to save the entire equivalent words with this query in a certain table. Next, when the same user types the same query, the system will generate nodes with all the equivalent terms at its creation time.
- Exploring rule-based approaches on how to display information. The system will observe user patterns. Based on these observations obtained before from a single user and neural network based forward model analysis, the system will build up corresponding user profile for that user. One implementation of these rules in the user profile is to decide what type of display method the system will use to display information to user.
- Scalability of users and data sources. User scalability means the system can handle multiple users. This is similar to the network support mentioned before. Data source scalability means the system will be able to incorporate more data sources into the system without change the majority of the system program. Since Element Ambassador and Navigation Model are implemented in the UMDA system, the system has the capability to integrate heterogeneous data sources.
- Building up a complete construction use case database. As the analysis in Chapter 9 indicates, the incompleteness in use case database affects the performance of the

entire system. A full-fledged use case database will help to improve the accuracy of the results retrieved by the system.

GLOSSARY

ACM SIGCHI	The Association for Computing Machinery Special Interest Group on Computer-Human Interaction
Adaptable system	In adaptable system, user changes (with substantial system support) the functionality of the system.
Adaptive system	In adaptive system, dynamic adaptation is performed by the system itself to current task and current user.
Adaptive Systems and Interaction group (ASI)	The Adaptive Systems and Interaction group (ASI) of Microsoft pursues research on automated reasoning, adaptation, and human-computer interaction. ASI is at the center of user modeling at Microsoft Research, focused on inferring the goals and needs of users from multiple sources of information about activity and interests.
Application Program Interfaces (APIs)	APIs are standard routines that can be called from within metadata access programs, remaining in that program's control.
Bayesian formula	Bayesian formula has been used extensively in statistics and probability. It is a useful equation from probability theory that expresses the conditional probability of an event A occurring, given that the event *B* has occurred (written P(A|B)), in terms of unconditional probabilities and the probability the event *B* has occurred, given that *A* has occurred. In other words, the Bayes formula inverts which of the events is the conditioning event. The formula is $P(A \mid B) = \frac{P(A,B)}{P(B)} = \frac{P(A) * P(B \mid A)}{P(B)}$
CADD	Computer-Aided Drawing/Design
CAE	Computer Aided Engineering
CIC	Computer Integrated Construction. CIC can be defined as a business process that links the project participants in a facility project into a collaborative team through all phases of a project.
Collaborative User Experience (CUE)	IBM's Collaborative User Experience (CUE) Research group, formerly Lotus Research, is a subdivision of the IBM Watson Research Center and is doing research with emphasis on the

	interaction between people and computer systems in support of collaboration. Their research themes are Activity Management, Large-Scale Collaborations, Information Visualizations, and Collaborative Development Environments.
Common Gateway Interface (CGI)	The Common Gateway Interface (CGI) is one of the most powerful methods of providing dynamic content on the Web. CGI is a generic interface for calling external programs to crunch numbers, query databases, generate customized graphics, or perform any other server-side task.
CORBA	Common Object Request Broker Architecture. A request broker sits as the basic mechanism for receiving and channeling requests. The Interface Definition Language (IDL) is the standard language required for these requests, and is independent of programming language, but contains mapping to popular programming languages so that applications can send and receive requests to CORBA facilities. The overall architecture defines the structure, interfaces services, and objects.
CPM	Critical Path Method
CUA	Common User Access
Data retrieval system	A data retrieval language aims at retrieving all objects that satisfy clearly defined conditions such as those in a regular expression or in a relational algebra expression. Thus, for a data retrieval system, a single erroneous object among a thousand retrieved objects means total failure.
DBMS	Database management system
DCE	Distributed Computing Environment
Derived data	Derived data means information inferred from observations and assumptions through inference procedure directly managed by the application.
ebXML	Electronic business XML
ECA command	ECA command is the extended create trigger statement which includes event definition
Embedded processes	Embedded processes are associated with distinct components of a standard meta model.

Extensible Markup Language (XML)	XML is a technology that attempts to address the areas associated with categorizing Web content. Briefly, XML allows previously unstructured organizational data (e.g., internal reports and email messages) to be categorized automatically with rudimentary, searchable metadata structures. These metadata structures are probably the first to separate style from content.
Formal information systems	Formal information systems are defined widely, as any device, product or system intended for information representation, storage, conservation, retrieval, or re-packaging.
Formal user model	Formal user model refers to the user model that itself is part of a software system or that can be converted into software systems without loosing its integrity and consistency.
Human-computer interaction research	A fundamental objective of human-computer interaction research is to provide users with experiences fitting their specific background knowledge and objectives. It is called customized software design.
I-AIM(sm) (Information About Information Markets)	Outsell (2002) did the research on I-AIM(sm) (Information About Information Markets), which is a study of the information habits and preferences of 10 user groups segments in 20 industries.
IBM User-Centered Design (UCD)	The IBM User-Centered Design (UCD) approach has become an industry standard for building usability into the design of IT products in IBM. As Vredenburg, et al. (2003) stated "UCD is more than a paradigm for designing usable products and systems. It is a blueprint for the practical implementation of product and system design, encompassing all aspects of the design process, from the gathering of user requirements during the early conception of a product through all aspects of the product cycle, including testing, product shipment, and beyond." The six stages of the UCD design process are Market Definition, Task Analysis, Competitive Evaluation, Design and Walk-through, Evaluation and Validation, and Benchmark Assessment.
IFIP TC13	The International Federation for Information Processing Technical Committee
Industry-wide integration	This will allow any project participant to communicate electronically with any segment of the industry (e.g., owners, designers, suppliers, financiers, regulators, etc.). It would also ensure consistency across different projects.
Information access and retrieval	The general area of information access and retrieval aims at modeling, designing and implementing systems able to provide fast and effective content-based access to a large amount of information. The aim of

	such systems is to estimate the relevance of the documents to a user's information need.
Information retrieval system	For an information retrieval system, the retrieved objects might be inaccurate and small errors are likely to go unnoticed. The main reason for this difference is that information retrieval usually deals with natural language text, which is not always well structured and could be semantically ambiguous. On the other hand, a data retrieval system (such as a relational database) deals with data that has a well-defined structure and semantics. Data retrieval, while providing a solution to the user of a database system, does not solve the problem of retrieving information about a subject or topic.
Information Tunnels	According to Tannenbaum (2002), We are organized and often reimbursed based on the "tunnel" with which we are associated (e.g., accounting, product development, customer). It is only natural, therefore, that supporting information is in line with that perspective. Information tunnels are typically based on processing requirements. Even though the same data may be required in more than one tunnel, it is usually not easy to go from one tunnel to another. In any case, data is typically converted to tunnel-resident information, reflective of a limited set of requirements.
Inter-application integration	This involves combining the computer applications of one company into one integrated system. These applications can then share data and call each other's procedures.
Interface	The mediator between the user and the system. Usually interface design is also the most difficult part in system design," since the interface design must satisfy three conflicting requirements: an interface should be simple, it should be complete, and it should admit a sufficiently small and fast implementation." (Lampson 1983)
Inter-system integration	This exists when one company's applications integrate with those used by other project participants.
ISAPI	Internet Server Application Programming Interface
IT	Information Technology
JDBC	Java Database Connectivity. JDBC is used to connect to the SQL Server. Applications can request not only relational SQL commands but also user-defined commands through JDBC, and JDBC gets values from the SQL server, then the values are sent back to applications. The JDBC is the tool to connect the Java applications to the relational DBMS.

Knowledge engineering	Knowledge Engineering refers to the aspect of systems engineering, which addresses uncertain process requirements by emphasizing the acquisition of knowledge about a process and representing this knowledge in a Knowledge Based System (Wilson 1993). Knowledge Engineering can be used on text understanding and document preparation, Speech Understanding, Translation/Generation, Expert Systems, Search for documents/photos, and many other areas. Knowledge Engineering has relations with system engineering, Human Computer Interaction, and data engineering. Knowledge Engineering addresses the structure of complex but ill-defined processes where the solution to defining the process is to define the knowledge involved in the process explicitly in a knowledge-based system (KBS).
LAN	Local area network
Lumiere project at Microsoft Research	The Lumiere project at Microsoft Research was initiated in 1993 with the goal of developing methods and an architecture for reasoning about the goals and needs of software users as they work with software. At the heart of Lumiere are Bayesian models that capture the uncertain relationships between the goals and needs of a user and observations about program state, sequences of actions over time, and words in a user's query (when such a query has been made) (Horvitz et al., 1998).
Maslow's Hierarchy of Needs	Maslow's Hierarchy of Needs divides human (i.e. user) needs into several classes, which have been depicted as forming a pyramid from bottom to top according to the importance of the needs to human life. The very bottom level must be satisfied for people to sustain their physical life. The need for achievement, for self-expression, and self-actualization are realized providing the physiological needs are satisfied.
Metadata	Data is useless without metadata. Metadata is often defined as the "data about the data" and most organizations perceive it as representing descriptive information (element names, definitions, lengths, etc.) about the populated data fields.
NSAPI	Netscape Server Application Programming Interface
OAGIS	Open Applications Group Integration Specification
Object Linking and Embedding (OLE)	OLE links information from other applications and provides Client/Server operation for accelerated processing and SQL access for reporting on the project database.

Object Management Group (OMG)	Object Management Group (OMG) is founded in April 1989 by eleven companies. OMG began independent operations as a not-for-profit corporation. The OMG is structured into three major bodies, the Platform Technology Committee (PTC), the Domain Technology Committee (DTC) and the Architecture Board.
Object View Interaction Design (OVID)	The OVID method draws from software engineering and related tools. It OVID utilizes UML diagrams to precisely specify the key aspects of a user experience, such as the user objects, their properties and relationships, and views of those objects that enable performance of user tasks. OVID models the user experience and not limited to software offerings. In fact it has been used for hardware, software, and Web site design projects.
OLAP	On-Line Analytical Processing. At each stage in the Analysis-Based Decision-Oriented Processing chain, from the original sources to final uses, the data is refined like any other raw material destined for production. When it finally reaches the user, the information has been cleansed, integrated, aggregated, extended, and otherwise enhanced to the point where it is immediately useful for decision making. Unlike typical end-user applications, OLAP products are regularly called upon to process large amounts of periodically refreshed data.
OMG	Object Management Group
On Demand Innovation Services (ODIS)	Example research projects are Instant Collaboration and Reinventing Email. CUE is also involved in ODIS (On Demand Innovation Services) and developing consulting assets -- software and methodologies -- that support collaboration in customer engagements.
OODB	Object-oriented database. Information is kept on a hierarchical tree structure.
OODBMS	Object Oriented Database Management System
OSI	Open Systems Interconnection
P3	Primavera Project Planner
Partner Interface Processes (PIPs)	Partner Interface Processes (PIPs) are specialized XML-based dialogues that define how business processes are conducted between trading partners.
Remote Procedure Calls (RPCs)	RPCs are standard routines that are executed by metadata accessing programs, outside of that program's control.

Result fusion	In multi-database systems, results from different databases may be related. In such situations, we need to fuse the results to obtain a final one. Result fusion usually happens after system collecting the information for the same query from different resources. It is also call result integration.
RPC	Remote Procedure Call
SQL	Structured Query Language. Most database systems accept SQL command (insert, delete, update, select, etc.)
Standards-based Languages	Standards-based Languages are defined by standards organizations and available for use by metadata accessing programs, In many scenarios, these languages are combined with interfaces and required architectures (Tannenbaum, 2002).
Static content filter	The static content filter refers to a filter, which is system environment independent or user independent. Once the rules of the content filter mechanism have been set, they will not change when users change or when the users' tasks change. So the static content filter mechanism does not have the ability to adapt to the change of users and the change of user tasks.
The General User Modeling System (GUMS)	The General User Modeling System software allows programmers of user-adaptive applications to define simple stereotype hierarchies. For each stereotype, Prolog facts describing stereotype members; and rules prescribing the system's reasoning about them (Finin, 1989; Finin and Drager, 1986).
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
Universal Description, Discovery and Integration (UDDI)	Universal Description, Discovery and Integration (UDDI) specification--could facilitate ad hoc relationships by allowing companies to describe their services in a central registry so they can be discovered and utilized by other businesses.
Usability test	Kuniavsky (2003) listed four major steps in the process of conducting a usability test: Define the audience and their goals; Create tasks that address those goals; Get the right people; and Watch them try to perform the tasks.
User	User is a term in many communication/information contexts. According to Wilson (2000), user can be seen as communicator, drawing upon personal or organizational information resources in communicating with organizational colleagues or fellows in society.

	In seeking the information needed, user can be identified by separate tasks; and the seeking process involves not only inter-personal communication, but also the use of formal information systems.
User assumptions	User assumption is default information inherited from the stereotypes.
User Centered Design (UCD)	The goal of the User Centered Design (UCD) technique is to allow people to talk about what they think or have experienced, and enables them to speak about their interpretations of others' actions.
User Engineering (UE)	IBM has expanded the focus of UCD, which is user-driven, to create User Engineering (UE), which is business value-driven. The purpose of UE is to transform the design process in order to meet business and market requirements as well as user requirements. UE is critical in the implementation of two aspects of IBM's e-business on demand strategy: autonomic computing and integration.
User model based filtering and customization of web pages	The architecture involves filtering information objects saved at a publication centre and only forwarding information about interesting objects to the user. Each object is accompanied by meta-data that contains information about the object (such as ratings, keywords, etc.). The filtering is carried out using information about the user from a user model and by examining the meta-data or actual data of the objects (Kay and Kummerfeld 1996).
User Modeling	The meaning of User Modeling in this research refers to the model built up to describe the user experience with a certain product, mainly software system. User modeling usually includes three possible kinds of facts: user observations, assumptions, and derived data.
User observations	User observations are certain information observed about the user or facts present in all the possible models.
User research	Covers a wide range of potential areas of study, from the question of users' choices of hamburgers in a fast food restaurant, to reactions to on-line surveys, to the in-depth analysis of the user needs that result in information seeking. Kuniavsky (2003) defined user research as the process of understanding the impact of design on an audience. Furthermore, he stated that surveys, focus groups, and other forms of user research conducted before the design phase could make the difference between a Web site (or any designed product) that is useful, usable, and successful; and one that is an unprofitable exercise in frustration for everyone involved.
WAN	Wide Area Network

WebObjects	The three-tier WebObjects architecture cleanly separates data access, business logic and user interface, making it easy to develop flexible, scalable applications.
Workflow	In 1996, the Workflow Management Coalition defined workflow as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (Allen, 2004).
XMI (XML Metadata Interchange, another OMG standard)	By using XMI (XML Metadata Interchange, another OMG standard), people can transfer the UML model from one tool into a repository, or into another tool for refinement or the next step in development process (OMG, 2003).

APPENDIX A
TABLE OF COMMERCIAL MULTIDIMENSIONAL STRUCTURING TECHNIQUES

According to Pendse and Greeth (1995), the commercial multidimensional structuring techniques can be organized into four groups: array, fixed record table, fixed record table in an RDBM, and variable length record structure. Table A-1 (Pendse and Greeth, 1995) shows the details of these groups.

Table A-1 Table of commercial multidimensional structuring techniques

Basic Structure	Compression or Indexing Technique	Speed Efficiency	Space Efficiency	Comments
Array	None	The fastest possible method for read and write, regardless of the number of dimensions.	Suitable for small or dense structures (>30 percent density). No overhead for keys or indexes.	Well Suited for complex modeling calculations with modest quantities of data. Often used for data held in RAM.
Array	Empty pages suppressed	Very fast for read, good for write.	Suitable for "clumpy" data on disk. Good for density > 10 percent.	A typical mainframe database technique where large pages of data could be moved rapidly between disk and memory.
Array	Strings of repeated values compressed	Slow for read, very slow for write. Unsuitable for random access.	Good for planning data (with many repeated or empty values), less good for actuals.	A good way of handling disk-based dynamic compression with black transfers and unpacking in memory.
Fixed record table	Hashing	Good for read and write with moderate data quantities.	Efficient dynamic sparsity in all but one dimension.	Good dynamic sparsity handling in all but the columns dimension.
Fixed record table	B-tree	Good for read and write.	Good for moderately sparse data without too many dimensions.	Good for moderate data quantities.

Table A-1. Continued

Basic Structure	Compression or Indexing Technique	Speed Efficiency	Space Efficiency	Comments
Fixed record table	Bitmap	Good for read and write	Good for moderately sparse data with more dimensions than B-tree	Good for moderate to large data quantities.
Fixed record table	Sorted	Good for read, very bad for write. Unaffected by sparsity.	Best with small numbers of dimensions.	Good for large, very sparse, read-only data sets.
Fixed record table in an RDBMS	As used in the RDBMS	Acceptable for occasional read and write. Can be slow for bulk storage.	Effectively no net space used if the data was stored in the RDBMS anyway.	Good for integrating with other systems and warehouses, and for using the RDBMS's data management facilities.
Variable length record structure	Sorted and compressed records	Acceptable read, very slow write.	Excellent data compression of randomly sparse data on disk.	An excellent way of storing, but not processing, very sparse data on disk.
Variable length record structure	Sorted records, with only incremental differences stored	Slow to read, completely unsuitable for write; updates must be stored separately and merged occasionally.	Highly flexible storage of tagged individual cells. Efficient for applications with up to a few million cells.	A specialized technique for storing mixed data types multi-dimensionally.

APPENDIX B
EXPLANATION OF THE RETE ALGORITHM (REINHARD 2003)

In forward chaining logical reasoning, a set of rules is checked against a fact base, which is represented as a set of Working Memory Elements (WMEs). If a rule identifies a WME that matches the criteria described in a rule, that rule is executed. The execution of the rule can trigger actions that add new facts (WMEs) to the fact base, delete facts from it, or trigger other actions. If a rule triggers the addition of new facts (WMEs) to the fact base, this matching operation has to be repeated. The cycle of matching the set of rules against the fact base is repeated until a rule is executed that interrupts this cycle or if the fact base does not change any more.

The purpose of the Rete Algorithm is to identify WMEs that match the rules modeled in the Rete networks. This matching process requires a population of the Rete network(s) with representations of WMEs, which are called tokens. Each token has a pointer to the WME that it represents and holds the information about whether the WME should be added or deleted from the two Rete networks.

The rules that can be applied in the Rete Algorithm match the requirements that the query imposes in terms of expressiveness and scope. The nested structure of predicate descriptions (attribute-relationship-value) in a query can be represented in the rules as well. The ability of the Rete Algorithm to retrieve WMEs based on inter-element features remains unused. One rule is represented in the created pattern network as one path in the tree representation of the pattern network. If a new rule is added to the rule base, a new

path is created in the pattern network. This new path can then be reused with as many nodes, which represent predicates, as it has in common with other existing rules.

APPENDIX C
ASSESSING USER INVOLVEMENT IN THE USER INTERFACE DESIGN

C.1 Sample Java Code for QueryParser

```
import java.util.*;
import java.sql.*;
import javax.swing.*;

public class QueryParser {
   private static String delimiter="\\bof\\b|\\bat\\b|\\bon\\b|\\band\\b";
   private Statement stmt;
   //private String delimiter2=" |,";
   /** Creates a new instance of QueryParser */
   public QueryParser() {
      try{
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String url = "jdbc:odbc:construction";
      Connection con = DriverManager.getConnection(url,"myLogin", "myPassword");
      stmt = con.createStatement();
      }
      catch (Exception e)  {System.err.println(e); }
   }
   public static String[] TokenStr(String query)
   {   … …
      while (st.hasMoreTokens()){
         res[i]=st.nextToken();
         i++;
      }
      return res;
   }
   public static String IsHit(String[] tokens, Statement stmt)
   {
      try{
      for (int i=0; i<tokens.length; i++)
      {
         … …
         ResultSet rs=stmt.executeQuery("select * from tbl_Terms where Term =
'"+term+"'");
         … …
         if (found) return term;
         else return "";
```

```
        }         }
        catch (Exception e)  { }
              return "";
    }


public ArrayList parse(String query)
{
        query=query.toLowerCase();
        String[] terms= query.split(delimiter);
        ArrayList result=new ArrayList();
        for (int i=0; i<terms.length; i++)
        {
           String[] tokens=QueryParser.TokenStr(terms[i]);
           nextterm:
           for (int beginindex=0; beginindex<tokens.length; beginindex++)
           {
              for (int len=tokens.length-beginindex; len>0; len--)
              {
                 String term="";
                 for (int p=0; p<len; p++) term+=" "+tokens[beginindex+p];
                 term=term.trim();
                 String key="";
                 if ((key=checkTerm(term,"tbl_CSITerms")).equals(""))
                 {
                    if((key=checkTerm(term,"tbl_Terms")).equals(""))
                    {
                       if (term.indexOf(" ")==-1)
                       {
                          if (0==JOptionPane.showConfirmDialog(null,"The word "'+term+
                          "' was not found in the dictionary, will you keep it as key in the
query?","Confirm",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSA
GE))
                          {
                             key=term;
                             result.add(key);
                          }
                       }
                    }
                    else{
                       if (key.endsWith("*")) key=key.substring(0,key.length()-1);
                       result.add(key);
                       beginindex+=len-1;
                       continue nextterm;
                    }
                 }
```

```
                else{
                    if (key.endsWith("*")) //keep term not in CSI as key
                    {
                        key=key.substring(0,key.length()-1);
                        result.add(key);
                        beginindex+=len-1;
                        continue nextterm;
                    }
                    else{    //expend the CSI term
                        ArrayList sarr=getCSISuggetions(key);
                        if (sarr.size()>0){
                            key=CSISuggestion.showCSISuggestion(null, true, key, sarr);
                            result.add(key);
                            beginindex+=len-1;
                            continue nextterm;
                        }
                        else {
                            result.add(key);
                            beginindex+=len-1;
                            continue nextterm;
                        } }  } } } }
        return result;
    }
    private ArrayList getCSISuggetions(String term)
    {
        … …
        ResultSet rs=stmt.executeQuery("select * from tbl_CSITerms where Term =
"'+term+'"");
        … …
    }
    private String checkTerm(String term, String table)
    {
        //term=term.toLowerCase();
        try{
        ResultSet rs=stmt.executeQuery("select * from "+table+" where Term =
"'+term+'"");
        boolean res=rs.next();
        if (res)
        { return term; }
        else
        { // use Edit Length to match the word. For details please see section 8.5.3 of this
dissertation
            int errorallowence=Math.round(term.length()*0.2f);
            rs=stmt.executeQuery("select * from "+table+" where Length>="+(term.length()-
errorallowence)+
            " and Length<="+(term.length()+errorallowence));
```

```
            ArrayList termlist=new ArrayList();
              … …
        }  }
      … …
    }
}
```

C.2 Sample Java Code Navigating a Construction Project Database

```
public class SearchEngine {
  private ArrayList keyarr;
  private String SystemPath="c:\\construction";
  public SearchEngine(ArrayList keys) {
     keyarr=new ArrayList();
     for (int i=0; i<keys.size();i++) { keyarr.add(keys.get(i).toString().toLowerCase());}
   }
   public ArrayList search(int st, String sp) { return search(st,new File(sp)); }
   public ArrayList search(int searchType, File sf)
   {
      ArrayList result=new ArrayList();
      if (sf.isDirectory())
      {
         //Continue Searching
         … …
      }
      else
      {
         String fname=sf.getName();
         String tname=fname.substring(0,fname.lastIndexOf("."));
         String fpath=sf.getPath();
         String tpath=fpath.substring(0,fpath.lastIndexOf("."));
         String pf=fpath.substring(fpath.lastIndexOf(".")+1);
         if (searchType==1)  //search filenames
         {
            if (pf.equalsIgnoreCase("doc") || pf.equalsIgnoreCase("pdf") ||
pf.equalsIgnoreCase("xls") || pf.equalsIgnoreCase("dwf"))
              {
                 int priority=checkKey(tname);
                 if (priority>0)
                 {
                    if (priority>9) priority=9;
                    result.add(priority+"-"+fpath);
                 }
              }
              else if (pf.equalsIgnoreCase("txt") || pf.equalsIgnoreCase("key")){}
              else {System.err.println("unrecognized file type:"+pf); return result;}
         }
         else if(searchType==2)  //search whole content
```

```
        {
          if (pf.equalsIgnoreCase("doc") || pf.equalsIgnoreCase("pdf") ||
pf.equalsIgnoreCase("xls")
          || pf.equalsIgnoreCase("dwf"))
          {
            File f=new File(tpath+".txt");
          if (f.exists())
            {
              String buf="";
              try{
                BufferedReader br=new BufferedReader(new FileReader(f));
                String line;
                while ((line=br.readLine())!=null)
                {
                  if (line.trim().length()>0) buf+=line;
                }
              } catch (Exception e) { System.err.println(e); }
              … …
            }
          }
          else if (pf.equalsIgnoreCase("txt") || pf.equalsIgnoreCase("key")){}
          else {
            System.err.println("unrecognized file type:"+pf);
            return result;
          }
        }
        else if (searchType==3)  //search Keys
        {
          if (pf.equalsIgnoreCase("doc") || pf.equalsIgnoreCase("pdf") ||
pf.equalsIgnoreCase("xls")
          || pf.equalsIgnoreCase("dwf"))
          {
            File f=new File(tpath+".key");
            if (f.exists())
            {
              //read the string
              … …
            }
          }
          else if (pf.equalsIgnoreCase("txt") || pf.equalsIgnoreCase("key")){}
          else {
            System.err.println("unrecognized file type:"+pf);
            return result;
          }
        }
      }
```

```
        return result;
    }
    private int  checkKey(String content)
    {
          //check and match the metadata
          … …
     }
}
```

APPENDIX D
TASK PROFILE AND USER PROFILE DATABASES

The following shows the task profiles and user profiles for the UMDA database. The XML format is implemented to manage the data. In the following examples, the task profiles and user profiles are separated.

D.1 Task Profile

D.1.1 Change Order

```
<?xml version="1.0" encoding="utf-8" ?>
<Change_Order  Response_Time="Immediately" Priority="Important"
Level="Detail" Primary_Actor="Project_Manager/Owner/Architect">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
         The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
         Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
      <!-- TaskID helps to combine user profile with task profile-->
      <!-- The symbol "," in between two words means their have OR relationship-->
                <Request TaskID="1" Location="\change_order"
SearchType="3">request_for_change_order</Request>
           <Request TaskID="2" Location="\memo"
SearchType="3">field_question</Request>
           <Request TaskID="3" SearchType="3">submittal_report_close-out</Request>
           <Request TaskID="4" SearchType="2">drawings</Request>
           <Request TaskID="5" SearchType="2">specifications</Request>
           <Request TaskID="6" SearchType="2">RFI</Request>
           <Request TaskID="7" Location="\change_order"
SearchType="3">subcontractor_quote</Request>
           <Request TaskID="8" Location="\change_order"
SearchType="3">quantity_take_off</Request>
           <Request TaskID="9" Location="\change_order"
SearchType="3">change_price_recap_estimate</Request>
           <Request TaskID="10" SearchType="3">memo</Request>
           <Request TaskID="11" SearchType="2">schedule</Request>
           <Request TaskID="12" SearchType="3">budget</Request>
```

```
        <Request TaskID="13" Location="\contract"
SearchType="2">general_contract</Request>
        <Request TaskID="14" SearchType="3">documents</Request>
        <Request TaskID="15" Location="\change_order"
SearchType="3">formal_change_order</Request>
            <Extensions TaskID="16" Location="\change_order"
SearchType="1">change_confirmation</Extensions>
        <Extensions TaskID="17" Location="\change_order"
SearchType="3">claim</Extensions>
      <Goal_in_Context>
            Primary actors discuss changes in the project based on the information
provided by system.
      </Goal_in_Context>
    <Precondition>
       after the contract has been awarded.
    </Precondition>
      <Scope>
            Seen by the people in the company, and Architect, Engineer, Owner, and
Subcontractors.
    </Scope>
      <Guarantee>
            Project Manager checks Request for Change Order proposal before
sending it to Architect or Owner.
      </Guarantee>
</Change_Order>
```

D.1.2 Close Out Administration

```
<?xml version="1.0" encoding="utf-8" ?>
<Close_Out_Administration  Response_Time="Varies" Priority="various"
Level="Detail" Primary_Actor="Project_Manager">
    <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
    <!-- Project Name is the root node
       The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
      Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
    <!-- The default value of variable Location is from root node-->
            <Request TaskID="1" Location="\submittal_report_close-out"
SearchType="1">close-out</Request>
        <!-- the close-out folder has manufactural warranty, subcontractor warranty,
          user manual, and as-built drawings-->
        <Request TaskID="2" SearchType="3">documents</Request>
        <!-- the document folder has transmittals-->
        <Request TaskID="3" SearchType="2"
SearchFileName="subcontractor_name_list">subcontractors</Request>
        <Request TaskID="4" Location="\submittal_report_close-out"
SearchType="1">report</Request>
```

```
            <!-- the report folder has test reports, inspection reports, certificates,
              and punch_out_list-->
                  <Extensions TaskID="5" SearchType="3">budget</Extensions>
            <!-- the budget folder has the list for the subcontractors' retainage-->

         <Goal_in_Context>
            Close-out project and transmit the close-out documents to Architect/Owner
         </Goal_in_Context>
      <Precondition>
            project construction finishes 90%.
      </Precondition>
         <Scope>
               Project Manager and Superintendent need to see the documents
      </Scope>
         <Guarantee>
               make sure all the warranties are provided to Owner.
         </Guarantee>
</Close_Out_Administration>
```

D.1.3 Competitive Bidding

```
<?xml version="1.0" encoding="utf-8" ?>
<Competitive_Bidding  Response_Time="Before deadline" Priority="Important"
Level="Detail" Primary_Actor="Project_Manager/President">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
               <Request TaskID="1" SearchType="2">budget</Request>
            <Request TaskID="2" SearchType="2">drawings</Request>
            <Request TaskID="3" SearchType="2">specifications</Request>
            <Request TaskID="4" SearchType="2">schedule</Request>
            <Request TaskID="5" Location="\budget"
SearchType="2">bid_proposal</Request>
            <Request TaskID="6" SearchType="3">documents</Request>
             <!-- the document folder has transmittals-->
         <Goal_in_Context>
               submit bid proposal based on the information provided by system
         </Goal_in_Context>
      <Precondition>
          have drawings and specifications.
      </Precondition>
         <Scope>
               Seen by Project Manager and President.
      </Scope>
```

<Guarantee>
estimated budget covers all the items under contractor's responsibility and the bidding proposal is submitted before deadline.
</Guarantee>
</Competitive_Bidding>

D.1.4 Establish Cost Model and Develop Cost Estimates

<?xml version="1.0" encoding="utf-8" ?>
<Establish_Cost_Model_and_Develop_Cost_Estimates Response_Time="various" Priority="various"
Level="Detail" Primary_Actor="Project_Manager">
<!--1:search filename; 2:search whole content; 3:match keywords metadata -->
<!-- Project Name is the root node
The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI, Schedule,
Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
<!-- The default value of variable Location is from root node-->
<Request TaskID="1" Location="\budget" SearchType="3">draw_request</Request>
<!-- The draw_request folder has subcontractor draw request files
and general contractor draw request files.-->
<Request TaskID="2" SearchType="2">budget</Request>
<Request TaskID="3" SearchType="3">documents</Request>
<!-- the document folder has transmittals-->
<Goal_in_Context>
control the construction cost within budget.
</Goal_in_Context>
<Precondition>
Contractor is awarded the bid
</Precondition>
<Scope>
seen by stuff inside the company, Architect and Owner
</Scope>
<Guarantee>
Project Manager reviews every draw request to Owner.
</Guarantee>
</Establish_Cost_Model_and_Develop_Cost_Estimates>

D.1.5 Identify Trade Contracts

<?xml version="1.0" encoding="utf-8" ?>
<Identify_Trade_Contracts Response_Time="Important" Priority="various"
Level="Detail" Primary_Actor="Project_Manager">
<!--1:search filename; 2:search whole content; 3:match keywords metadata -->
<!-- Project Name is the root node

```
        The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
        Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
    <!-- The default value of variable Location is from root node-->
            <Request TaskID="1" SearchType="2">budget</Request>
        <Request TaskID="2" SearchType="2">drawings</Request>
        <Request TaskID="3" SearchType="2">specifications</Request>
        <Request TaskID="4" Location="\budget"
SearchType="2">bid_proposal</Request>
        <Request TaskID="5" Location="\memo"
SearchType="1">phone_record</Request>
        <Request TaskID="6" SearchType="2">contract</Request>
      <Goal_in_Context>
            Project Manager preparing contracts based on the information provided by
system.
      </Goal_in_Context>
    <Precondition>
        Has drawings and subcontractors' proposals.
    </Precondition>
      <Scope>
            Seen by the people in the company
    </Scope>
      <Guarantee>
            Project Manager checks the quantities, scope of work, subcontractor's
construction cost, etc.
        in the contractor's proposal, negotiates with subcontractor, and prepares the
subcontract for that trade.
      </Guarantee>
</Identify_Trade_Contracts>
```

D.1.6 Maintain Project Site Documents

```
<?xml version="1.0" encoding="utf-8" ?>
<Maintain_Project_Site_Documents  Response_Time="Important" Priority="Important"
Level="Detail" Primary_Actor="Superintendent">
    <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
    <!-- Project Name is the root node
        The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
        Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
    <!-- The default value of variable Location is from root node-->
        <Request TaskID="1" SearchType="2">drawings</Request>
        <Request TaskID="2" SearchType="2">specifications</Request>
        <Request TaskID="3" Location="contract"
SearchType="2">general_contract</Request>
        <Request TaskID="4" SearchType="2">contract</Request>
```

```
            <Request TaskID="5" Location="\submittal_report_close-out"
SearchType="3">submittal</Request>
            <!-- the submittal folder has submittals and shop_drawings-->
            <Request TaskID="6" SearchType="3">change_order</Request>
            <Request TaskID="7" Location="\memo"
SearchType="3">field_report</Request>
            <Request TaskID="8" Location="\submittal_report_close-out\close-out"
SearchType="1">as_built</Request>
            <!-- the close-out folder has manufactural warranty, subcontractor warranty,
              user manual, and as-built drawings-->
         <Goal_in_Context>
                Maintain Project Site Documents
         </Goal_in_Context>
      <Precondition>
           project construction has started.
      </Precondition>
         <Scope>
                Project Manager and Superintendent need to see the documents
      </Scope>
         <Guarantee>
                make sure all the site changes are documented.
         </Guarantee>
</Maintain_Project_Site_Documents>
```

D.1.7 Perform Constructability Review

```
<?xml version="1.0" encoding="utf-8" ?>
<Perform_Constructability_Review  Response_Time="Varies" Priority="various"
Level="Detail" Primary_Actor="Project_Manager">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
         The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
         Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
                <Request TaskID="1" SearchType="2">budget</Request>
            <Request TaskID="2" SearchType="2">drawings</Request>
            <Request TaskID="3" SearchType="2">specifications</Request>
            <Request TaskID="4" SearchType="2">schedule</Request>
            <Request TaskID="5" Location="\memo"
SearchType="1">feasibility_report</Request>
         <Goal_in_Context>
                study project feasibility for owner.
         </Goal_in_Context>
      <Precondition>
           None
      </Precondition>
```

```
        <Scope>
                seen by Project Manager, President, and Owner
      </Scope>
        <Guarantee>
                Project Manager reviews the market price and technical feasibility.
        </Guarantee>
</Perform_Constructability_Review>
```

D.1.8 Perform Quality Control Inspection

```
<?xml version="1.0" encoding="utf-8" ?>
<Perform_Quality_Control_Inspection  Response_Time="Important"
Priority="Important"
Level="General" Primary_Actor="Project_Manager/Superintendent">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
            <Request TaskID="1" SearchType="2">drawings</Request>
            <Request TaskID="2" SearchType="2">specifications</Request>
            <Request TaskID="3" Location="\submittal_report_close-out\close-out"
SearchType="2">as_built</Request>
            <Request TaskID="4" Location="\change_order"
SearchType="3">formal_change_order</Request>
            <Request TaskID="5" Location="\memo"
SearchType="3">field_report</Request>
            <Request TaskID="6" Location="\memo"
SearchType="1">inspection</Request>
             <!-- the inspection folder has inspection reports and inspection log files-->
        <Goal_in_Context>
                keep track of the Quality Control/Inspection in the construction process.
        </Goal_in_Context>
      <Precondition>
          project construction has started
      </Precondition>
        <Scope>
                Seen by people inside the company
      </Scope>
        <Guarantee>
                Project quality needs to be guaranteed.
        </Guarantee>
</Perform_Quality_Control_Inspection>
```

D.1.9 Perform Value Engineering

```
<?xml version="1.0" encoding="utf-8" ?>
<Perform_Value_Engineering  Response_Time="Varies" Priority="various"
Level="Detail" Primary_Actor="Project_Manager/Superintendent">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
         The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
         Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
                <Request TaskID="1" SearchType="2">drawings</Request>
           <Request TaskID="2" SearchType="2">specifications</Request>
           <Request TaskID="3" Location="\memo"
SearchType="3">field_report</Request>
           <!-- The field_report folder has the daily field report done by superintendent
             and the field visit report done by project manager-->
           <Request TaskID="4" Location="\memo"
SearchType="3">correspondence</Request>
           <!-- The correspondence folder has expert suggestion files-->
           <Request TaskID="5" SearchType="2">budget</Request>
           <Request TaskID="6" SearchType="2">schedule</Request>
           <Request TaskID="7" Location="\memo" SearchType="3">memo</Request>
           <!-- The memo folder under memo has value_engineering_proposal
             and architect_approval files-->
           <Request TaskID="8" Location="\memo"
SearchType="3">meeting_minutes</Request>
           <Request TaskID="9" Location="\change_order"
SearchType="3">request_for_change_order</Request>
        <Goal_in_Context>
                propose value-engineering options for owner to choose, based on the
information provided by system.
        </Goal_in_Context>
      <Precondition>
          None
      </Precondition>
        <Scope>
                seen by the people in the company and Owner, Architect, and Engineer.
Final value-engineering choice will be sent to the corresponding subcontractors and/or
vendors
      </Scope>
        <Guarantee>
                Architect approves the structural integrity and functionality.
        </Guarantee>
</Perform_Value_Engineering>
```

D.1.10 Pre-Qualification

```
<?xml version="1.0" encoding="utf-8" ?>
<Pre-Qualification  Response_Time="Varies" Priority="Mediate"
Level="General" Primary_Actor="Project_Manager/Owner/Architect">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
                  <Request TaskID="1" SearchType="2">drawings</Request>
            <Request TaskID="2" SearchType="2">specifications</Request>
            <Request TaskID="3" SearchType="2">subcontractors</Request>
             <!-- The subcontractors folder has subcontractor_insurance,
              subcontractor_license, and pre_qualified_subcontractor_list-->
         <Goal_in_Context>
                  Select out the qualified subcontractors and vendors based on the
information provided by system.
         </Goal_in_Context>
      <Precondition>
           have drawings and specifications.
      </Precondition>
         <Scope>
                  Seen by the people in the company
      </Scope>
         <Guarantee>
                  all trades included in the project have subcontractors or vendors to do
them.
         </Guarantee>
</Pre-Qualification>
```

D.1.11 Preparing Budget

```
<?xml version="1.0" encoding="utf-8" ?>
<Preparing_Budget  Response_Time="Meet the bidding date deadline"
Priority="Important"
Level="Detail" Primary_Actor="Project_Manager">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
                  <Request TaskID="1" SearchType="2">drawings</Request>
            <Request TaskID="2" SearchType="2">specifications</Request>
```

```
            <Request TaskID="3" Location="\memo"
SearchType="3">field_report</Request>
            <!-- The field_report folder has the daily field report done by superintendent
              and the field visit report done by project manager-->
            <Request TaskID="4" SearchType="2">budget</Request>
            <!-- The budget folder has quantity_take_off, cost_summary,
            subcontractor_proposal, vendor_proposal, and general contractor's
bid_proposal files-->
        <Goal_in_Context>
                Project Manager preparing budget for a project, based on the information
provided by system.
        </Goal_in_Context>
      <Precondition>
          Has drawings.
      </Precondition>
        <Scope>
                The material take-off and estimating are seen by the people in the
company.
      </Scope>
        <Guarantee>
                Project Manager approves that every item in the project that is under
contractor's responsibility has been covered in the estimated budget.
        </Guarantee>
</Preparing_Budget>
```

D.1.12 Project Scheduling

```
<?xml version="1.0" encoding="utf-8" ?>
<Project_scheduling  Response_Time="Varies" Priority="Various"
Level="Detail" Primary_Actor="Project_Manager/Superintendent">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
                <Request TaskID="1" SearchType="2">drawings</Request>
            <Request TaskID="2" SearchType="2">specifications</Request>
            <Request TaskID="3" Location="\budget"
SearchType="3">productivity</Request>
            <Request TaskID="4" Location="\contract"
SearchType="2">general_contract</Request>
            <Request TaskID="5" SearchType="2">schedule</Request>
            <!-- the schedule folder has project_schedule and look_ahead_schedule-->
        <Goal_in_Context>
                Project Manager developing project schedule and work with
Superintendent to maintain it.
```

```
        </Goal_in_Context>
    <Precondition>
        Drawings are ready
    </Precondition>
        <Scope>
            The final approved master schedule is seen by the people in the company
and architect, owner, subcontractors, and vendors.
    </Scope>
        <Guarantee>
            Project Manager makes sure that the final completion date is met.
        </Guarantee>
</Project_scheduling>
```

D.1.13 Provide Construction Reports

```
<?xml version="1.0" encoding="utf-8" ?>
<Provide_Construction_Reports  Response_Time="Same day" Priority="Varies"
Level="Detail" Primary_Actor="Superintendent">
    <Request TaskID="1" SearchType="2">drawings</Request>
    <Request TaskID="2" SearchType="2">specifications</Request>
    <Request TaskID="3" Location="\memo" SearchType="3">field_report</Request>
    <!-- The field_report folder has the daily field report done by superintendent
     and the field visit report done by project manager-->
        <Goal_in_Context>
            Generate Field Report Process
        </Goal_in_Context>
    <Precondition>
        project construction has started.
    </Precondition>
        <Scope>
            Project Manager and Superintendent need to see the documents
    </Scope>
        <Guarantee>
            make sure all the site conditions are documented.
        </Guarantee>
</Provide_Construction_Reports>
```

D.1.14 Provide Project Staffing

```
<?xml version="1.0" encoding="utf-8" ?>
<Provide_Project_Staffing  Response_Time="Varies" Priority="Varies"
Level="General" Primary_Actor="Project Manager/Architect/Owner">
    <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
    <!-- Project Name is the root node
        The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
        Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
```

```
<!-- The default value of variable Location is from root node-->
<Request TaskID="1" SearchType="2">subcontractors</Request>
  <Goal_in_Context>
        keep track of the activities of project staff in the construction process.
  </Goal_in_Context>
<Precondition>
   None
</Precondition>
  <Scope>
        Seen by all construction participants
</Scope>
  <Guarantee>
        all the project staff has their names on the list.
  </Guarantee>
</Provide_Project_Staffing>
```

D.1.15 Safety

```
<?xml version="1.0" encoding="utf-8" ?>
<safety Response_Time="Varies" Priority="Important"
Precondition="project construction has started"
Level="Detail" Primary_Actor="Project Manager/Superintendent">
<!--1:search filename; 2:search whole content; 3:match keywords metadata -->
<!-- Project Name is the root node
   The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
   Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
<!-- The default value of variable Location is from root node-->
  <Request TaskID="1" SearchType="2">memo</Request>
<!-- The memo folder has variaty of reports, including safety_meeting_minutes
 safety_reports, field_report, police_report, first_aid, safety_sign,
 safety_inspection, and field_report-->
  <Extension TaskID="2" SearchType="2">documents</Extension>
  <Goal_in_Context>
        Project Manager reviews shop drawings and submit to Architect/Engineer
for approval.
  </Goal_in_Context>
  <Scope>
        Provide safe construction environment
  </Scope>
  <Guarantee>
        make sure of job safety.
  </Guarantee>
</safety>
```

D.1.16 Shop Drawing Review

```
<?xml version="1.0" encoding="utf-8" ?>
<shop_drawing_review Response_Time="Important" Priority="Important"
Precondition="project construction has started" Level="Detail"
Primary_Actor="Project Manager">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
         The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
         Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
        <Request TaskID="1" SearchType="2">drawings</Request>
      <Request TaskID="2" SearchType="2">specifications</Request>
      <Request TaskID="3" Location="\submittal_report_close-out"
SearchType="1">submittal</Request>
       <!-- the submittal folder has shop_drawings and submittal documents-->
      <Extension TaskID="4" SearchType="1">change_order</Extension>
      <FunctionTrigger>Change_Order</FunctionTrigger>
        <Goal_in_Context>
              Project Manager reviews shop drawings and submit to Architect/Engineer
for approval.
        </Goal_in_Context>
        <Scope>
              Project Manager, Superintendent, Architect, Engineer, Subcontractors
need to see the shop drawings.
        </Scope>
        <Guarantee>
              Project quality needs to be guaranteed.
        </Guarantee>
</shop_drawing_review>
```

D.1.17 Submittal Process

```
<?xml version="1.0" encoding="utf-8" ?>
<submittal_process Response_Time="Before the specific work starts"
Priority="Important" Precondition="drawings and specifications"
Level="General" Primary_Actor="Project Manager/Architect/Owner">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
         The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
         Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
        <Request TaskID="1" SearchType="2">drawings</Request>
      <Request TaskID="2" SearchType="2">specifications</Request>
        <Request TaskID="3" SearchType="2">schedule</Request>
```

```
        <Request TaskID="4" Location="\submittal_report_close-out"
SearchType="1">submittal</Request>
      <!-- the submittal folder has shop_drawings and submittal documents-->
        <Request TaskID="5" SearchType="3">documents</Request>
      <!-- the documents folder has submittal_transmittals-->
        <Extension TaskID="6" SearchType="1">change_order</Extension>
        <Goal_in_Context>
              Project Manager submits the required items for Architect/Owner review
and approval and keep track of the process.
        </Goal_in_Context>
        <Scope>
              Seen by Project Manager and Superintendent.
        </Scope>
        <Guarantee>
              Architect/Engineer approves the work before it starts.
        </Guarantee>
</submittal_process>
```

D.1.18 Material Management

```
<?xml version="1.0" encoding="utf-8" ?>
<material_management Response_Time="Instant" Priority="Various"
Level="Summary" Primary_Actor="Project_Manager">
      <!--1:search filename; 2:search whole content; 3:match keywords metadata -->
      <!-- Project Name is the root node
          The root node has 11 children: Document, ChangeOrder, Budget, Memo, RFI,
Schedule,
          Contract, Submittal, Drawings, submittal_report_close-out, and Specifications-->
      <!-- The default value of variable Location is from root node-->
        <Request TaskID="1" SearchType="2">drawings</Request>
      <Request TaskID="2" SearchType="2">specifications</Request>
        <Request TaskID="3" SearchType="2">budget</Request>
        <Request TaskID="4" SearchType="2">schedule</Request>
        <Request TaskID="5" Location="\memo" SearchType="1">storage</Request>
      <Request TaskID="6" SearchType="2">contract</Request>
      <Request TaskID="7" Location="\budget"
SearchType="1">quantity_take_off</Request>
      <Request TaskID="8" Location="\budget"
SearchType="1">material_receipt</Request>
        <Goal_in_Context>
              Project Manager buys material through the system, gets it. Does not
include paying for it.
        </Goal_in_Context>
      <Precondition>
            None
      </Precondition>
        <Scope>
```

Business – The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
</Scope>
<Guarantee>
Every order sent out has been approved by the Project Manager.
</Guarantee>
</material_management>

D.2 User Profile

D.2.1 Architect

```
<?xml version="1.0" encoding="utf-8" ?>
<Architect>
    <!-- User ID, security check and other general user information are included in
    a user table in the UMDA project database. -->
    <!-- The symbol "," in between two words means their have OR relationship-->
            <View_Order Order="1">drawings</View_Order>
        <View_Order Order="2">specifications</View_Order>
        <View_Order Order="3">schedule</View_Order>
        <View_Order Order="4">contract</View_Order>
        <View_Order Order="5">budget</View_Order>
        <View_Order Order="6">RFI</View_Order>
        <View_Order Order="7">change_order</View_Order>
        <View_Order Order="8">submittal_report_close-out</View_Order>
        <View_Order Order="9">memo</View_Order>
      <Granularity>
            <Gran_Level DataType="drawings, specifications, contract, budget,
RFI">high</Gran_Level>
        <Gran_Level DataType="change_order, submittal_report_close-
out">medium</Gran_Level>
        <Gran_Level DataType="schedule, memo">low</Gran_Level>
      </Granularity>
    <View_Type>orginal_format and text format</View_Type>
</Architect>
```

D.2.2 BD_Official (Building Department Official)

```
<?xml version="1.0" encoding="utf-8" ?>
<BD_Official>
    <!-- User ID, security check and other general user information are included in
    a user table in the UMDA project database. -->
    <!-- The symbol "," in between two words means their have OR relationship-->
            <View_Order Order="1">drawings</View_Order>
        <View_Order Order="2">specifications</View_Order>
        <View_Order Order="3">submittal_report_close-out</View_Order>
```

```
        <Granularity>
          <Gran_Level DataType="drawings">medium</Gran_Level>
          <Gran_Level DataType="specifications, submittal_report_close-
out">low</Gran_Level>
        </Granularity>
      <View_Type>orginal_format and text format</View_Type>
</BD_Official>
```

D.2.3 Engineer

```
<?xml version="1.0" encoding="utf-8" ?>
<Engineer>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
            <View_Order Order="1">drawings</View_Order>
          <View_Order Order="2">specifications</View_Order>
          <View_Order Order="3">RFI</View_Order>
          <View_Order Order="4">change_order</View_Order>
          <View_Order Order="5">submittal_report_close-out</View_Order>
        <Granularity>
            <Gran_Level DataType="drawings, specifications,
RFI">high</Gran_Level>
          <Gran_Level DataType="submittal_report_close-out">medium</Gran_Level>
          <Gran_Level DataType="change_order">low</Gran_Level>
        </Granularity>
      <View_Type>orginal_format and text format</View_Type>
</Engineer>
```

D.2.4 Estimator

```
<?xml version="1.0" encoding="utf-8" ?>
<Estimator>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
            <View_Order Order="1">drawings</View_Order>
          <View_Order Order="2">specifications</View_Order>
        <Granularity>
            <Gran_Level DataType="drawings, specifications">high</Gran_Level>
        </Granularity>
      <View_Type>orginal_format</View_Type>
</Estimator>
```

D.2.5 Office Manater

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<Office_Manager>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
            <View_Order Order="1">contract</View_Order>
            <View_Order Order="2">budget</View_Order>
            <View_Order Order="3">documents</View_Order>
            <View_Order Order="4">schedule</View_Order>
            <View_Order Order="5">change_order</View_Order>
         <Granularity>
                  <Gran_Level DataType="contract">high</Gran_Level>
            <Gran_Level DataType="budget">medium</Gran_Level>
            <Gran_Level DataType="documents, schedule,
change_order">low</Gran_Level>
         </Granularity>
      <View_Type>text format</View_Type>
</Office_Manager>
```

D.2.6 Owner

```
<?xml version="1.0" encoding="utf-8" ?>
<Owner>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
                  <View_Order Order="1">drawings</View_Order>
            <View_Order Order="2">budget</View_Order>
            <View_Order Order="3">contract</View_Order>
            <View_Order Order="4">schedule</View_Order>
            <View_Order Order="5">documents</View_Order>
            <View_Order Order="6">change_order</View_Order>
            <View_Order Order="7">submittal_report_close-out</View_Order>
            <View_Order Order="8">memo</View_Order>
         <Granularity>
                  <Gran_Level DataType="budget, contract">high</Gran_Level>
            <Gran_Level DataType="drawings, schedule,
change_order">medium</Gran_Level>
            <Gran_Level DataType="documents, submittal_report_close-out,
memo">low</Gran_Level>
         </Granularity>
      <View_Type>orginal_format and text format</View_Type>
</Owner>
```

D.2.7 President

```
<?xml version="1.0" encoding="utf-8" ?>
<President>
```

```
<!-- User ID, security check and other general user information are included in
a user table in the UMDA project database. -->
<!-- The symbol "," in between two words means their have OR relationship-->
        <View_Order Order="1">contract</View_Order>
        <View_Order Order="2">budget</View_Order>
        <View_Order Order="3">schedule</View_Order>
        <View_Order Order="4">change_order</View_Order>
        <View_Order Order="5">memo</View_Order>
    <Granularity>
            <Gran_Level DataType="contract">high</Gran_Level>
        <Gran_Level DataType="budget, schedule,
change_order">medium</Gran_Level>
        <Gran_Level DataType="memo">low</Gran_Level>
    </Granularity>
<View_Type>orginal_format and text format</View_Type>
</President>
```

D.2.8 Project Manager

```
<?xml version="1.0" encoding="utf-8" ?>
<Project_Manager>
<!-- User ID, security check and other general user information are included in
a user table in the UMDA project database. -->
<!-- The symbol "," in between two words means their have OR relationship-->
        <View_Order Order="1">drawings</View_Order>
        <View_Order Order="2">specifications</View_Order>
        <View_Order Order="3">schedule</View_Order>
        <View_Order Order="4">contract</View_Order>
        <View_Order Order="5">budget</View_Order>
        <View_Order Order="6">documents</View_Order>
        <View_Order Order="7">subcontractors</View_Order>
        <View_Order Order="8">RFI</View_Order>
        <View_Order Order="9">change_order</View_Order>
        <View_Order Order="10">submittal_report_close-out</View_Order>
        <View_Order Order="11">memo</View_Order>
    <Granularity>
            <Gran_Level DataType="drawings, specifications, contract, budget,
RFI">high</Gran_Level>
        <Gran_Level DataType="subcontractors, change_order,
submittal_report_close-out">medium</Gran_Level>
        <Gran_Level DataType="schedule, documents, memo">low</Gran_Level>
    </Granularity>
<View_Type>orginal_format and text format</View_Type>
</Project_Manager>
```

D.2.9 Subcontractor

```
<?xml version="1.0" encoding="utf-8" ?>
<Subcontractor>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
                  <View_Order Order="1">drawings</View_Order>
            <View_Order Order="2">specifications</View_Order>
            <View_Order Order="3">schedule</View_Order>
            <View_Order Order="4">RFI</View_Order>
            <View_Order Order="5">change_order</View_Order>
            <View_Order Order="6">submittal_report_close-out</View_Order>
         <Granularity>
                  <Gran_Level DataType="drawings, specifications">high</Gran_Level>
            <Gran_Level DataType="RFI">medium</Gran_Level>
            <Gran_Level DataType="schedule, change_order, submittal_report_close-
out">low</Gran_Level>
         </Granularity>
      <View_Type>orginal_format and text format</View_Type>
</Subcontractor>
```

D.2.10 Superintendent

```
<?xml version="1.0" encoding="utf-8" ?>
<Superintendent>
      <!-- User ID, security check and other general user information are included in
      a user table in the UMDA project database. -->
      <!-- The symbol "," in between two words means their have OR relationship-->
                  <View_Order Order="1">drawings</View_Order>
            <View_Order Order="2">specifications</View_Order>
            <View_Order Order="3">RFI</View_Order>
            <View_Order Order="4">change_order</View_Order>
            <View_Order Order="5">submittal_report_close-out</View_Order>
            <View_Order Order="6">schedule</View_Order>
            <View_Order Order="7">memo</View_Order>
            <View_Order Order="8">contract</View_Order>
            <View_Order Order="9">budget</View_Order>
            <View_Order Order="10">documents</View_Order>
            <View_Order Order="11">subcontractors</View_Order>
         <Granularity>
                  <Gran_Level DataType="drawings, specifications">high</Gran_Level>
            <Gran_Level DataType="contract, schedule, RFI, change_order,
submittal_report_close-out">medium</Gran_Level>
            <Gran_Level DataType="budget, documents, memo,
subcontractors">low</Gran_Level>
         </Granularity>
```

<View_Type>orginal_format and text format</View_Type>
</Superintendent>

APPENDIX E
INFORMATION RETRIEVAL PERFORMANCE

The evaluation of the effectiveness and the efficiency of the implemented UMDA system is important, as the information retrieval process is computationally expensive and is repeatedly executed when people need to use the system in a construction project. Besides the search functions, the UMDA system can also help users configure the user interface layer (i.e. the display functions) and configure the applications layer of the system, for example, the different operations provided to different users. The information retrieval process is expensive because it involves an input/output operation, a database connection, which in turn means the reading and writing of disk, matching algorithms to calculate the similarity of queries with the saved data, and rules to perform the matching operations. Besides the high number of matching operations, the computational cost of individual matches between the Extensible Task Element Tree (EXTET) and the properties of the elements from the navigating models, in many cases, has a significant on system performance.

After a user logs into the UMDA system, s/he propagates requests for information about a construction projects' products and processes. Then the system executes the Query Parser, Compare and Match Network, and Task Analysis Module. The system commands will be generated after the UMDA analysis matches the user profiles and task profiles of the query. The initial population of navigational models is executed after receiving the system commands. The system commands will trigger the provision of the navigational model with the search schema (loading an existing model or defining one)

and the use of the navigational model. Then the user can read the information from the system display and s/he can edit the project documents and files. The population of navigational models is a frequently used operation. In case the documents or files in the construction project database are changed, such as revised, added, or deleted, the navigational model schema is changed and the population process will be called to regenerate the schema. Different navigational models may be required in a project and navigational models may have to be added or adjusted.

The UMDA system needs to operate efficiently but it is uncertain how efficiently user requests can be answered. The computational cost of information retrieval requests on the product and process model of a project may be influenced by a lot of factors, including frequency of the user's misspellings, navigational model, and computer hardware. It is difficult to find general cases to relate the configurations of UMDA models to execution times of the information retrieval process. The validation of the efficiency of the UMDA system therefore establishes a set of metrics that allows for the determination of the overall time for the information retrieval process.

The first component, denoted **E**, of the metrics is for the performance of the Extensible Task Element Tree (EXTET) structure. This structure has two parts: the task profiles and the user profiles database and the task analysis module. The second component, denoted **C**, is the number of system commands for a navigational model. The metrics also include the number of comparisons (**Co**) and the specific time to extract features of an entity (**Ex**). A retrieval time (**Rt**) that is specific for a product and process model is established as follows:

$$Rt = E + C + Co + Ex \qquad \text{(E-1)}$$

The following performs an analysis of each of the factors in equation E-1.

- **Structure of the EXTET system.** The design objective for the EXTET system is to provide flexible and extensible structures to configure the system environment. The space requirement of the EXTET structure depends on the size of the user profile and task profile databases and a larger EXTET structure will slow down the performance of Task Analysis Module, which means component **E** and **C** in the formula for **Rt** is increased. For the performance tests, the author uses 5 different user groups and 17 task profiles as discussed in Chapter 8.

- **Size of the navigational model.** Navigational models can provide structures of all or part of ElementAmbassadors of the project for a particular task. Depending on the size of the navigational model and the ElementAmbassadors contained in it, the value of **C** in Equation E-1 will be changed.

- **Size of the construction project database.** The performance times for the information retrieval should be established for different project sizes. The test project size that is considered for this evaluation is a one-story commercial project, which has 1191 files in the database. The size of the construction project database will affect the value of **Co** and **Ex** in Equation E-1.

The objective of the validation of the efficiency of the information retrieval process is to demonstrate that the EXTET system has acceptable performance and helps users find the information they need within reasonable time. In order to assess the performance, the performance time of the prototype was measured as shown in Tables E-1 and E-2.

The abbreviations used in Table E-1 and Table E-2 are as follows:

- R – the number of parsed results
- SQL – the number of SQL queries. Here, SQL = R.
- Ma – the number of results retrieved for the requirement of the query
- Time – the performance time spending on information retrieval, in milliseconds.
- $\overline{PTM_{i\_j}}$ - the average performance time of each task module. PTM stands for Performance of Test Module. i is test module number. j is the index of whether or not a task profile is implemented.
- STDVi – the standard deviation of Test Module group i.

The following conclusions can be derived from the results in Table E-1:

- The UMDA system can improve the performance of the information retrieval operation.

- The numbers of parsed results, SQL queries, and the results matching the requirement of the query shown in this table comply with the analysis in Chapter 5.

- Larger construction project tends to have more documents and data than small ones. The size of the project database will affect the information retrieval performance. The larger the size is, the more time it needs to take to find the results.

- In the Test Module numbers, TM1 through TM11 have the comparisons between the performance results between using or not using User Profiles, Task Profiles, or Task Analysis Module. The average of $\overline{PTM_{i\_0}}$ is the performance time for information retrieval without using the User Profiles, Task Profiles, or Task Analysis Module. The number of $\overline{PTM_{i\_0}}$ is 4700.22 milliseconds.

- The average of $\overline{PTM_{i\_1}}$ is the performance time for information retrieval using the User Profiles, Task Profiles, or Task Analysis Module. The number of $\overline{PTM_{i\_1}}$ is 615.71 milliseconds. The significance in the difference between $\overline{PTM_{i\_1}}$ and $\overline{PTM_{i\_0}}$ means that the proposed UMDA system can improve the performance of information retrieval.

Table E-1 Design of the test modules for information retrieval in the EXTET system

Test Module Number	Project Name	User Group	Task Profile	Query	Explanation
TM1_0	1-story Commercial	PM	None	Column A-3 reinforcement change	1. Without task profile, the system needs to search through the entire database for requested data. 2. TM stands for Test Module. The first number following TM is the test number. If the second number is 0, the query does not have any task profile. If it is 1, it has a task profile. 3. 10 test modules are performed for PM user group. For each test module, the test is repeated 10 times. 4. The intention is to show that the UMDA system can help the information retrieval system to focus on needed documents and files for a query. 5. TM3_0&1 represent the situation for "window 2 delivery" in material management. TM4_0&1 show the RFI process. Task 5_0&1 are for the maintenance of punch list, as the case study discussed in Chapter 5.
TM1_1	1-story Commercial	PM	Change Order	Column A-3 reinforcement change	
TM2_0	1-story Commercial	PM	None	Office floor area	
TM2_1	1-story Commercial	PM	Preparing Budget	Office floor area	
TM3_0	1-story Commercial	PM	None	Window 2 delivery	
TM3_1	1-story Commercial	PM	Material Management	Window 2 delivery	
TM4_0	1-story Commercial	PM	None	RFI 3	
TM4_1	1-story Commercial	PM	Shop drawing review	RFI 3	
TM5_0	1-story Commercial	PM	None	Punch list electrical installation	
TM5_1	1-story Commercial	PM	Close out administration	Punch list electrical installation	
TM6_0	1-story Commercial	Superintendent	None	Window 2 delivery	1. Compare these results with TM3_0 and TM3_1. 2. Each test module is repeated 10 times
TM6_1	1-story Commercial	Superintendent	Material Management	Window 2 delivery	
TM7_0	1-story Commercial	Superintendent	None	RFI 3	1. Compare these results with TM4_0 and TM4_1.

Table E-1. Continued

TM7_1	1-story Commercial	Superintendent	Shop drawing review	RFI 3	2. Each test module is repeated 10 times.
TM8_0	1-story Commercial	Superintendent	None	Look ahead schedule roof	1. Each test module is repeated 10 times. 2. Prove that UMDA has better performance than system without user or task profiles. UMDA system also helps users find the things they need and save time.
TM8_1	1-story Commercial	Superintendent	Project scheduling	Look ahead schedule roof	
TM9_0	1-story Commercial	President	None	Window sound attenuation change	1. Compare these results with TM2_0 and TM2_1. 2. Each test module is repeated 10 times.
TM9_1	1-story Commercial	President	Perform Value Engineering	Window sound attenuation change	
TM10_0	1-story Commercial	Estimator	None	Office floor area	1. Compare these results with TM2_0 and TM2_1. 2. Each test module is repeated 10 times.
TM10_1	1-story Commercial	Estimator	Preparing Budget	Office floor area	
TM11_0	Multi-story Condo	PM	None	Project profit	1. Compare these results with TM2_0 and TM2_1. 2. Each test module is repeated 10 times.
TM11_1	Multi-story Condo	PM	Perform constructability review	Project profit	
TM12_1	Multi-story Condo	PM	Cost Module	Roof structure cost	Each test module is repeated 10 times.
TM13_1	High-rise Commercial	PM	Change Order	Elevator pit	1. Each test module is repeated 10 times. 2. Change Order process should have RFI procedure.
TM14_1	High-rise Commercial	Superintendent	Change Order	Elevator pit	
TM15_1	Industrial	Superintendent	Safety	Signs	Each test module is repeated 10 times.
TM16_1	Industrial	Superintendent	Maintain Project Site Documents	Field reports	Each test module is repeated 10 times.

Table E-2 Performance time for information retrieval in the EXTET system (Each TM is tested 10 times. Time is in milliseconds.)

TM No.	R	SQL	Ma	1	2	3	4	5	6	7	8	9	10	$\overline{PTM_{i\_j}}$	STDVi
TM1_0	73	73	503	7003	6912	6990	6992	6992	6994	6992	7005	7001	7000	6,988.10	27.27
TM1_1	6	6	62	741	741	741	740	745	739	744	741	741	743	741.60	1.84
TM2_0	73	73	722	3700	4100	4034	3903	4008	3998	3872	4103	4200	3632	3,955.00	180.37
TM2_1	14	14	23	491	491	540	510	491	521	530	501	511	520	510.60	17.26
TM3_0	73	73	841	4591	5300	3843	4740	3891	3890	4201	4104	3998	4136	4,269.40	468.98
TM3_1	5	5	53	445	438	456	426	634	433	532	497	480	498	483.90	62.87
TM4_0	73	73	940	6428	6979	6530	6098	6409	6083	6086	6080	6658	6680	6,403.10	314.57
TM4_1	17	17	78	234	342	433	483	497	447	463	466	409	452	422.60	79.25
TM5_0	73	73	322	1579	1808	1579	1469	1769	1670	1798	1577	1980	1977	1,720.60	174.99
TM5_1	10	10	39	257	240	232	265	232	268	263	244	241	242	248.40	13.62
TM6_0	73	73	180	572	597	574	545	598	509	544	580	601	499	561.90	36.52
TM6_1	8	8	32	176	144	102	78	157	186	152	146	157	167	146.50	32.97
TM7_0	73	73	196	674	579	508	524	646	648	593	603	698	548	602.10	64.03
TM7_1	24	24	9	32	12	45	32	25	19	31	22	25	20	26.30	9.17
TM8_0	73	73	483	4709	6439	5811	5319	5892	6242	6245	6002	5924	5770	5,835.30	502.94
TM8_1	29	29	181	279	384	376	351	356	371	368	299	380	347	351.10	35.23
TM9_0	73	73	1220	8975	5856	9094	9004	10472	8906	9701	8970	9801	8919	8,969.80	1,211.11
TM9_1	32	32	386	1235	2398	3012	1909	2668	2755	2748	2389	2347	2304	2,376.50	505.35
TM10_0	73	73	943	7734	7318	7981	7931	7439	7949	7903	7668	7884	7454	7,726.10	244.79
TM10_1	20	20	90	943	975	942	956	899	962	804	947	946	963	933.70	49.86
TM11_0	43	43	442	4793	4979	4763	4860	4084	4787	4623	5021	4896	3904	4,671.00	376.49
TM11_1	27	27	467	529	503	599	534	593	563	523	460	490	587	531.64	49.30
TM12_1	23	23	432	3980	3662	3798	3655	3329	3895	3672	3768	3093	3805	3,665.70	266.72
TM13_1	43	43	269	4743	2882	4762	2526	4362	2524	3979	3652	3763	3656	3,684.90	827.85
TM14_1	21	21	30	433	453	422	234	362	332	247	362	278	298	342.10	77.91
TM15_1	16	16	66	642	594	565	540	681	638	603	593	683	328	586.70	102.09
TM16_1	17	17	63	742	432	535	743	632	482	473	534	543	783	589.90	126.58

APPENDIX F
MODEL VALIDATION

After finishing the conceptual design discussion and implementing the UMDA system, the system was used to retrieve information for some queries as shown in Appendix E. The execution time was recorded and the related statistical indices were calculated. The next question to answer is "Has the most relevant material been found or are some important items missing?" But getting "everything" while at the same time avoiding "junk" is difficult to accomplish. Jizba (2000) proposed the use of two parameters to measure how well a search has performed. These two parameters are precision and recall. They are the basic measures used in evaluating search strategies.

Figures F-1 and F-2 show the standard information retrieval precision and recall techniques proposed by Jizba (2000). As shown in Figure F-1, precision and recall measures assume the following:

- There is a set of records in the database, which is relevant to the search topic (i.e. query).
- Records are assumed to be either relevant or irrelevant (these measures do not allow for degrees of relevancy).
- The actual retrieval set may not perfectly match the set of relevant records.

Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. It is usually expressed as a percentage (Jizba 2000; Tsuruoka and Tsujii 2004).

R = Percentage of all relevant documents that are found by a search

$$= \frac{\#of\_relevant\_items\_retrieved}{\#of\_relevant\_items\_in\_collection} \qquad \text{(F-1)}$$

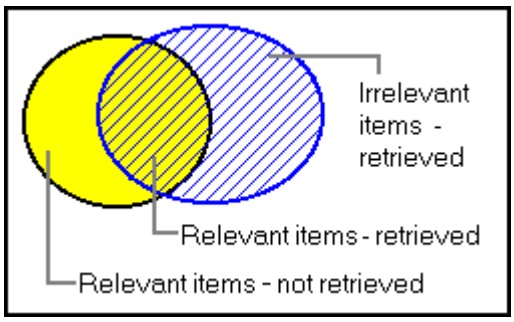
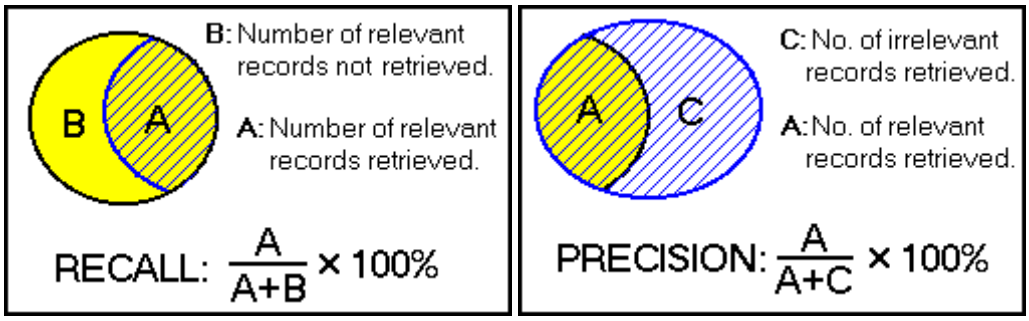


Figure F-1 Explanation of precision and recall assumptions (Jizba 2000) (The solid colored oval represents the set of records in the database, which is relevant to the search topic. The hatched oval shows the set of items retrieved by using a certain information retrieval system.)

Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage (Jizba 2000; Tsuruoka and Tsujii 2004). Figure F-2 shows the calculation of these two measures.

P = Percentage of retrieved documents that are relevant

$$= \frac{\#of\_relevant\_items\_retrieved}{\#of\_items\_retrieved} \qquad \text{(F-2)}$$



(A) (B)

Figure F-2 Explanation of precision and recall measures (Jizba 2000). A) Shows the calculation for Recall. B) Shows the calculation for Precision.

Suppose a database has 10 relevant items to a query. A DBMS system retrieves 8 items. Six (6) out of the eight (8) items are relevant. Then,

R = 6/10 = 60% P = 6/8 = 75%

The UMDA system implemented user profiles and task profiles to configure the GUI layer and the application layer of the information retrieval system. To validate the performance of the UMDA system regarding its quality of search results, standard information retrieval precision/recall techniques were used. A pool of relevant records (to a query) was manually identified. Then the list of all possible answers retrieved manually was compared with the list of all possible answers retrieved by the UMDA system. The next step was to determine what proportion of the pool the search has retrieved.

As suggested by (Cox et al. 2003), to evaluate construction management performance, we can use some Key Performance Indicators (KPI) to measure construction productivity and project management. KPIs are the methods management uses to evaluate employee performance. Quantitative performance indicators include Units/MH, $/Unit, Cost, On-Time Completion, Resource Management, Quality Control/Rework, Percent Complete, Earned Man-Hours, Lost Time Accounting, and Punch List. Qualitative Performance Indicators include Safety, Turnover, Absenteeism, and Motivation. The Test Modules shown in Table E-1 and E-2 in Appendix E reflect most of the quantitative KPIs and qualitative KPIs except Earned Lost Time Accounting and Absenteeism. The reason is because the construction project database used for the validation process did not have these items. So by using the Test Modules developed in Table E-1 and E-2 in Appendix E, the usefulness of the UMDA system to construction project management can be validated.

LIST OF REFERENCES

Aalami, F. B. (1998). "Using method models to generate 4D production models." PhD thesis, Stanford University, Stanford, CA.

Adolph, S., and Bramble, P. (2003). *Patterns of effective use cases*, Addison-Wesley, Boston, MA.

Alban, H. (2000). *Distributed objects and object wrapping*, Auerbach Publications, Boca Raton, FL.

Allen, J. F. (1979). "A Plan-based approach to speech act recognition." *131/79*, Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Allen, R. (2004). "Workflow: An introduction." *The Workflow Management Coalition*, <http://www.wfmc.org/information/Workflow-An_Introduction.pdf> (Feb. 23, 2004).

Allison, L., and Dix, T. I. (1986). "A bit-string longest-common-subsequence algorithm." *Information Processing Letters*, 23(6), 305-310.

Alpert, S. R., Karat, J., Karat, C.-M., Brodie, C., and Vergo, J. G. (2003). "User attitudes regarding a user-adaptive eCommerce web site." *User Modeling and User-Adapted Interaction*, 13(4), 373-396.

Al-Rasheed, K. (1997). "Construction schedule logic development: the impact of visualization." PhD thesis, University of Colorado, Denver, CO.

Anderson, J. R., and Milson, R. (1989). "Human memory: An adaptive perspective." *Psychological Review*, 96, 703-719.

AppleGroup. (2004). "WebObject development workflow." *Apple Group*, <http://www.apple.com/webobjects/getting_started.html> (Oct. 16, 2003).

ATG. (2000). "Dynamo product suite." *Art Technology Group*, <http://www.atg.com/products/highlights> (Nov. 10, 2002).

Baeza-Yates, R., and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*, Addison Wesley Longman Publishing, Santiago, Chile.

Belkin, N. (1978). "Information concepts for information science." *Journal of Documentation*, 34, 55-85.

Bell, L. C., and Elzarka, H. (1992). *Expert System Application and Tutorial*, the Construction Industry Institute and the University of Texas at Austin, Clemson University, Clemson, South Carolina.

Berger, M. (2004). "Microsoft patches security holes, old and new." *CNN.com/SCI-TECH*, <http://www.cnn.com/2002/TECH/internet/02/13/ms.security.patches.idg/> (Feb. 5, 2003).

Bernal, J. D. (1948). "Preliminary analysis of pilot questionnaire on the use of scientific literature." *Proc., Royal Society: Royal Society Scientific Information Conference*, London, 101-102, 589-637.

Berry, D., Hungate, C., and Temple, T. (2003). "Ease of use." *IBM Systems Journal*, 42(4), 556.

Blackwell, A., and Green, T. (2003). *Notational systems--The cognitive dimensions of notations framework*, Morgan Kaufmann Publishers, San Francisco, CA.

BRC. (2004). "Building research council." *University of Illinois at Urbana-Champaign Research*, <http://www.arch.uiuc.edu/> (Jul. 22, 2003).

Burns, R. W., and Hasty, R. W. (1973). *A survey of user attitudes toward selected services offered by the Colorado State University Libraries*, Colorado State University Libraries, Fort Collins, CO.

Cassidy, R. (2004). "Contractors are getting with IT." *Building Design & Construction*, 5(1), 3.

CFMA. (2004). "Contractors' Use of Internet Technology." *Construction Financial Management Association*, <http://www.cfma.org> (Dec. 10, 2003).

Chaudhuri, S., and Dayal, U. (1996). "An Overview of Data Warehousing and OLAP Technology." *Proc., SIGMOD Record*, 26(1), 12-34.

Chen, Q., Hsu, M., and Dayal, U. (2000). "A Data-Warehouse/OLAP Framework for Scalable Telecommunication Tandem Traffic Analysis." *Proc., 2000 IEEE 16th International Conference on Data Engineering*, San Diego, California, 201-210.

Cho, W.C. and Richards, D. (2004). "Improvement of Precision and Recall for Information Retrieval in a Narrow Domain: Reuse of Concepts by Formal Concept Analysis." *Dept. of Computing, Macquarie University*, <http://www.comp.mq.edu.au/~richards/papers/WI04.pdf> (Dec. 17, 2004).

Citadon. (2004). "Citadon features." *Citadon, Inc.* <http://www.citadon.com> (Dec. 22, 2003).

Clark, P. A. (1972). *Action research and organizational change*, Noordhoff, Leyden, The Netherlands.

Cohen, A. L., Stachel, B., Foley, S., Cash, D., and Muller, M. (2004). *PeopleFlow: Shared documents and ad hoc workflow*, IBM Watson Research Center, Westchester County, New York.

Cohen, P. R., and Perrault, C. R. (1979). "Elements of a plan-based theory of speech acts." *Cognitive Science*, 3, 177-212.

Computing-Dictionary. (2004). "Computing Dictionary." *Farlex, Inc., Huntingdon Valley, PA.* <http://computing-dictionary.thefreedictionary.com> (Sep. 7, 2004).

Cooper, A. (2003). *About Face 2.0: The Essentials of Interaction Design*, Wiley Publishing, Inc., Indianapolis, IN.

Cox, R. F., Issa, R. R. A., and Ahrens, D. (2003). "Management's Perception of Key Performance Indicators for Construction." *Journal of Construction Engineering and Management*, 129(2), 142-151.

CSINet. (2004). "All About MasterFormat 2004 Edition." *The Construction Specifications Institute, Alexandria, VA.* <http://www.csinet.org/s_csi/sec.asp?CID=253&DID=4495> (Aug. 11, 2004).

Davis, J., Kay, J., Kummerfeld, B., Poon, J., Quigley, A., Saunders, G., and Yacef, K. (2004). "Using workflow, user modeling and tutoring strategies for just-in-time document delivery." *Smart Internet Technology CRC*, <http://www.smartinternet.com.au/SITWEB/publication/files/105_$$$_7711/P04_008.pdf> (May 23, 2003).

Debons, A. (1980). "Foundations of information science, in Theory and application of information research." *Proc., Mansell: Second international research forum on information science*, London, 112-136.

Farradane, J. (1976). "Towards a true information scientist." *Information Scientist*, 10, 91-101.

FIATECH. (2004). "Strategic over view: Capital projects technology roadmap initiative (version 1)." FIATECH <http://www.fiatech.org> (Nov. 16, 2003).

Finin, T. (1989). *A general user modeling system*, Springer-Verlag, New York.

Finin, T., and Drager, D. (1986). "GUMS--A General User Modeling System." *Proc., 1986 Canadian Society for Computational Studies of Intelligence (CSCSI-86)*, Montreal, Canada, 43-55.

Fischer, G. (1993). *Shared knowledge in cooperative problem-solving systems-integrating adaptive and adaptable components*, Elsevier Science Publishers, Amsterdam, Netherlands.

Fischer, G. (2000). "User Modeling in Human-Computer Interaction." *Proc., UMUAI 10th Anniversary Issue*: *User Modeling and User-Adapted Interface*, Honolulu, HI, 1-18.

Forgy, C. L. (1982). "Rete: A fast algorithm for the many pattern/many object pattern match problem." *Artificial Intelligence*, 19(1), 17-37.

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, Reading, MA.

Garza, J. M., and Howitt, I. (1997). *Wireless communication and computing at the construction jobsite*, The Construction Industry Institute and The University of Texas at Austin, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Grand, M. (2002). *Patterns in Java: a catalog of reusable design patterns illustrated with UML*, Wiley Pub., Indianapolis, IN.

HCIbib. (2004). "Human Computer Interaction Bibliography." *HCI Bibliography* <http://www.hcibib.org/> (Oct. 18, 2004).

Heinckiens, P. M. (1998). *Building scalable database applications-object-oriented design, architectures, and implementations*, Addison-Wesley, Reading, MA.

Helander, M. G., Landauer, T. K., and Prabhu, P. V. (1997). *Handbook of Human-Computer Interaction*, Elsevier Science Ltd., Amsterdam, Netherlands.

Hendrickson, C., and Au, T. (1989). *Project management for construction: fundamental concepts for owners, engineers, architects, and builders*, Prentice Hall, Englewood Cliffs, NJ.

Herbrich, R., Zaragoza, H., and Hill, S. (2004). “A statistical analysis of the Precision-Recall graph.” *Microsoft Research, Cambridge University*, <http://www-2.cs.cmu.edu/~lafferty/ml-stat/zaragoza.ppt> (Dec. 18, 2004).

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1997). "The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users." *Proc., American Association for Artificial Intelligence: Fourteenth Conference on Uncertainty in Artificial Intelligence (AAAI-97)*, Providence, RI, 1-10.

IBM. (2004). "IBM Tivoli business systems manager release notes version 2.1." *Watson Research Center*, < http://www.ibm.com> (Sep. 21, 2003).

Ibushi, K., Collier, N., and Tsujii, J. (2004). “Classification of MEDLINE abstracts.” *Dept. of Information Science, University of Tokyo*, <http://www.jsbi.org/journal/GIW99/GIW99P35.pdf> (Dec. 19, 2004).

Jizba, R. (2004). "Measuring search effectiveness." Creighton University, Omaha, Nebraska, <http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html> (Dec. 4, 2004).

Johnson, R. (2003). *Expert one-on-one J2EE design and development*, Wrox, IN.

Karttam, N. A., and Levitt, R. E. (1990). "Intelligent planning of construction projects." *Proc., Journal of Computing in Civil Engineering*. 4(2), 6.

Kay, J., and Kummerfeld, R. J. (1996). "User model based filtering and customisation of web pages." *Proc., UM96 workshop: Workshop on User Modeling for Information Filtering on the World Wide Web (Fifth International Conference on User Modeling)*, Honolulu, HI, 2-10

Kobsa, A. (2001). "Generic User Modeling Systems." *User Modeling and User-Adapted Interaction*, 11, 49-63.

Kobsa, A., and Wahlster, W. (1989). "User models in dialog systems." Springer-Verlag, Heidelberg, Berlin.

Kotler, P., Armstrong, G., Brown, L., and Adam, S. (1998). *Marketing*, Prentice Hall, Englewood Cliffs, NJ.

Kriwaczek, F. (2004). "On Line Analytical-Processing (OLAP)." *Imperial College London*, <http://www.doc.ic.ac.uk/~frk/frank/kmt/OLAP%20handout.pdf > (Feb. 3, 2003).

Kuniavsky, M. (2003). *Observing the user experience: A practitioner's guide to user research*, Morgan Kaufmann Publishers, San Francisco, CA.

Lampson, B. W. (1983). "Hints for computer system design." *Proc., ACM: 9th Symposium on Operating Systems Principles*, Bretton Woods, NH, 33-48.

Lazar, Z. P., and Holfelder, P. (1997). "Web database connectivity with scripting languages." *World Wide Web Journal*, 2(2), 4.

Mackay, W. E. (1991). "Triggers and barriers to customizing software." ACM Press, New York.

Mays, P. C., and Novitski, B. J. (1997). *Construction administration-An architect's guide to surviving information overload*, John Wiley & Sons, Inc. Indianapolis, IN.

McQuail, D. (1975). *Communication*, Longman, New York.

McTear, M. (1993). "Artificial Intelligence Review." Special issue on user modeling, MIT Press, Cambridge, MA.

Mellor, S. J., and Balcer, M. J. (2002). *Executable UML: A foundation for model-driven architecture*, Addison-Wesley, Boston, MA.

OAGIS. (2004). “OAGIS 8.0 Design Document.” *Open Applications Group*, <http://www.openapplications.org > (Oct. 18, 2003).

Olson, S. (2003). "Developer Tools for the Sybase Enterprise Product Suite for Mac OS X." *Proc., O’Reilly: O'Reilly Mac OS X Convention*, Westin Santa Clara, CA, 1-39.

OMG. (2004). "Object Management Group." OMG, <http://www.omg.org> (Sep. 22, 2003).

Ott, R. L., and Longnecker, M. (2001). *An introduction to statistical methods and data analysis*, 5th Ed., Duxbury-Thompson Learning, Pacific Grove, CA.

Outsell. (2004). “Make analysis for the information industry.” *Outsell, Inc.* <http://www.outsellinc.com/outsell/team/> (May 27, 2003).

Pendse, N. (2004). "What is OLAP?" <http://www.olapreport.com/fasmi.htm> (Nov. 18, 2003).

Pendse, N., and Creeth, R. (1995). *The OLAP report succeeding with Online Analytical Processing*, Business Intelligence Inc., Norwalk, CT.

Primavera. (2004). "Primavera Features." *Primavera, Inc.* <http://www.primavera.com> (Jan. 25, 2003).

Reinhardt, J. (2003). "Navigational models for effective and efficient interaction with integrated product and process models on construction site." PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Rich, E. (1979a). "Building and exploiting user models." PhD Thesis, Carnegie-Mellon University, Pittsburgh, PA.

Rich, E. (1979b). "User modeling via stereotypes." *Cognitive Science*, 3, 329-354.

Rich, E. (1989). *Stereotypes and user modeling*, Springer-Verlag, New York.

RosettaNet. (2004). “Introduction to RosettaNet: XML Working Group 5-16-01 (Minus RosettaNet Template).” *XML.Gov*, <http://xml.gov/presentations/rosettanet/RNet1a/> (Jan. 26, 2003).

Shan, Y., and Earle, R. H. (1998). *Enterprise computing with Objects-from Client/Server environments to the Internet*, Addison-Wesley, Reading, MA.

Simon, H. A. (1992). "What is an "explanation" of behavior?" *Psychological Science*, 3, 150-161.

Sleeman, D. (1985). "UMFE: A user modeling front-end subsystem." *International Journal of Man-Machine Studies*, 23, 71-88.

Soboroff, I. (2004). “Information retrieval evaluation.” *Dept. of Computer Science and Electrical Engineering, University of Maryland Baltimore County,* <http://www.csee.umbc.edu/~ian/irF02/lectures/09Evaluation.pdf> (Dec. 18, 2004).

SphinxSoftware. (2004). "Sphinx software." *Sphinx Software Inc.*, <http://www.sphinxsoftware.com/default.asp> (Feb. 15, 2004)

Suchman, L. A. (1987). *Plans and situated actions*, Cambridge University Press, Cambridge, UK.

Sugiyama, K., Hatano, K., and Yoshikawa, M. (2004). "Adaptive web search based on user profile constructed without any effort from users." *Proc., WWW2004: The Thirteenth International World Wide Web Conference*, New York, NY, 675-684.

Tannenbaum, A. (2002). *Metadata solutions: Using metamodels, repositories, XML, and enterprise portals to generate information on demand*, Addison-Wesley, Boston, MA.

Thomsen, E. (2002). *OLAP solutions: building multidimensional information systems*, John Wiley & Sons, Inc., New York.

Timberline. (2004). "Timberline features." *Timberline*, <http://www.timberline.com> (Mar. 1, 2003).

Tockey, S. (2004). “Object Modeling with UML.” *Construx Software Builders*, < http://www.omg.org/news/meetings/workshops/presentations/eai_2001/tutorial_monday/tockey_tutorial/1-Intro.pdf> (Dec. 27, 2004)

Tsuruoka, Y. and Tsujii, J. (2004). “Boosting Precision and Recall of dictionary-based protein name recognition.” *Dept. of Computer Science, University of Tokyo,* <http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/papers/ad03bio.pdf> (Dec. 18, 2004).

UCD. (2004). “User centered design.” *IBM*, <http://www.ibm.com/ucd> (Mar 5, 2003).

Vorster, M. C., Magrogan, S. A., and McNeil, B. W. (1998). "A breakthrough project delivery system that improves performance by reforming owner, contractor, supplier relationships." Virginia Polytechnic University, Blacksburg, VA.

Vredenburg, K., Ritsko, J., and Seidman, D. (2003). "Preface." *IBM Systems Journal*, 42(4), 515-517.

Weinshall, T. D. (1979). *Managerial communication: concepts, approaches and techniques*, Academic Press, London.

White, B. A. (2000). *Software configuration management strategies and rational ClearCase: A Practical Introduction*, Addison-Wesley, Boston, MA.

Whittaker, S., and Schwartz, H. (2004). "Big Boards: Using large displays in project management." *IBM Watson Research Center, Armonk, New York*. <http://domino.research.ibm.com/cambridge/research.nsf/0/c03fce974b1a2fc28525698b0071f952?OpenDocument> (Jan. 21, 2004).

Wiedenbeck, S., Zila, P. L., and McConnell, D. S. (1995). "End-User training: An empirical study comparing on-line practice methods." *Proc., CHI 1995: Human Factors in Computing Systems*, Denver, CO, 74-81.

Wilson, J. L., Lennox, J., and Bryan, P. Y. (1994). *3-D modeling as a tool to improve integrated design and construction*, Construction Industry Institute, the University of Texas at Austin, Austin, TX.

Wilson, M. (1993). "Knowledge Engineering." Proc., *SOFSEM'93: Current Trends in Theory and Practice of Informatics*, Hrdonov, Sumava, Czech Republic, 17.

Wilson, T. D. (2000). "Recent trends in user studies: action research and qualitative methods." *Information Research*, 5(3), 23.

Xie, H., Issa, R., and O'Brien, W. (2003). "User model and configurable visitor for construction project information retrieval." *Proc., The Fourth Joint International Symposium on Information Technology in Civil Engineering--Toward a Vision for Information Technology in Civil Engineering*, Nashville, TN, 1-11

Yianilos, P. N., and Kanzelberger, K. G. (1997). "The LIKEIT intelligent string comparison facility." NEC Research Institute, Princeton, NJ.

BIOGRAPHICAL SKETCH

Haiyan Xie earned her PhD from the M.E. Rinker, Sr. School of Building Construction at the University of Florida (UF) in Gainesville. While earning that degree, she also worked as an assistant professor in the Department of Construction Management, University of Arkansas at Little Rock (UALR). She has worked for Taylor Contractors of Florida, Inc. as a Project Manager before she went to UALR. Her research interests are in the areas of analysis, design, and implementation of construction information systems; user modeling; and project-management improvement.

She also holds an MS in computer engineering, Department of Computer and Information Science and Engineering (CISE), from the University of Florida. She received a BE and a ME in construction engineering and management from the Xi'an University of Architecture and Technology, China. She is a member of American Society of Civil Engineers (ASCE), Associated Schools of Construction (ASC), and Association of American Colleges and Universities (AACU).