XML-Based Context Maps and CMapViewer Application

Qiao Li

A Thesis
in
the Department of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

June, 2005

Your file Votre référence
ISBN: 0-494-10291-8
Our file Notre référence
ISBN: 0-494-10291-8

Abstract

XML-Based Context Maps and

CMapViewer Application

Qiao Li

Context maps are a kind of high-level notational technology which has been increasingly used in information system and software engineering over the last few years. Recent efforts using Microsoft (MS) Excel spreadsheets to represent the context maps have shown some progress toward meeting the various needs for the development of context maps. However, as more applications developed for the manipulation of context maps, complying with the technical requirements imposed by MS Excel limits their functionalities and further development. Extensible Markup Language (XML) and related technologies may be used to help to overcome current limitations. This thesis introduces a new way of XML-based context maps to represent formatted knowledge. Structured information stored in XML format is very easily exchanged among different applications. This makes XML a natural choice to be used as a representation format for the context maps. A Java-based application named CMapViewer has been developed to view and manipulate the XML-based context maps and it demonstrates an easy, intuitive and powerful solution for manipulating the formatted knowledge. In addition, an appropriate tool has been developed to allow users to transform current context map documents into the XML-based formats. This service demonstrates the implementation of solutions for the interoperability of XML-based context maps.

Acknowledgments

The author particularly wishes to express his thanks to Dr. Wojciech M. Jaworski and Prof. Peter Grogono for their knowledgeable suggestions and helpful guidance to the completion of this project.

The author also wishes to express his thanks to Mrs. Halina Monkiewicz for her support and collaboration.

This thesis is dedicated to author's loving and supportive wife Ying for her patience and understanding during his master study and the writing of this thesis.

Table of Contents

List of Figures

List of Examples

Chapter 1 : Introduction

1.1 Background

Context maps are a kind of high-level notational technology which was first introduced by Dr. W. M. Jaworski [1-12]. By using the notation of the context maps, it allows efficient recovery and modeling of generic schemata. This technology can be applied in many domains, such as modeling, mining, evaluation of contexts, enterprises, methods, processes, projects, artifacts, databases, websites, information system, knowledge models with generic templates, domain experts, and proprietary notational technology. Thus, it has been practically used in information system and software engineering.

The context maps represent the information knowledge set by linking the relationships among different information nodes in spreadsheet vertical columns. Thus, the relevant parts of the information knowledge set can be pointed to the appropriate topics. With the wide needs of information representation, the context maps have been recently used in information system and software engineering. Dealing with large amounts of information data and representing knowledge assets in an appropriate format represent a formidable challenge.

Recent efforts using the spreadsheet in MS Excel to represent the knowledge assets but providing a user-friendly graphical interface (GUI) with programmed Visual Basic (VB) macros to manipulate the formatted knowledge [26] have shown some progress on the efficient visualization tools to meet various needs for the studies on the context maps.

Numerous studies on the developing MS Excel-based applications have been conducted [25, 26 and 29]. The latest application CONTEXT+ [26] demonstrated the functionalities to increase the efficiency of data manipulation and visualization on the context maps especially for representing the context maps in a 3P-able (Plug-able, Process-able, and Pattern-able) format. However, these applications have limits on the size of the context maps, the time consuming on the queries of a large context map, usability issues of the GUI and dependency on the third-party software. Therefore, exploring a new format to represent the context maps and providing a more efficient and powerful tool to manipulate context maps are necessary and important innovations.

1.2 Objectives

The overall objectives of this study are outlined below:

1) Exploring a new format to represent the context maps by investigating the feasibility of using XML as the document container for the context maps.
2) Developing a new platform independent application to display and manipulate the XML-based context maps by applying Java language to build the program.
3) Enhancing the functionalities of the manipulation on the context maps and providing a highly intuitive and user-friendly GUI for easy learning and using this newly developed application.
4) Providing a tool to transform the existing Excel-based context maps to the XML-based context maps by choosing Extensible Stylesheet Language (XSL) to perform the task between two different document formats.

Chapter 2 : Related Work

2.1 Introduction

In the field of information system and software engineering, recent interest in modeling notations and techniques has resulted in a rapidly growing body of research work. Lots of research results have a significant and immediate impact on the current state of the practice in the software community and lay firm foundations for the development of more sophisticated model-based development techniques. With a deeper understanding of modeling languages and techniques, the software and system developers can develop more effective applications.

This chapter will present related work in the areas of modeling notations and techniques. Some important types of modeling notations and techniques will be described. However, the details of Context Maps notational technology and its tools will be described specially in Chapter 3.

2.2 Sowa's Conceptual Graphs

In the early 1980s, J.F. Sowa introduced the conceptual graph (CG) theory [13] which provides a knowledge representation framework consisting of a form of logic with a graph notation and integrating several features from semantic net and frame representations. Conceptual graphs [14, 15 and 16] are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. Conceptual graphs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.

Conceptual graphs are formally defined by an abstract syntax that is independent of any notation, but the formalism can be represented in either graphical or character-based notations. Sowa demonstrated an example of conceptual graphs using Figure 2-1 (sourced from http://www.cs.uah.edu/~delugach/CG/Sowa-intro.html web site) which shows the display form (DF) of a conceptual graph that represents the English sentence John is going to Boston by bus.

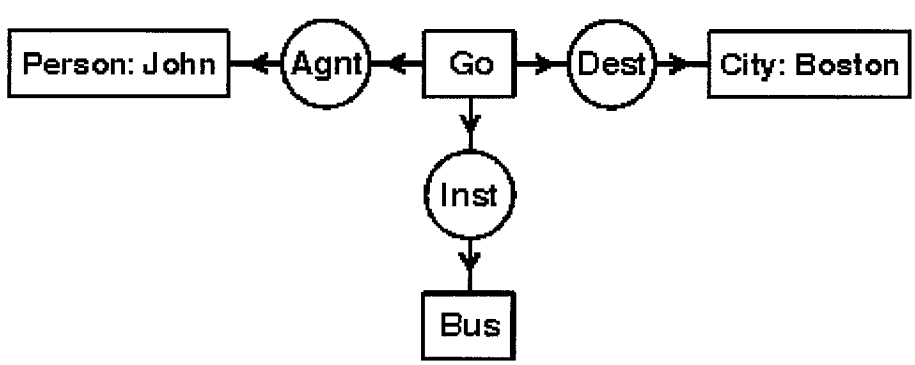


Figure 2-1: CG Display Form for "John is going to Boston by bus."

In DF, concepts are represented by rectangles: [Go], [Person: John], [City: Boston], and [Bus]. Conceptual relations are represented by circles or ovals: (Agnt) relates [Go] to the agent John, (Dest) relates [Go] to the destination Boston, and (Inst) relates [Go] to the instrument bus. The arcs that link the relations to the concepts are represented by arrows. For relations with more than two arguments, the arcs are numbered.

The linear form (LF) is intended as a more compact notation than DF, but with good human readability. It is exactly equivalent in expressive power to the abstract syntax and the display form. Following is the LF for Figure 2-1:

```
[Go]-
     (Agnt)->[Person: John]
     (Dest)->[City: Boston]
     (Inst)->[Bus].
```

In this form, the concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles. The hyphen on the first line indicates that the relations attached to [Go] are continued on subsequent lines.

Both DF and LF are designed for communication with humans or between humans and machines. For communication between machines, the conceptual graph interchange form (CGIF) has a simpler syntax. Following is the CGIF for Figure 2-1:

```
[Go *x] (Agnt ?x [Person 'John']) (Dest ?x [City 'Boston']) (Inst ?x [Bus])
```

CGIF is intended for transfer between computer systems that use CGs as their internal representation. For communication with systems that use other internal representations, CGIF can be translated to another logic-based formalism called the Knowledge Interchange Format (KIF):

```
(exists ((?x Go) (?y Person) (?z City) (?w Bus))
(and (Name ?y John) (Name ?z Boston)
(Agnt ?x ?y) (Dest ?x ?z) (Inst ?x ?w)))
```

Although DF, LF, CGIF, and KIF look very different, their semantics is defined by the same logical foundations. Semantic information expressed in any one of them can be translated to the others without loss or distortion. Formatting and stylistic information, however, may be lost in translations between DF, LF, CGIF, and KIF.

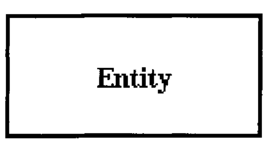
2.3 The Entity-Relationship Model

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [17] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design [18].

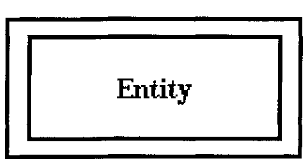
There are three basic elements in ER models:

- Entities are the "things" about which we seek information.
- Attributes are the data we collect about the entities.
- Relationships provide the structure needed to draw information from multiple entities.

An entity-relationship diagram (ERD) is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. The ER diagram notations (sourced from http://www.smartdraw.com/tutorials/software-erd/erd.htm web site) are described below:



Entity
An entity is an object or concept about which you want to store information.



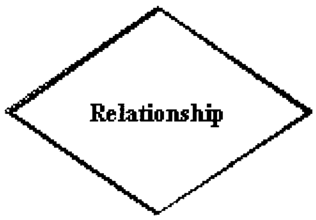
Weak Entity
A weak entity is dependent on another entity to exist.

Attributes
Attributes are the properties or characteristics of an entity.

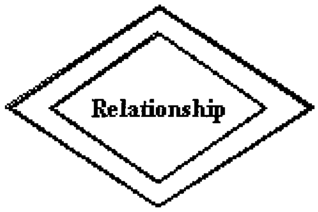
Key attribute
A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

Multivalued attribute
A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.

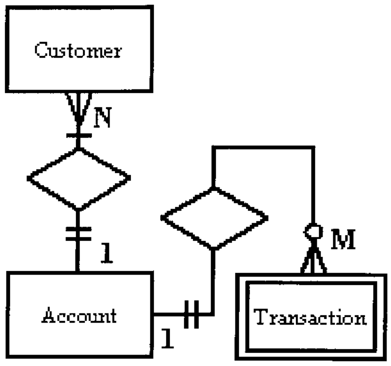
Derived attribute
A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



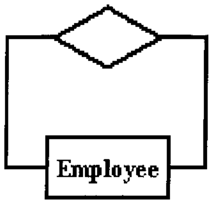
Relationships
Relationships illustrate how two entities share information in the database structure.



Weak relationship
To connect a weak entity with others, you should use a weak relationship notation.



Cardinality
Cardinality specifies how many instances of an entity relate to one instance of another entity.



Recursive relationship
In some cases, entities can be self-linked. For example, employees can supervise other employees.

2.4 Unified Modeling Language

In 1997 UML (Unified Modeling Language) [19, 20 and 21] became an OMG (Object Management Group) standard and since then it has become the de facto language for visualizing, specifying, and documenting the different models created in software development projects. UML has had a great impact on the software community in both the area of research and the area of practice. Its success has been impressive, as its use throughout the world for building applications in different domains and of different sizes testifies. Almost all industrial development environments (for example, Microsoft, IBM, Borland, as well as non-proprietary development environments) integrate UML modeling tools. UML (Unified Modeling Language) has become one of the most widely used standards for specifying and documenting information systems.

Notation plays an important role in modeling and the goal of the UML is to become a universal notation for creating models of object-oriented (OO) software. A model is a description of (part of) a system written in a well-defined language. There are many different types of UML notation. But at the centre of the UML are its modeling diagrams. The following are the descriptions of these modeling diagrams:

- Use case diagrams - used to show main elements and processes that outline the system.
- Class diagrams - used to refine the use case diagram and define a more detailed design of the system.
- Sequence diagrams - Stands for the interaction between different objects in the system that is time ordered.

- Collaboration diagrams - The collaboration diagram helps identify the interactions that each object has with other objects.
- Component diagrams - A component diagram shows the parts cut after the system has undergone the development phase.
- Object diagrams - The object diagram captures the state of different classes and their relationships or associations at a given point.
- Statechart diagrams - Represents the different states that objects in the system undergo during their life cycle.
- Activity diagrams - process flows are captured in the activity diagram and similar to the state diagram.
- Deployment diagrams - The deployment diagram captures the configuration of the runtime elements of the application.

2.5 Model Driven Architecture

MDA (Model Driven Architecture) [22, 23] is a recent initiative from the OMG that supports the definition of models as first class elements for the design and implementation of systems. According to the MDA approach, the most important activities are now modeling the different aspects of a system and then defining transformations from one model to another one in away that allows them to be automated. Focusing on model definition and leaving implementation details until the end make these models more portable, more adaptable to new technologies (e.g. .NET, J2EE or Web Services) and more interoperable with other systems regardless of the technology they use.

MDA defines three conceptual levels. At the first level, system requirements are modeled in a Computation Independent Model (CIM) that defines the system within an operating environment. At the next level it defines the Platform Independent Model (PIM). A PIM describes the system functionality, but without considering details about where and how this system is going to be implemented (e.g. a PIM can be independent from system distribution and the supporting object platform, such as CORBA, J2EE, .NET, etc.). The aim is then to transform a PIM into a target platform dependent model known as a PSM (Platform Specific Model). In this way a high degree of independency between the description of functionality and the implementation details of the target platform can be obtained.

The most important advantage of this approach is that it allows software engineers to define automatic transformations from PIMs to PSMs. By inputting the system PIM, a description of the PSM to be used to implement the system, and a set of transformation rules, we will be able to implement a system in the most automated way possible.

2.6 Business Process Modeling Notation

Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN) [24]. The BPMN 1.0 specification was released to the public in May, 2004. The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.

BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance.

A BPD is made up of a set of graphical elements. These elements enable the easy development of simple diagrams that will look familiar to most business analysts (e.g., a flowchart diagram). With a small set of notation categories, the reader of a BPD can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram. The four basic categories of elements are:

- Flow Objects: A BPD has a small set of core elements, which are the Flow Objects including Event, Activity and Gateway.
- Connecting Objects: Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. The Connecting Objects including Sequence Flow, Message Flow and Association provide this function.
- Swimlanes: Many process modeling methodologies utilizes the concept of swimlanes as a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN

supports swimlanes with two main constructs. The two types of BPD swimlane objects are Pool and Lane.

- Artifacts: BPMN was designed to allow modelers and modeling tools some flexibility in extending the basic notation and in providing the ability to additional context appropriate to a specific modeling situation, such as for a vertical market (e.g., insurance or banking). Any number of Artifacts can be added to a diagram as appropriate for the context of the business processes being modeled. The current version of the BPMN specification pre-defines only three types of BPD Artifacts, which are Data Object, Group and Annotation.

More technical information and further details of BPMN notations can be found at http://www.bpmn.org/ web site.

Chapter 3 : Context Maps

3.1 Background

The technology of the context maps was first introduced by Jaworski [9]. It was initially developed as a means for recovering and refining knowledge from legacy systems. This technology is used as a notation and method for representing systems architecture, structures, processes and reusable templates [1, 9]. Between the late 1970s and early 1980s, based on the conceptual graphs introduced by J. F. Sowa, it was named ABL, or Array Based Language [5]. It was then renamed ABL/W4 in the late 1980s [6]. W4 represents as what, when, where and why. In the early 1990s, Jaworski, by considering the existing notations and methodologies, named this technology jMap (jointed map) notation [6]. Jaworski first introduced the concept of the context maps and named this technology **Context Maps** in the late 1990s. Until now, the context maps can represent knowledge assets in a 3P-able format [25].

The context maps are a formal representation method for information system with a set of predefined formal notations [11]. In a context map, context can be defined by an aggregation of context tuples which is a generic association of set members cast in roles. In an extended spreadsheet, a column of roles and the related set members define the context tuples. From a graphical point of view, the context tuples are represented by a compounded edge and the connected compounded nodes. A directed edge object consists of the tail object, the middle object and the head object. While the context tuples can represent system behaviors, processes, tasks, procedures and programs, the aggregation of the context tuples will form a context map.

The notation of the context maps allows efficient recovery and modeling of the generic schemata for processes, objects and views in information systems [12]. Thus, this technology can be applied in many domains, such as modeling, mining, evaluation of contexts, enterprises, methods, processes, projects, artifacts, databases, websites, information system, knowledge models with generic templates, domain experts, and proprietary notational technology.

Recently, the technology of the context maps has been widely and increasingly recognized as an important methodology and a formalized notation for representing the knowledge in many domains. The jMaps technology was used as a tool for web content and structure mining by which the information on the process of using Breadth-First-Search algorithm to discover pages in the site, retrieve and analyze information from the web sites which is transformed to a series of joined maps [29]. Minghui Han [25] studied the possibility to use the technology of the context maps to represent associative data model by converting a set of context maps into the database or retrieving the data information from the database to the context maps. He demonstrated his application by using the MS Excel spreadsheet as the platform to display the associative model of data but using MS Access database to store a set of converted context maps. Jing Bai [28] described the procedures of converting the OPEN Process methodology to a 3P-able format by using the context maps technology. The functionalities and attributes of the context maps are demonstrated by modeling the OPEN Process Framework in terms of the concepts and relationships from the http://www.donald-firesmith.com website. The context maps can also represent the Due Diligence Strategy [27] and model the Unified Process Concepts, Rational Unified Process framework [30].

3.2 Context Maps – Syntax and Process

The syntax of the context maps is based on the Relationship-Oriented paradigms with related Sets (Concepts) and Set Members (Instances). In another word, the fundamental structure of a context map is to link the concepts and the instances together by their roles. Thus, in a context map, the relationships among the concepts and the instances are represented by kTuples (i.e., vertical columns in the map) and one kTuple consists of Set, Set Members and Role Tuples. The relating mechanism is implemented by allocating the roles to the sets in the schema and their instances to the set components in the map. Compared to the diagrams, maps are very compact and they can offer a rich context within a limited space on a computer screen. Maps are created or edited within an organized electronic sheet (i.e., MS Excel spreadsheet) that assures efficient manipulation of the relationships (columns) and the heavy reuse of components (rows).

The following example shows how to use a context map to represent a workflow in a Unified Process. In order to simplify the demonstration, a simple diagram (Figure 3-1: The Diagram of Workflow in Inception Phase of Unified Process) sourced from http://www.gen-strategies.com/images/Inception.htm is chosen as the input diagram with only the "Inception" phase in the Unified Process. In this phase, each stage can be transferred to its subsequent stage after achieving all the tasks involved in this stage.

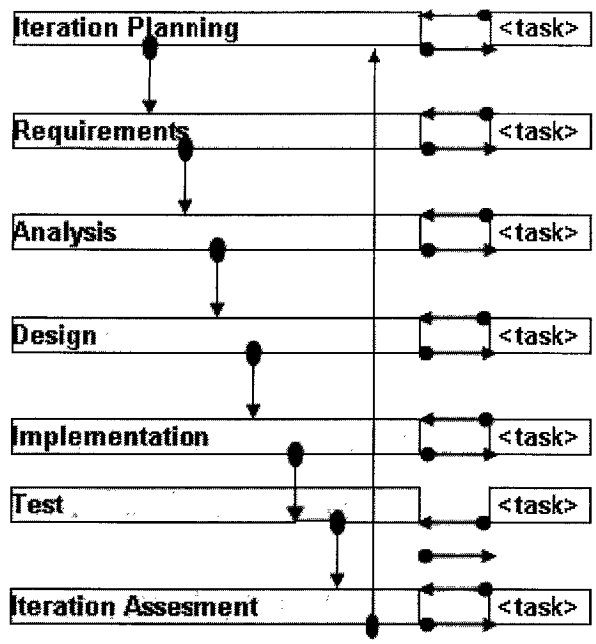


Figure 3-1: The Diagram of Workflow in Inception Phase of Unified Process

Figure 3-2 illustrates the context map of this workflow. In this context map, the Sets ({Context View}, {Phase}, {Task}, {Workflow} and {Author & Copyright}) are listed in the column 17. Under Set {Workflow}, the Iteration Planning, Requirements, Analysis, Design, Implementation, Test, and Iteration Assessment stages represent the Set Members. The arrows in the diagram represent the Set Member Roles in the columns 1-14 marked as "f" "t" "l" in the lowercase letters but the Set Roles, namely, the 'A', 'S' and 'L' are in uppercase letters. The terminology and symbols of the context maps are described in section 3.3.

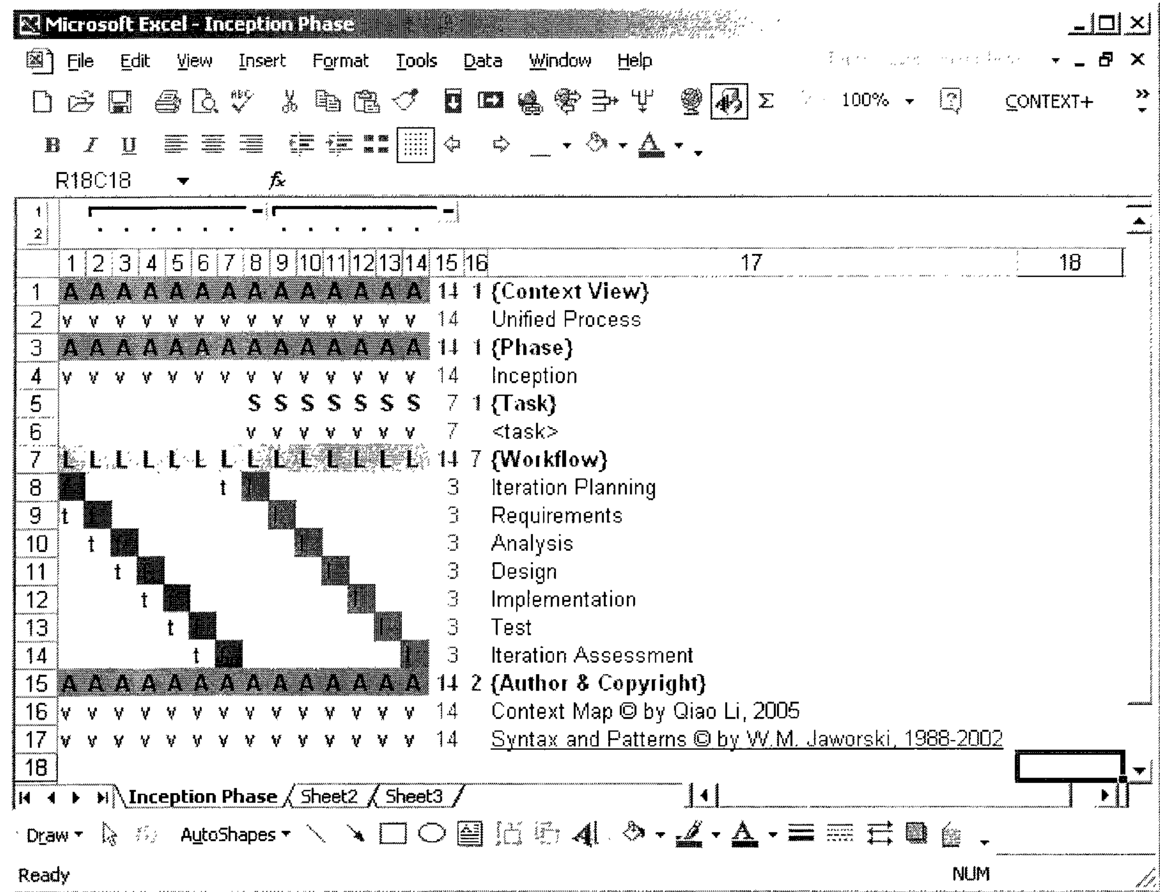


Figure 3-2: The Context Map of Workflow in Inception Phase of Unified Process

In general, the sets are listed at the right side of the map but indicated between a pair of bold curly brackets '{ }'. This way they can be regarded as the headings of the table rows and columns. Namely, the table rows are the set roles such as 'L' for the set {Workflow} and the table columns are the set members such as 'Iteration Planning', 'Requirements', 'Analysis', 'Design', 'Implementation', 'Test' and 'Iteration Assessment' for the same set {Workflow}. The listed roles at the left side of the map for each set or set member are the actual contents and relationships but by means of different notations. Each column needs to be read vertically in a way up or down the column and then across toward to the right side of the map to find which set or set member the specified role is referring to. With this kind of notation, we can easily represent the information knowledge on a spreadsheet. The overall schema of a context map is exhibited by hiding the set members and irrelevant columns (roles). Thus, an abstract view of the whole context map can be presented to the readers.

3.3 Context Maps – Terms and Notations

3.3.1 Terms

The terms of the context maps used in this report include definitions, acronyms and abbreviations. They are listed below:

- Context Map: A map which represents the relationships among different information sets and provides the functionality of arrays, graphs, relational tables, etc.
- JMaps: Abbreviation of Jointed map. A former name for the context map.
- CONTEXT+: A set of tools that was developed for processing the context maps.
- Context Tuple: A generic association of set members cast in roles. In an extended spreadsheet the column of roles and the related set members define the context tuples.
- kTuple: Abbreviation of Knowledge Tuple. It consists of Set, Set Members and Role Tuples. It has its own Schema and contains Identifier, Type and Descriptors.
- Schema: Work frame of a context map. The context map integrates Concepts and Concept Instances with abstract architecture.

- Set: The set name listed between a pair of bold curly brackets '{ }', such as {Workflow} in Figure 3-2.
- Set Members: The members of particular set, such as 'Iteration Planning', 'Requirements', 'Analysis', 'Design', 'Implementation', 'Test' and 'Iteration Assessment' for the set {Workflow} in Figure 3-2.
- Set Roles: Indicated in the uppercase letters in the spreadsheet cells, such as letters "A", "S", and "L" in Figure 3-2.
- Set Member Roles: Indicated in the lowercase letters or numbers in the spreadsheet cells, such as "v" "f", "t", and "l" in Figure 3-2.
- Cardinality of Roles: The number of the not-empty roles for each set or set member, such as the values in column 15 in Figure 3-2.
- Cardinality of Set Members: The number of set members under each set, such as the values in column 16 in Figure 3-2.
- Atom: The set or the set member.
- Spreadsheet: An electronic page in MS Excel which is used to store and process the data in the context maps.

3.3.2 Notations

The notations of the context maps can be illustrated as the following context map:

I	I	I	I	I	I	I	I	I	I	I	I	I	I	14	1	**{Context Tuple Unique Id}**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	14		Id
A	A	A	A	A	A	A	A	A	A	A	A	A	A	14	1	**{Context View}**
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14		Syntax
E	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**E**	14	2	**{Association}**

m	m													2		Is a
		m	m	m	m	m	m	m	m	m	m	m	m	12		Is valid member role for
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	14	35	**{Set}**
t														1		Set Roles
	t													1		Set Member Roles
f		t												2		A - (A)ggregation of columns (Context Tuples)
			t											2		E - (E)dge properties
				t										2		F - (F)low graph nodes
					t									2		L - (L)flow graph with cycles
						t								2		N - (N)ode properties
							t							2		V - (V)alue
								t						2		S - (S)equence
									t					2		G - (G)uard
										t				2		R - (R)esource
											t			2		O - (O)bject
												t		2		I - (I)dentifier
													t	2		X - Cartesian Product
														13		v - marker
														3		m - (m)iddle of 'arrow'
														3		f - tail of 'arrow'
														3		t - head of 'arrow'
														3		b = f/t
														3		f - (f)rom node
														3		t - (t)o node
														2		l - (l)oop
														2		b = f/t - both nodes component
														2		f - (f)rom node component
														2		t - (t)o node component
														2		l - (l)oop node component
														2		numerical value
														3		integer value
														2		y - yes
														2		o - otherwise
														2		c - (c)reate
														2		r - (r)ead
														2		u - (u)pdate
														2		d - (d)elete
														2		x - component of Cartesian Product
A	A	A	A	A	A	A	A	A	A	A	A	A	A	14	**1**	**{Author}**
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14		Syntax and Patterns © by W.M. Jaworski, 1988-2002

3.3.3 Context Maps Tools – CONTEXT+

MS Excel is a spreadsheet application used to perform financial calculations, statistical analysis, and other related operations. It enables you to turn your data into information with powerful tools to analyze, communicate, and share results. Now MS Excel clearly dominates the spreadsheet market. Thus, at the beginning of the development, MS Excel was chosen as the developing environment of creating and manipulating the context maps. All current tools developed for the context maps are built up on the MS Excel's spreadsheet by implementing a series of macros to automate the task. A macro is a series of commands and functions that are coded in a MS VB module and can be run whenever you need to perform the task. The latest version of the context maps tool is the CONTEXT+ [26] which will be described in the following sections.

3.3.3.1 Introduction to CONTEXT+

The development tool for the context maps, named "CONTEXT+", was initially developed by Jaworski to retrieve the useful information and generate the corresponding maps. With the developing of the context maps, more and more functionalities were added to CONTEXT+. The current version of CONTEXT+ can process 3P-able format maps and it has a user-friendly interface which allows user easily and efficiently manipulates the data especially for a big map by providing a number of options.

3.3.3.2 3P-able Format Maps

The new concept "3P-able" was first introduced by Jaworski [26] in order to meet the needs for dealing with the large amounts of data and represented knowledge assets during the development of the context maps. The 3P-able can be expressed as the following formula [26]:

3P-able = Plug-able + Process-able + Pattern –able

- **Plug-able**: Merge-able horizontally and vertically

One context map is a collection of different knowledge connected together with a logical manner. Many scattered knowledge assets concepts and relationships among these concepts can be processed and integrated into one context map, so that a formatted and cleared view can be presented to the users. In a word, the context map shall be able to merge separate knowledge into one view in the horizontal and vertical. The "Join Maps" function in the CONTEXT+ was developed according to this requirement.

- **Process-able**: Create-able, Read-able, Update-able, Delete-able

In order to speed up the efficiency of processing maps, CONTEXT+ enhanced these kinds of functionalities by providing more flexible and convenient tools to manipulate and view the information of the context maps.

- **Pattern -able**: Search-able and Navigate-able by patterns

The context maps are a kind of notational technology. This notational pattern strongly supports search and navigation. With the use of spreadsheet structure, a large amount of date can be organized logically. It can simplify the procedure in processing the context maps. The Query function of CONTEXT+ makes them available to the users.

3P knowledge is represented by (vertical) Knowledge Tuples (KTuples).

- KTuple consists of Set, Set Members, and Role Tuples.
- KTuple includes its own Schema.
- KTuple contains Identifier, Type and Descriptors.

3.3.3.3 Functionalities of CONTEXT+

CONTEXT+.xls file is a working platform for the context maps development. The built-in CONTEXT+ macro needs to be enabled after opening the CONTEXT+.xls (Figure 3-3). Its functionalities then can be accessed from the toolbar (see the bold rectangle on the toolbar in Figure 3-4: CONTEXT+ Macro).

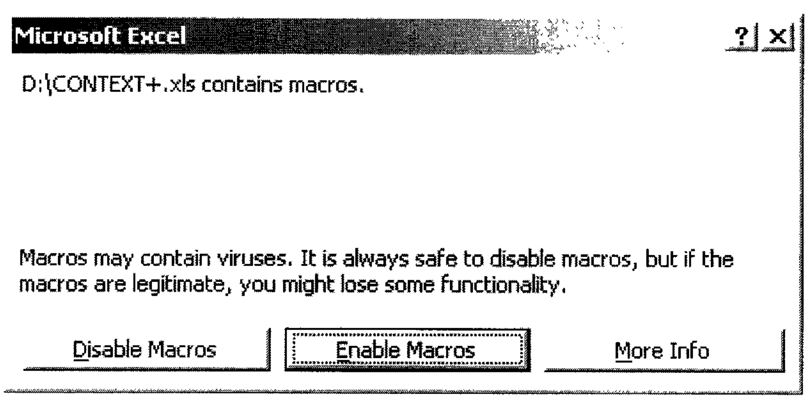


Figure 3-3: MS Excel Standard Open File Dialog Box

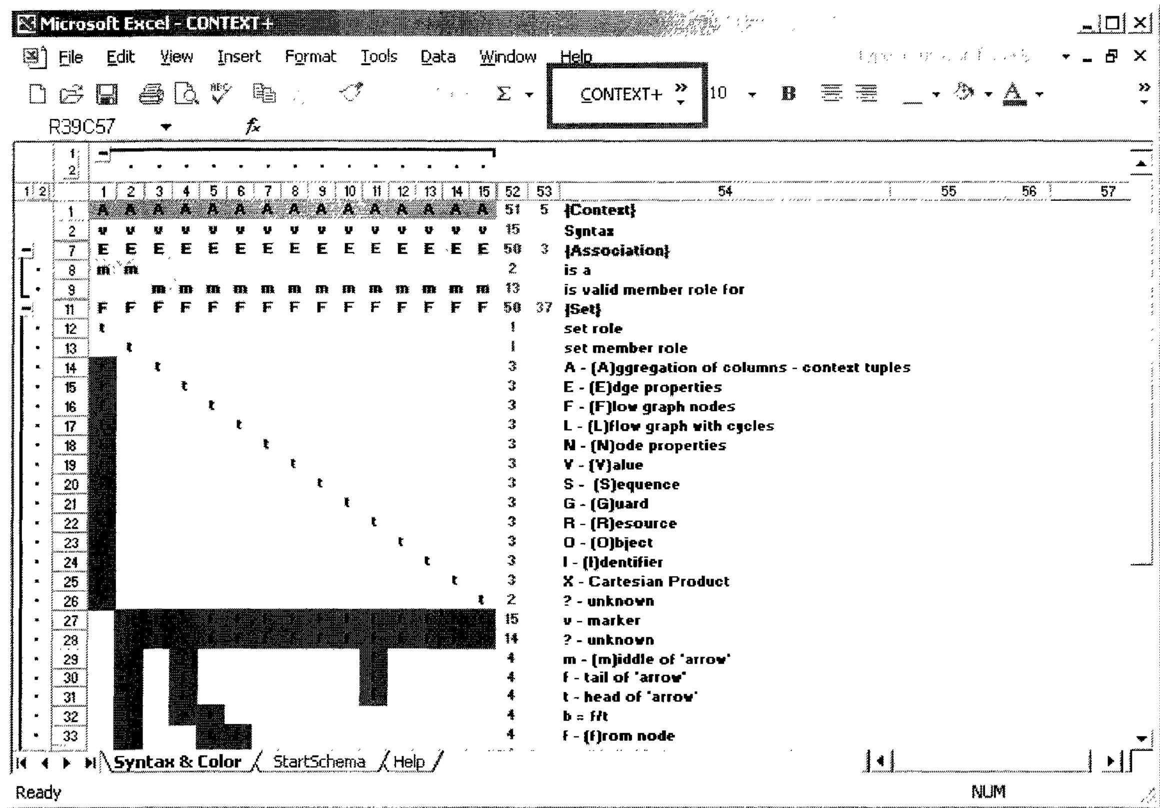


Figure 3-4: CONTEXT+ Macro

To make the CONTEXT+ macro run whenever, you need to click on the CONTEXT+ button on the toolbar. There are four main operation categories in CONTEXT+ as you can see from Figure 3-5: **Mode**, **Query**, **Output** and **Run**. The **Mode** operation is used as input criteria for **Query** operation. The **Output** section is the output format for the result of the **Query** operation while the **Run** section has only one function so far.

The **Mode** operation category has the following functions:

- **Visible Maps**: This function needs to work with the query operation. If the "Visible Maps" is selected, the result map will only display the visible parts of the current map. Otherwise, the result map will display all the hidden rows and columns.

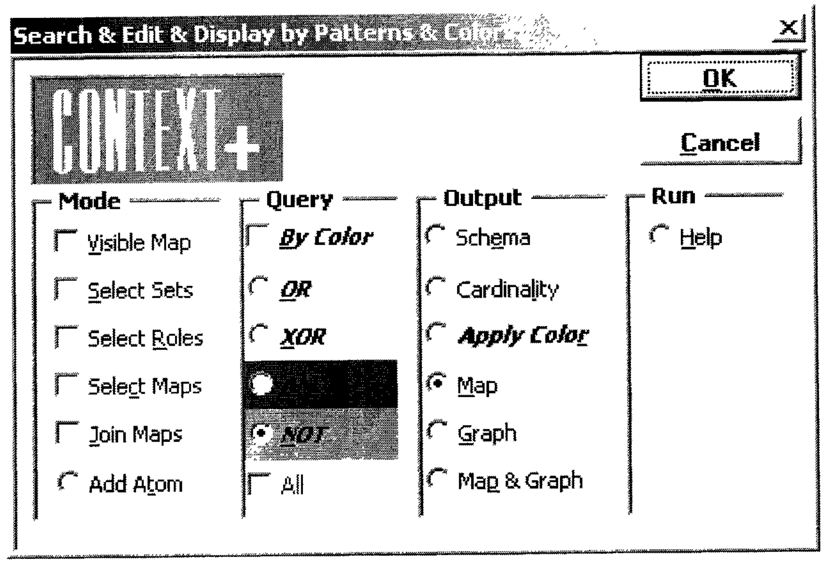


Figure 3-5: CONTEXT+ Operation Dialog

- **Select Sets**: This function needs to work with the query operation. When user selects "Select Sets", it will automatically list all the set values in the current maps. Thus, users can select the set values when doing query operations.
- **Select Roles**: This function needs to work with the query operation. When user selects "Select Roles", it will automatically list all the roles values in the current maps. Thus, users can select the roles values when doing query operations.
- **Select Maps**: This function needs to work with "By Color" function. It will list all the available worksheets (maps) for the current workbook and enable user to select all or some worksheets to perform "By Color" operation simultaneously.

- **Join Maps**: This function lists all the available worksheets (maps) for the current workbook and allow the user to merge the selected maps. The criteria of merging are to compare the values of concept columns within each selected map. Then the operation joins the columns with the same values of the concepts or adds addition rows with different values of the concepts. The result map will be put into a separate worksheet named "Merge Result".
- **Add Atom**: This function enables user to insert rows to the spreadsheets of the active workbook.

The **Query** operation category has the following functions:

- **AND**: Implementing conjunction operation based on the values of the columns related to the selected 2 – n cells in the current map. The limitation of this function is that the selected cells must have values.
- **XOR**: Implementing exclusion operation based on the values of the columns related to the selected 2 – n cells in the current map. The limitation of this function is that the selected cells must have values.
- **OR**: Implementing disjunction operation based on the values of the columns related to the selected 1 – n cells in the current map. The limitation of this function is that the selected cells must have values.
- **NOT**: Implementing negation operation based on the values of the columns related to the selected one cell in the current map. The limitation of this function is that the selected single cell must have a value.

- **By Color**: This function needs to work with one of the four operations, 'AND', 'XOR', 'OR' and 'NOT', based on selected predefined query in the map. In order to run this operation, each map shall have at least one predefined query. The predefined query is defined as cells with Red, Yellow, Green and Blue as background color in one column. In a predefined query, the Red means "AND" operation, Yellow means "XOR" operation, "Green" means "OR" operation and "Blue" means "NOT" operation. This function can also work with standalone "AND", "XOR", "OR" and "NOT" operations. However, it must select at least two predefined queries.
- **All**: One of OR, XOR, AND, NOT operations is performed on the related columns without selecting any cells.

The **Output** operation category has the following functions:

- **Show Schema**: This function produces a Zoom-In map, which is a work frame of a context map.
- **Cardinality**: This function computes the Cardinality of Roles and the Cardinality of Set Members.
- **Apply Color**: This function applies different background colors to the Set Roles, Set Members Roles, Cardinality of Roles, and Cardinality of Set Members based on the element's usage and value in the context maps (see Figure 3-6 and Example 3-1).
- **Map**: This function presents query result in a map format.

- **Graph (Being implemented)**: This function presents query result in a graph format.
- **Map & Graph (Being implemented)**: This function presents query result in both map and graph formats.

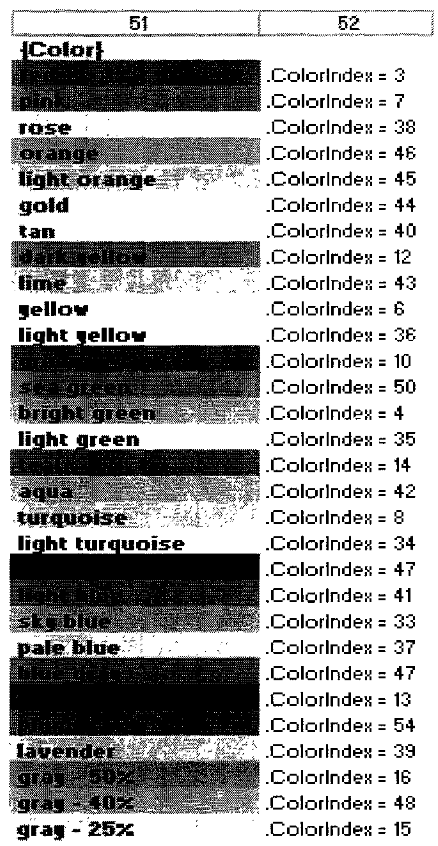


Figure 3-6: Color Index

Example 3-1: Notation and Color Mappings of Context Maps

"A, I, o" --- Dark Gray

"v, u" --- Light Gray

"V" --- Bright Green

“X, x, F” --- Light Purple

“E” --- Lime

“b” --- Violet

“O” --- Object

“R, m” --- Aqua

“c” --- Green

“d” --- Plum

“Y” --- Light Turquoise

“f” --- Red

“L” --- Pink

“G” --- Dark Green

“N” --- Rose

“S” --- Yellow

“t” --- Pale Blue

The **Run** operation category has one function:

- **Help**: This function displays a short version of “Help” file (Figure 3-7) in an Excel spreadsheet. It has a brief description for each function as shown in Figure 3-5.

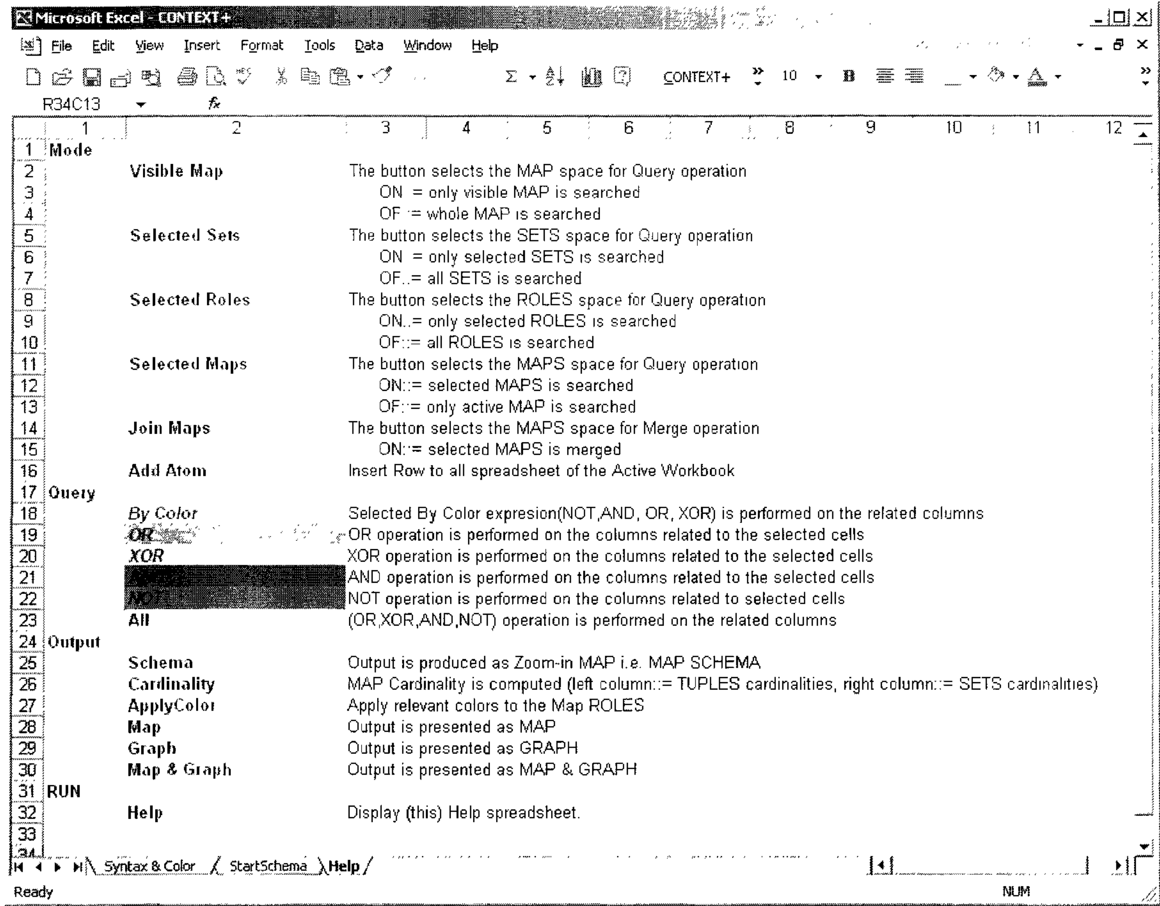


Figure 3-7: CONTEXT+ Help Sheet

3.4 Constraints and Limitations

The context maps are not only a methodology and formalized notation representing the knowledge, but also a powerful tool used directly to manipulate the formatted knowledge. However, there are many constraints which limit the opportunities for further development of the context maps and related tools or applications. Among these are the dependency on the use of MS Excel, lack of an integrated, highly intuitive and user-friendly interface for manipulating the context maps, and lack of on-line help for using the application and understanding of the technology and notations of the context maps.

Due to the constraints on the use of MS Excel for the development of the context maps, any programs developed for the context maps must comply with the technical requirements imposed by MS Excel. This limits the wide use of the context maps, especially for advanced manipulation of the context maps. If the ultimate goal for the context maps development is to create and manipulate the formatted knowledge on the web, the Excel based context maps will only make this task more difficult.

Lack of an integrated, highly intuitive and user-friendly interface and on-line help, are though, two important limitations for the use of CONTEXT+. Users of CONTEXT+ are expected to be experts with a good working knowledge of context maps. However, all applications need to be made easier to grasp by the novice and faster to use by more experienced users. It is not always an easy task for a novice to use CONTEXT+ when there is only one page of on-line help file provided.

Thus, given the history of the development of the context maps and the fact that there are some constraints and limitations on the current format of the context maps, a new way to represent the context maps and a new tool to create and manipulate them with enhanced functionalities are necessary.

Chapter 4 : XML-Based Context Maps

4.1 Introduction to XML

This section will introduce the basics and some major features of XML. The purpose is to provide information of XML and give an idea on how XML can be used as information storage and interchange for the context maps. The details of the XML syntax and its technologies are beyond the scope of this report. For further information on XML and related technologies please refer to the World Wide Web Consortium (W3C) website at http://www.w3c.org/XML.

4.1.1 What Is XML?

In recent years, XML is fast becoming the standard for data interchange on the Web. XML stands for eXtensible Markup Language and it is a text-based markup language much like HTML. But XML is not a replacement for HTML which was designed to display data and to focus on how data looks. While XML was designed to describe data and to focus on what data is.

As with HTML, XML identify data using tags (identifiers enclosed in angle brackets, like this: <...>). But XML tags are not predefined. Users must define their own tags. XML tags are used to identify the data, rather than specifying how to display it. It is just like a label on a piece of data that identifies it (for example: <name>...</name>). Thus, users are free to use any XML tags that make sense for their own purpose.

Here is an example of some XML data might be used for a note:

```
<note>
  <to>Dr. W. M. Jaworski</to>
  <from>Qiao Li</from>
  <subject>XML Is Really Useful</subject>
  <text>
    Why don't we store Context Maps as XML? Let me do the
job...
  </text>
</note>
```

The tags in this example identify the note as a whole, the destination and sender, the subject, and the text of the note. As in HTML, the <to> tag has a matching end tag: </to>. The data between the tag and its matching end tag defines an element of the XML data. Also the content of the <to> tag is entirely contained within the scope of the <note>..</note> tag. It is this ability for one tag to contain others that gives XML its ability to represent hierarchical data structures.

Unlike HTML, however, in XML user could easily search a data set for notes containing "XML" in the subject, because the XML tags identify the content of the data, rather than specifying its representation.

XML tags can also contain attributes -- additional information included as part of the tag itself, within the tag's angle brackets. The following example shows a note structure that uses attributes for the "to", "from", and "subject" fields:

```
<note to="Dr. W. M. Jaworski" from="Qiao Li" subject="XML Is
Really Useful">
  <text>
    Why don't we store Context Maps as XML? Let me do the
job...
```

```
  </text>
</note>
```

As in HTML, the attribute name is followed by an equal sign and the attribute value, and multiple attributes are separated by spaces. Unlike HTML, however, in XML commas between attributes are not ignored -- if present, they generate an error.

Since users could design a data structure like <note> equally well using either attributes or tags, it can take a considerable amount of thought to figure out which design is best for their purposes.

4.1.2 Why Is XML So Useful?

Given the history of XML development, it has been amazing to see how quickly the XML standard has been developed and how quickly a large number of software vendors have adopted the standard. XML has been enormously successful as a markup language for documents and data. It enjoys wide support and deployment in a variety of programming languages, computing platforms, applications and specifications. There are a number of reasons for XML's surging success. Here are a few of the most prominent.

- **XML is using plain text**

  Since XML is not a binary format, user can create and edit files with anything from a standard text editor to a visual development environment. That makes it easy for users to debug their programs, and makes it useful for storing data.

- **XML can be used to share data**

Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

- **XML can used to exchange data**

  In the real world, computer systems and databases contain data in incompatible formats. However, with XML, data can be exchanged between incompatible systems. Converting the data to XML can greatly reduce the complexity of exchanging data between different systems and create data that can be read by many different types of applications.

- **XML can be used to store data**

  XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

- **XML is flexible and extensible**

  XML allows users to effectively creating new tag sets that can be used for data and documents, and so that multiple applications can display or process the content of these novel tags. As such, XML provides flexibility and extensibility by not providing a standard tag set like HTML.

- **Easy to process and parse**

  XML's regular and consistent notation makes it easier to build a program to process XML data because an XML document is well-formed. In addition, since XML is a vendor-neutral standard, user can choose among several XML parsers, any one of which takes the work out of processing XML data. Users don't have to create custom parsers with hard coded structural knowledge.

4.1.3 Document Type Definition (DTD)

A DTD is a little like a global glossary that outlines what kinds of XML tags a document can contain and where they are placed within it. The purpose of a DTD is to provide the rules and framework for the content and organization of the elements, attributes, and entities contained within the document. It describes what tags are valid, what attributes of the tags are valid, and the relationships between tags. Then parsers can use the DTD to validate an XML document. While DTDs are not a required component of valid XML documents, the reusable nature of XML tags makes them highly desirable. DTDs can easily be shared by many different XML documents or web sites.

4.2 XML-Based Context Maps

4.2.1 The Design of DTD for the XML-Based Context Maps

Defining a DTD is the first step to make the context map as an XML-based document. It is a very important decision stage in making XML design for the context maps.

First, the various entities (objects) that make up a context map need to be identified. From pervious chapters, we know each set (concept) and set member in a context map can be regarded as an Atom. Each atom has its own attributes such as the index of this atom in the context map, atom name, atom type and so on. Once the objects in the context map regarded as the atoms are identified, the next is to formulate the relationships among these atoms. From the presentation point of view, the links among set roles and set member roles can be decided by the same relationship index (column index). Following these links, we can further identify the relationships among these atoms. As to what kind of relationship exists between two atoms, it can be interpreted from the relationship label according to the notations of the context maps.

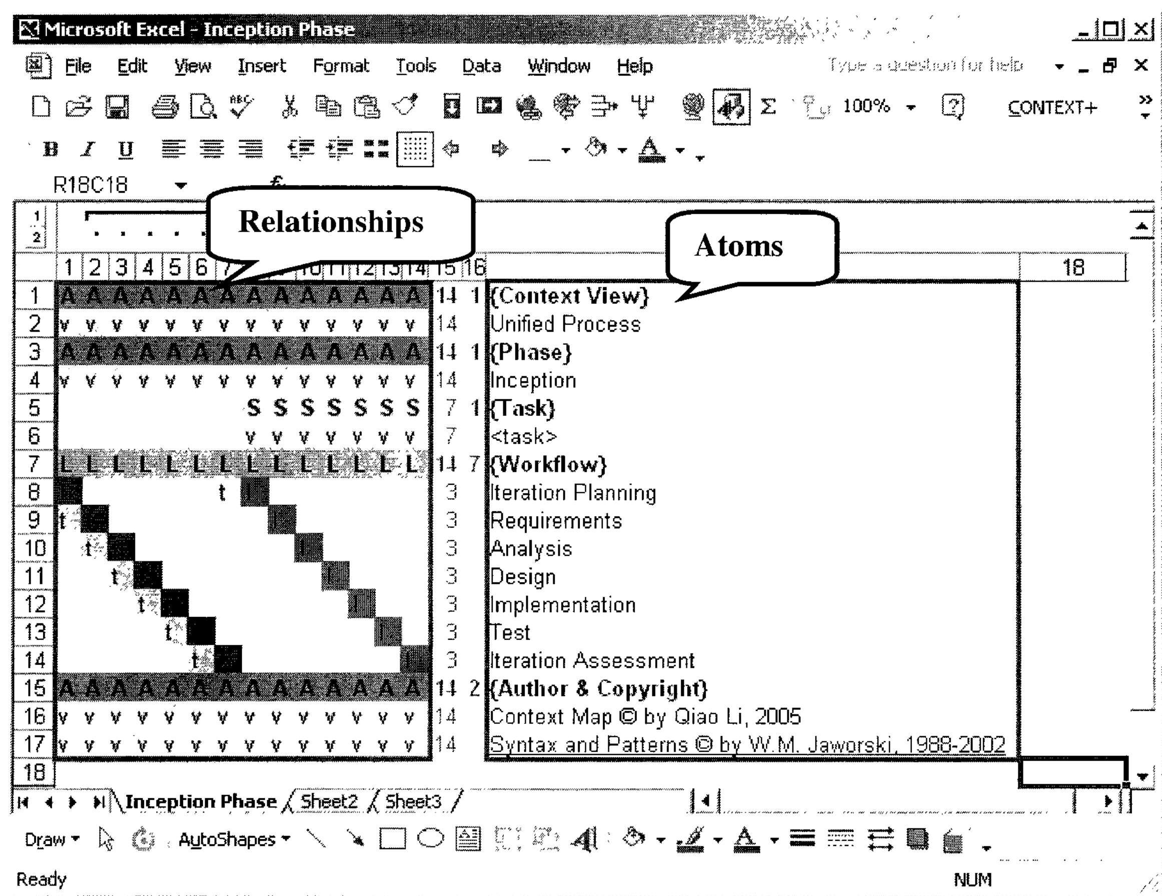


Figure 4-1: The Illustration of a Context Map Analysis

The below DTD was defined for the context map XML document based on the identified objects, their attributes and their relationships with other objects within a context map as illustrated in Figure 4-1.

Example 4-1: ContextMap.DTD

```
<!-- ................................................ -->
<!-- Context Maps Specification DTD .................. -->
<!-- Author: Qiao Li ............ ................... -->
<!-- Date: June 26, 2004 ........ ................... -->
<!-- ................................................ -->

<!ELEMENT ContextMap (
                CMapDoc,
                ContextMapType
               )>
<!ELEMENT CMapDoc (
          Location,
          RevisionTable,
          Reference*
          )>

<!ELEMENT Location EMPTY>
<!ATTLIST Location
          locatedAt  CDATA  #REQUIRED>

<!ELEMENT RevisionTable (Revisions+)>

<!ELEMENT Revisions EMPTY>
<!ATTLIST Revisions
          version CDATA #REQUIRED
          date    CDATA #REQUIRED
          author  CDATA #REQUIRED
          desc    CDATA #REQUIRED>

<!ELEMENT Reference EMPTY>
<!ATTLIST Reference
          document_name CDATA #REQUIRED
          location CDATA #REQUIRED
          date CDATA #REQUIRED
          author  CDATA #REQUIRED>
```

```
<!ELEMENT ContextMapType (
             Description,
             Atoms+
            )>
<!ATTLIST ContextMapType
          Name ID #REQUIRED
          cmap_type_name  CDATA #REQUIRED
          cmap_atom_numbers CDATA #REQUIRED
          cmap_relationship_numbers CDATA #REQUIRED
          Copyright CDATA #REQUIRED>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT Atoms (
             Relationships,
             Description
            )>
<!ATTLIST Atoms
          atom_index CDATA #REQUIRED
          atom_name CDATA #REQUIRED
          is_a_schema  CDATA #REQUIRED
          atom_type  CDATA #REQUIRED
          location  CDATA #REQUIRED
          roles_cardinality CDATA #REQUIRED
          set_members_cardinality CDATA #REQUIRED>

<!ELEMENT Relationships (Relationship*)>
<!ATTLIST Relationships
          atom_index CDATA #REQUIRED>

<!ELEMENT Relationship EMPTY>
<!ATTLIST Relationship
          atom_index CDATA #REQUIRED
          index  CDATA #REQUIRED
          label  CDATA #REQUIRED>
```

4.2.2 Description of the XML-Based Context Maps

The definition of each element and each attribute defined in the DTD will be discussed in this section following an XML template of the context map document. The descriptions of each element or attribute will make the user easily understand the Context Maps Development process under the new XML approach.

Example 4-2: CMapTemplate.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ContextMap SYSTEM "ContextMap.dtd">

<ContextMap>
  <CMapDoc>
    <Location locatedAt="http://localhost/docs/"/>
    <RevisionTable>
      <Revisions
          version="Initial"
          date="June 26, 2004"
          author="Qiao Li"
          desc="XML Template of Contxet Map "/>
    </RevisionTable>
    <Reference
        document_name=""
        location=""
        date=""
        author=""/>
  </CMapDoc>
  <ContextMapType
      Name="Template"
      cmap_type_name=""
      cmap_atom_numbers=""
      cmap_relationship_numbers=""
      Copyright="">
    <Description>
      Describe this Context Map in detail ...
    </Description>
    <Atoms
        atom_index=""
        atom_name=""
        is_a_schema=""
        atom_type=""
        location=""
        roles_cardinality=""
        set_members_cardinality="">
      <Relationships
       atom_index="">
        <Relationship
         atom_index=""
         index=""
         label="">
        </Relationship>
      </Relationships>
    </Atoms>
```

```
  </ContextMapType>
</ContextMap>
```

CMapDoc: This defines information about the context map that this document defines. It defines a specific version and related information of this context map. Its parameters are:

- Location: The location of this document. This would be very important when the context map is used by a web application.
- RevisionTable: The revision history of this context map.
    - Revisions: One particular version of this context map.
        - version: the version, of this context map, that this document defines.
        - date: the date, of this context map, that this document was created or modified.
        - author: the author of this context map.
        - desc: a brief description of this version.
- Reference: This defines reference information used for this document.
    - document_name: the reference document name.
    - location: the location of reference document.
    - date: the date of this reference document was published.
    - author: the author of this reference document.

ContextMapType: This is the starting point of the context map definitions. It is discussed in detail below.

- ContextMapType is the starting point of XML definition for a context map. Each context map has a unique context map type that this type captures all the information about the context map. Each context map MUST have one and only one ContextMapType definition associated with it. Its parameters are:

  - Name: this is the name of this context map.
  - cmap_type_name: a unique name given for this context map. This is what is used, internally by the various applications, to identify this context map. It is also used as the string that will be used to get the context map title to be displayed on the application windows. Currently, it is same as Name attribute.
  - cmap_atom_numbers: the number of atoms which make up this context map.
  - cmap_relationship_numbers: the number of relationships (columns) which make up this context map.
  - Copyright: copyright information about this context map.

- Description: describes this context map type in detail.
- Atoms: Atoms specifies the characteristics of each atom in a context map. Each ContextMapType MUST have one, but can have more, Atoms definition associated with it. Its parameters are:
  - atom_index: a unique index for each atom within a context map.
  - atom_name: a name for each atom.
  - is_a_schema: whether this atom will be used as a schema atom for this context map. Valid values are Yes (1) and No (0).
  - atom_type: defines the type of this atom. Valid values are, in the order of increasing restriction, Concept (0), Text (1), Image (2), Link (3), Email (4) or Map (5). Concept belongs to "Set" category, Text, Image, Link, and Email

belong to “Set Member” category, and Map belongs to “Context Map” category, respectively.

- location: if the atom_type is one of Image, Link, Email or Map atom types, the hyperlink, email address or location need to be provided, respectively.
- roles_cardinality: the cardinality of the roles that this atom has with other associated atoms.
- set_members_cardinality: the cardinality of the set members that this atom has with other associated atoms.

- Relationships: A container for all the relationship definitions associated with one atom. Each atom can have zero or one Relationship definition associated with it.
  - atom_index: a pointer back to the index of the atom to which these relationships belong.
  - Relationship: defines a detail relationship associated with this atom.
    - atom_index: a pointer back to the index of the atom to which this relationship belongs.
    - index: a unique index for each relationship within a Relationships definition.
    - label: the string to be used to identify this relationship, null value means there is no relationship.

Chapter 5 : Transformation to XML-Based Context Map

5.1 Background

5.1.1 Why Is Transformation Needed?

Given the history of the development of the context maps and the fact that currently all the existing context maps were created by using MS Excel. A tool to transform all existing Excel-based context maps to the XML-based context maps is a necessity. The CMapConverter tool will provide this functionality by using XSLT (the Extensible Stylesheet Language Transformations) technology.

5.1.2 What Are XSL and XSLT?

The eXtensible Stylesheet Language (XSL) was developed by World Wide Web Consortium (W3C) because there was a need for an XML-based Stylesheet Language. XSL is used to describe how the document should be displayed in addition to the XML document itself.

XSL consists of three parts: XSLT is a language for transforming XML documents, XPath (the XML Path Language) is a language for defining parts of an XML document and XSL-FO (the Extensible Stylesheet Language Formatting Objects) is a language for formatting XML documents.

XSLT is the most important part of the XSL Standards. It can be used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element. XSLT can also add new elements into the output file, or remove elements. It can rearrange and sort elements, and test and make decisions about which elements to display, and a lot more. A common way to describe the transformation process is to say that XSLT transforms an XML source tree into an XML result tree.

In this report, we will use XSLT to transform context maps from the MS Excel based format to the XML-based format. The details of the XSLT syntax and its coding techniques are beyond the scope of this report. For further information on XSLT language please refer to http://www.w3.org/TR/xslt web site. The XSLT processor can be found on http://xml.apache.org/xalan-j/index.html web site.

5.2 CMapConverter

5.2.1 Introduction

In order to develop the CMapConverter tool, it is necessary to understand how MS Excel works and what the current development trend of MS Excel is. The new and the most important feature in MS Excel 2003 version is to support XML. Starting from Excel 2003, the spreadsheets can be saved in a native XML file format which can be manipulated and searched using any program that can process industry standard XML.

The CMapConverter takes advantage of a 2003 Excel saved XML context map and then directly transforms it to an XML-based context map using XSLT technology. In order to better understand the design idea of the CMapConverter, an example used in following sections will be used to illustrate it.

5.2.2 Excel saved XML Context Map

Figure 5-1 is a schema view of the context map (Figure 3-2) as shown in Chapter 3 for the inception phase in an Excel spreadsheet format. Example 5-1 is a part of the saved XML file of Figure 5-1 which shows the XML format for one displayed atom Context View and one hidden atom Iteration Assessment.

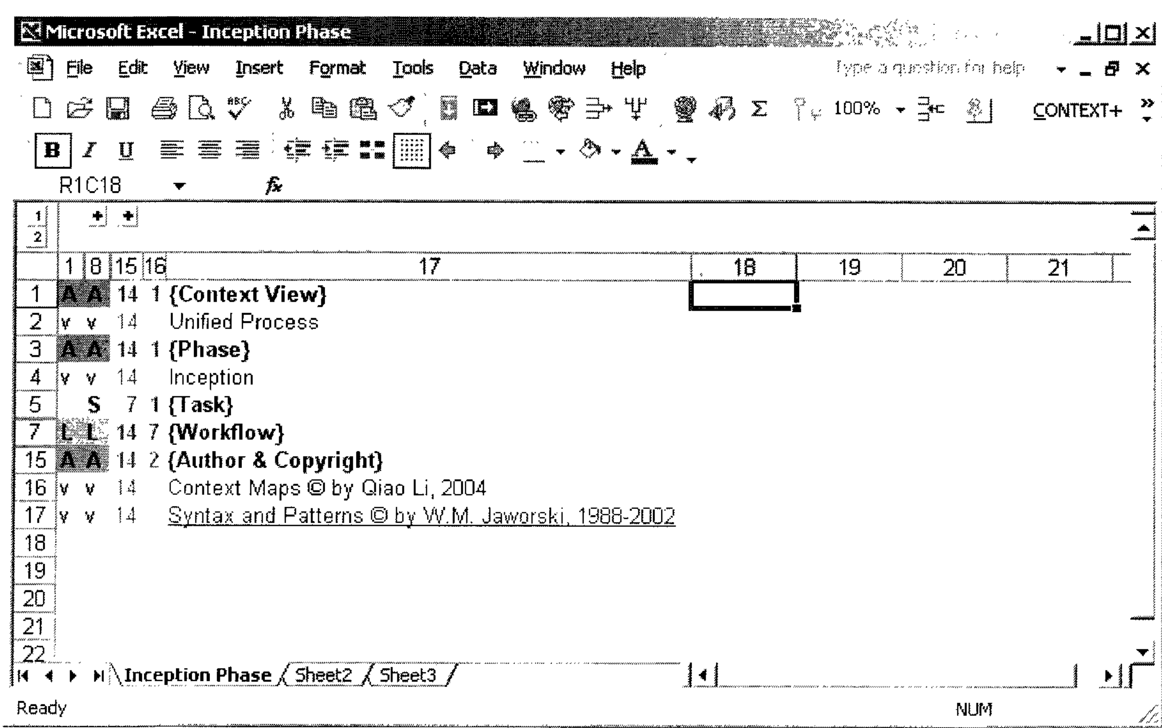


Figure 5-1: Schema View of Inception Phase Context Map in Excel Spreadsheet

Example 5-1: A Part of Inception.xml

```
<Worksheet ss:Name="Inception Phase">
  <Table ss:ExpandedColumnCount="17"
ss:ExpandedRowCount="17" x:FullColumns="1"
   x:FullRows="1">
   <Column ss:AutoFitWidth="0" ss:Width="12"/>
   <Column ss:Hidden="1" ss:AutoFitWidth="0" ss:Width="12"
ss:Span="5"/>
   <Column ss:Index="8" ss:AutoFitWidth="0" ss:Width="12"/>
   <Column ss:Hidden="1" ss:AutoFitWidth="0" ss:Width="12"
ss:Span="5"/>
   <Column ss:Index="15" ss:Width="15.75"/>
   <Column ss:Width="10.5"/>
   <Column ss:AutoFitWidth="0" ss:Width="240"/>
   <Row ss:StyleID="s23">
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s25"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s26"><Data
ss:Type="Number">14</Data></Cell>
```

```
    <Cell ss:StyleID="s26"><Data
ss:Type="Number">1</Data></Cell>
    <Cell><Data ss:Type="String">{Context
View}</Data></Cell>
   </Row>

    .........................
    .........................

   <Row ss:Hidden="1">
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s33"><Data
ss:Type="String">t</Data></Cell>
    <Cell ss:StyleID="s32"><Data
ss:Type="String">f</Data></Cell>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s29"/>
    <Cell ss:StyleID="s34"><Data
ss:Type="String">1</Data></Cell>
    <Cell ss:StyleID="s28"><Data
ss:Type="Number">3</Data></Cell>
    <Cell ss:StyleID="s28"/>
    <Cell><Data ss:Type="String">Iteration
Assessment</Data></Cell>
   </Row>
```

5.2.3 Design Idea for the CMapConverter

As we saw in Example 5-1, Microsoft defined an XML spreadsheet format which is designed specifically for Excel worksheets. However, the CMapConverter can make use of this information for our own transformation purpose. Here are some examples:

- The name of the worksheet can be regarded as the name of the context map.

  ```
  <Worksheet ss:Name="Inception Phase">
  ```

- The values of cmap_atom_numbers and cmap_relationship_numbers can be calculated from the row numbers and column numbers in the worksheet table, respectively.

```
<Table ss:ExpandedColumnCount="17"
ss:ExpandedRowCount="17" x:FullColumns="1"
x:FullRows="1">
```

- The information for each row and each cell can be used to describe the atoms and relationships. For example, the value of `ss:Hidden` in each row can be used to decide whether this atom is a part of schema view and the data string value of each cell can be regarded as the relationship label.

Example 5-2 shows the code on how CMapConverter transforms an Excel saved XML context map to an XML-based context map based on above design idea. Example 5-3 shows a part of the result of transformed context map from Example 5-1. For a complete result of this XML-based context map file please see Appendix A: An XML Example.

Example 5-2: CMapConverter.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--**********************************************************
This is a part of the CMapViewer software
Copyright (C) 2005 Context Map Group
Author: Qiao Li
**********************************************************-->

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  xmlns:java="http://xml.apache.org/xslt/java"
  xmlns:redirect="xalan://org.apache.xalan.lib.Redirect"
```

```
  xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
  extension-element-prefixes="redirect ss"
  exclude-result-prefixes="java ss">
<xsl:output
  method="xml"
  indent="yes"
  encoding="ISO-8859-1"
  doctype-system="ContextMap.dtd" />

<!-- *********** Global variables. ********************* -->

<xsl:variable name="CMapName"
              select="//ss:Worksheet/@ss:Name" />
<xsl:variable name="CMapRows"

select="//ss:Worksheet/ss:Table/@ss:ExpandedRowCount" />
<xsl:variable name="CMapColumns"

select="//ss:Worksheet/ss:Table/@ss:ExpandedColumnCount" />
<xsl:variable name="dateFormatter"
select="java:java.text.SimpleDateFormat.new('MM/dd/yy')" />
<xsl:variable name="today"
select="java:java.util.Date.new()" />
<xsl:variable name="version"
              select="system-property('cmap.version')" />
<xsl:variable name="author"
              select="system-property('camp.author')" />

<!-- ************** Main template.  ******************** -->

<xsl:template match="/">
<xsl:element name="ContextMap">

<!-- *************** CMapDoc   ************************* -->

<xsl:element name="CMapDoc">
  <xsl:element name="Location">
    <xsl:attribute name="locatedAt">
      <xsl:value-of select="'http://localhost/docs/'" />
    </xsl:attribute>
  </xsl:element>
  <xsl:element name="RevisionTable">
    <xsl:element name="Revisions">
      <xsl:attribute name="version">
        <xsl:value-of select="$version" />
      </xsl:attribute>
      <xsl:attribute name="date">
```

```
        <xsl:value-of select="java:format($dateFormatter,
$today)" />
      </xsl:attribute>
      <xsl:attribute name="author">
        <xsl:value-of select="$author" />
      </xsl:attribute>
      <xsl:attribute name="desc">
        <xsl:value-of select="'Template'" />
      </xsl:attribute>
    </xsl:element>
  </xsl:element>
  <xsl:element name="Reference">
    <xsl:attribute name="document_name">
      <xsl:value-of select="' '" />
    </xsl:attribute>
    <xsl:attribute name="location">
      <xsl:value-of select="' '" />
    </xsl:attribute>
    <xsl:attribute name="date">
      <xsl:value-of select="' '" />
    </xsl:attribute>
    <xsl:attribute name="author">
      <xsl:value-of select="' '" />
    </xsl:attribute>
  </xsl:element>
</xsl:element>

<!-- **************** ContextMapType ******************* -->

<xsl:element name="ContextMapType">
  <xsl:attribute name="Name">
    <xsl:value-of select="'Template'" />
  </xsl:attribute>
  <xsl:attribute name="cmap_type_name">
    <xsl:value-of select="$CMapName" />
  </xsl:attribute>
  <xsl:attribute name="cmap_atom_numbers">
    <xsl:value-of select="$CMapRows" />
  </xsl:attribute>
  <xsl:attribute name="cmap_relationship_numbers">
    <xsl:value-of select="($CMapColumns)-3" />
  </xsl:attribute>
  <xsl:attribute name="Copyright">
    <xsl:value-of select="' '" />
  </xsl:attribute>
  <xsl:element name="Description">
    <xsl:value-of select="' '" />
```

```
  </xsl:element>
  <xsl:call-template name="CMapAtoms" />
</xsl:element>
</xsl:element>
</xsl:template>

<!-- ***************** Atoms.  *********************** -->

<xsl:template name="CMapAtoms">
<xsl:for-each select="//ss:Table/ss:Row">
  <xsl:variable name="rowIndexNum">
    <xsl:value-of select="position()" />
  </xsl:variable>
  <xsl:element name="Atoms">
    <xsl:attribute name="atom_index">
      <xsl:value-of select="$rowIndexNum" />
    </xsl:attribute>
    <xsl:attribute name="atom_name">
      <xsl:value-of
select="ss:Cell[position()=$CMapColumns]/ss:Data" />
    </xsl:attribute>
    <xsl:attribute name="is_a_schema">
      <xsl:choose>
        <xsl:when test="@ss:Hidden='1'">
          <xsl:value-of select="'0'" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="'1'" />
        </xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
    <xsl:variable name="atom_ref">
      <xsl:value-of
select="ss:Cell[position()=$CMapColumns]/@ss:HRef" />
    </xsl:variable>
    <xsl:attribute name="atom_type">
      <xsl:choose>
        <xsl:when test="string-length($atom_ref) = 0">
          <xsl:choose>
            <xsl:when test="starts-
with(ss:Cell[position()=$CMapColumns]/ss:Data, '{')">
              <xsl:value-of select="'0'" />
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="'1'" />
            </xsl:otherwise>
          </xsl:choose>
```

```
        </xsl:when>
        <xsl:otherwise>
          <xsl:choose>
            <xsl:when test="starts-with($atom_ref,
'http:')">
              <xsl:value-of select="'3'" />
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="'4'" />
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="location">
      <xsl:value-of select="$atom_ref" />
    </xsl:attribute>
    <xsl:attribute name="roles_cardinality">
      <xsl:value-of
select="ss:Cell[position()=($CMapColumns)-2]/ss:Data" />
    </xsl:attribute>
    <xsl:attribute name="set_members_cardinality">
      <xsl:value-of
select="ss:Cell[position()=($CMapColumns)-1]/ss:Data" />
    </xsl:attribute>

<!-- ***************** Relationships.  ***************** -->

<xsl:element name="Relationships">
  <xsl:attribute name="atom_index">
    <xsl:value-of select="$rowIndexNum" />
  </xsl:attribute>
  <xsl:for-each select="ss:Cell">
    <xsl:variable name="dataValue" select="ss:Data" />
    <xsl:if test="(($CMapColumns)-2)>position()">
      <xsl:element name="Relationship">
        <xsl:attribute name="atom_index">
          <xsl:value-of select="$rowIndexNum" />
        </xsl:attribute>
        <xsl:attribute name="index">
          <xsl:value-of select="position()" />
        </xsl:attribute>
        <xsl:attribute name="label">
          <xsl:value-of select="$dataValue" />
        </xsl:attribute>
      </xsl:element>
    </xsl:if>
```

```
  </xsl:for-each>
</xsl:element>
</xsl:element>
</xsl:for-each>
</xsl:template>

<!-- ************************************************** -->
</xsl:stylesheet>
```

Example 5-3: A part of Inception.cm.xml

```
<ContextMapType Name="Template" cmap_type_name="Inception
Phase" cmap_atom_numbers="17" cmap_relationship_numbers="14"
Copyright=" ">
<Description> </Description>
<Atoms atom_index="1" atom_name="{Context View}"
is_a_schema="1" atom_type="0" location=""
roles_cardinality="14" set_members_cardinality="1">
<Relationships atom_index="1">
<Relationship atom_index="1" index="1" label="A"/>
<Relationship atom_index="1" index="2" label="A"/>
<Relationship atom_index="1" index="3" label="A"/>
<Relationship atom_index="1" index="4" label="A"/>
<Relationship atom_index="1" index="5" label="A"/>
<Relationship atom_index="1" index="6" label="A"/>
<Relationship atom_index="1" index="7" label="A"/>
<Relationship atom_index="1" index="8" label="A"/>
<Relationship atom_index="1" index="9" label="A"/>
<Relationship atom_index="1" index="10" label="A"/>
<Relationship atom_index="1" index="11" label="A"/>
<Relationship atom_index="1" index="12" label="A"/>
<Relationship atom_index="1" index="13" label="A"/>
<Relationship atom_index="1" index="14" label="A"/>
</Relationships>
</Atoms>
    ........................
    ........................

<Atoms atom_index="14" atom_name="Iteration Assessment"
is_a_schema="0" atom_type="1" location=""
roles_cardinality="3" set_members_cardinality="">
<Relationships atom_index="14">
<Relationship atom_index="14" index="1" label=""/>
<Relationship atom_index="14" index="2" label=""/>
<Relationship atom_index="14" index="3" label=""/>
```

```
<Relationship atom_index="14" index="4" label=""/>
<Relationship atom_index="14" index="5" label=""/>
<Relationship atom_index="14" index="6" label="t"/>
<Relationship atom_index="14" index="7" label="f"/>
<Relationship atom_index="14" index="8" label=""/>
<Relationship atom_index="14" index="9" label=""/>
<Relationship atom_index="14" index="10" label=""/>
<Relationship atom_index="14" index="11" label=""/>
<Relationship atom_index="14" index="12" label=""/>
<Relationship atom_index="14" index="13" label=""/>
<Relationship atom_index="14" index="14" label="l"/>
</Relationships>
</Atoms>
```

5.2.4 How to Use CMapConverter

CMapConverter is developed for user to convert the Excel-based context maps into the XML-based context maps which can be opened and manipulated by the CMapViewer application (see Chapter 6).

5.2.4.1 Installation Instructions

To install the CMapConverter, you should follow the following simple steps:

1. Unzip the program (CMapConverter.zip) to a location on your hard drive (e.g., D:\cmap).
2. Update the JAVA_BIN parameter in the cmap_run.bat file. You need a jdk1.3 or above version installed on your PC.

5.2.4.2 How to Run the Program

Only MS Excel 2003 version or above has the function of saving an Excel file as an XML file. We need to convert this XML file to one can be opened by CMapViewer application. In order to make the CMapConverter works, currently there are some limitations on the Excel-based context maps, the steps below need to be followed:

1. Open an Excel-based context map with Excel. The format of the context map has to be the schema format (only schema rows and columns are displayed while other rows and columns are hidden). Why? Because the converter regards the displayed rows, not the hidden rows, as a part of the schema.
2. Create a new blank Excel file. Copy the spreadsheet and paste it into the new file. The copied context map must only contain the context map view from first column to the column which has atom names. No extra column, such as comment column, is needed. Name this spreadsheet same as the name of the context map.
3. Highlight the columns except the atom names column. From the Format menu choose the Cells menu item and make the Horizontal Alignment "Center" and Vertical Alignment "Bottom", the Font has the same font and size for all.
4. Save the file as an XML file.
5. Update the input file and output file names in the xml_run.bat file. The output file shall be ended with .cm.xml because the CMapViewer application distinguishes it as an XML-based context map from other XML files.
6. After all the above steps, we can run the program by double-clicking on the xml_run.bat file. The output XML file will be created under same folder. Finally, this XML file can be opened by the CMapViewer application.

Chapter 6 : CMapViewer Application

6.1 Introduction

Currently the context maps are created and manipulated by using CONTEXT+ tool which was developed based on MS Excel-based context maps. With the introduction of XML-based context maps, a new tool to view and manipulate XML-based context maps is a necessity. The CMapViewer application will provide this functionality. In this Chapter, the requirements and the software architecture of the CMapViewer application will be described and outlined, respectively. In addition, the major features which make this CMapViewer so great for context maps manipulation will be described. For a complete user guide of the CMapView application please refer to Appendix B: User Guide of the CMapViewer Application.

6.2 Requirements

One of the major goals of the CMapViewer application is to provide an easy, intuitive and powerful solution for manipulating the XML-based context maps. The following are the requirements that need to be implemented.

6.2.1 General

R6.2.1.1 The CMapViewer application shall be able to read the context map data from an XML file.

The context map metadata shall be read and loaded into memory when the application opens an XML-based context map file and the metadata can be accessed by various java objects. A mechanism shall validate the metadata format according to the DTD of the context maps.

R6.2.1.2 The CMapViewer application shall provide a mechanism to apply different colors on the context map based on the notations of the context maps.

The mappings between color and notations shall be stored in one data file. The application shall be able to access this file when a context map is opened.

6.2.2 Main Window

R6.2.2.1 The GUI shall conform to common GUI standards based on Windows guidelines.

The design of the GUI shall follow the general guidelines for designing visually and functionally consistent user interfaces for Windows programs.

R6.2.2.2 The main window shall be mainly composed of a Tree View on the left hand side of the GUI and a Table pane on the other side. The two panes are separated by a moveable splitter bar.

The purpose of the Tree View pane is to display a hierarchy of context map atoms and to provide a convenient and intuitive way of navigating among atoms. Different types of atoms shall be indicated in the Tree View.

The purpose of the Table pane is to display the context map view in a table format just like the ones currently used in MS Excel spreadsheet. The context map view shall comply with the syntax and notations of the context maps.

R6.2.2.3 The main application menus shall be composed of the following elements in order: File, Edit, Search, View, Query, Tools and Help.

ToolTips text descriptions of each menu item must be provided. Menu items shall be disabled if the command does not apply to the current atom selection or there is no atoms are selected. Some menu items may not require any selections. When no "require" section is specified for a menu item in the description below, that menu item shall be always enabled. Otherwise, it should be enabled only when the conditions are met. The descriptions of each menu item under the corresponding menu are shown below:

File Menu

Open: will display an Open window which allows user to open an XML-based context map.

Exit: will exit and close CMapViewer application window.

Edit Menu

Select: will mark a selection of one atom (requires selection of one tree node or table row).

Unselect: will unmark an atom selection (requires selection of already marked selection of one atom).

Unselect All: will unmark all atom selection (requires at least one already marked selection of atoms).

Search Menu

Find: will bring up the Find window.

View Menu

Show Toolbar: will show or hide the Toolbar (check mark displayed if Toolbar is displayed).

Show Status Bar: will show or hide the Status Bar (check mark displayed if Status Bar is displayed).

Query Menu

Default: will display the whole context map.

Schema: will display the schema view of the context map.

OR: will perform the OR query on the selected atoms (requires selection of one or more atoms).

XOR: will perform the XOR query on the selected atoms (requires selection of more than one atom).

AND: will perform the AND query on the selected atoms (requires selection of more than one atom).

NOT: will perform the NOT query on the selected atom (requires selection of only one atom).

Tools Menu

Go To: will bring up the corresponding application if the selected atom's type is Link or Email (currently requires selection of one atom and the atom type is Link or Email).

Report: will display query result in IE browser.

Help Menu

Help Topics: will bring up the Help Topics Browser.

About...: will bring up the About window for the application.

R6.2.2.4 The toolbar shall be composed of the following elements in order: Open, Select, Unselect, Default, Schema, OR, XOR, AND, NOT, Find, Go To, Report and Help.

ToolTips text descriptions of each button on the toolbar must be provided. Toolbar button shall be disabled if the command does not apply to the current atom selection or there is no atoms are selected. Some toolbar buttons may not require any selections. When no "require" section is specified for a button in the description below, that toolbar button shall be always enabled. Otherwise, it should be enabled only when the conditions are met. The descriptions of each button are shown below:

Open: will display an Open window which allows user to open an XML-based context map.

Select: will mark a selection of one atom (requires selection of one tree node or table row).

Unselect: will unmark an atom selection (requires selection of already marked selection of one atom).

Default: will display the whole context map.

Schema: will display the schema view of the context map.

OR: will perform the OR query operation on the selected atoms (requires selection of one or more atoms).

XOR: will perform the XOR query operation on the selected atoms (requires selection of more than one atom).

AND: will perform the AND query operation on the selected atoms (requires selection of more than one atom).

NOT: will perform the NOT query operation on the selected atom (requires selection of only one atom).

Find: will bring up the Find window.

Go To: will bring up the corresponding application if the selected atom's type is Link or Email (currently requires selection of one atom and its atom type is Link or Email).

Report: will display query result in IE browser.

Help: will bring up the Help Topics Browser.

R6.2.2.5 The status bar shall be able to display the expanded tool tips when the user is browsing through menu items or toolbar buttons.

6.2.3 Find Window

R6.2.3.1 The find window shall contain one text criteria field which allows the user to find atoms.

R6.2.3.2 The text search criteria shall not be case sensitive.

R6.2.3.3 Once an atom is found, there must be a way to indicate the found atom on the tree view.

6.2.4 Help Topics Browser

R6.2.4.1 The standard Help Topics Browser shall be provided for online documentation.

R6.2.4.2 The topics shall contain both the introduction to the context maps and the user guide of the CMapViewer application.

6.3 Design

The purpose of this section is to outline the software architecture of the CMapViewer application. It will discuss the separation of the CMapViewer's functionality into subsystems. The main responsibilities of each subsystem will also be discussed.

Although basic design is covered here, the details of implementation of each subsystem are captured in the Java source code. The Java technology and its coding techniques are beyond the scope of this report.

6.3.1 High-Level Architecture

The CMapViewer application can be functionally decomposed into the following major subsystems (Figure 6-1):

- Navigation Subsystem brings up the tree view of the context map on the application main window and handles all user actions on the navigation tree.
- Result Table Subsystem displays the query result in the result table and handles all user actions on the result table.
- Search Subsystem brings up the Find window for the user.
- Report Subsystem displays the user query result in a HTML format.
- CMap Window Subsystem serves as the central operation handler and interfaces with the appropriate subsystems based on the operation detected.
- Metadata Handling Subsystem reads and loads the context map metadata into memory when a context map file is opened.

The Main Driver has responsibilities to bring up the main application window and interact with the various user actions.

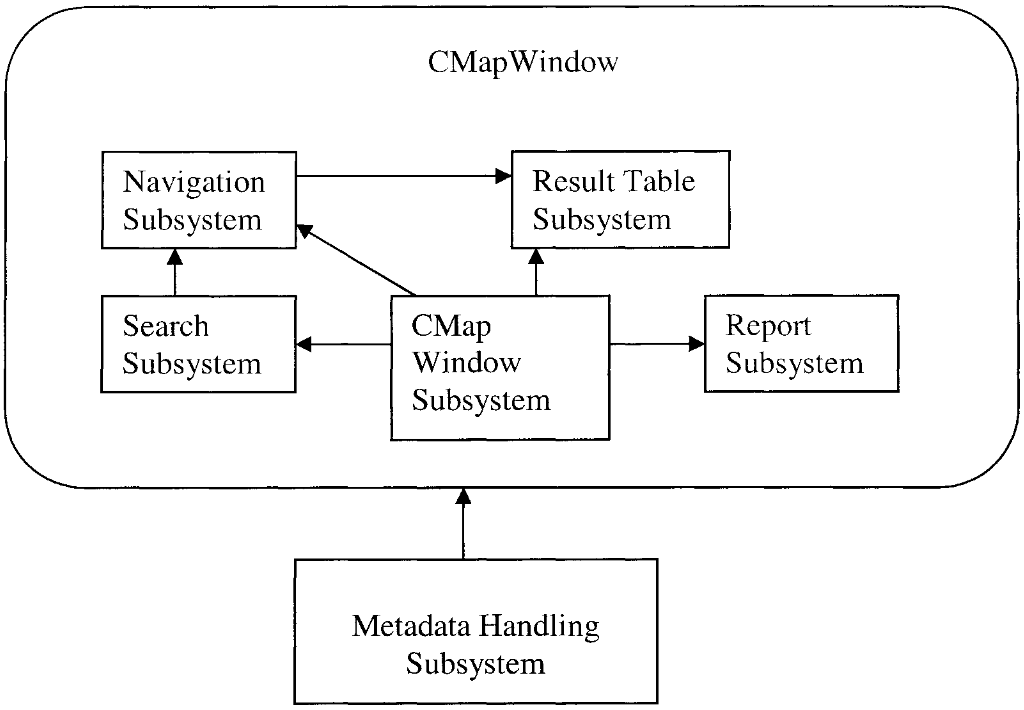


Figure 6-1: High Level Architecture of CMapViewer Application

6.3.2 Main Driver

The main driver is responsible for application initialization and shutdown. The main window of the application looks similar to most windows applications and complies with the common GUI standards. Figure 6-2 shows a snapshot of the main window of the CMapViewer application.

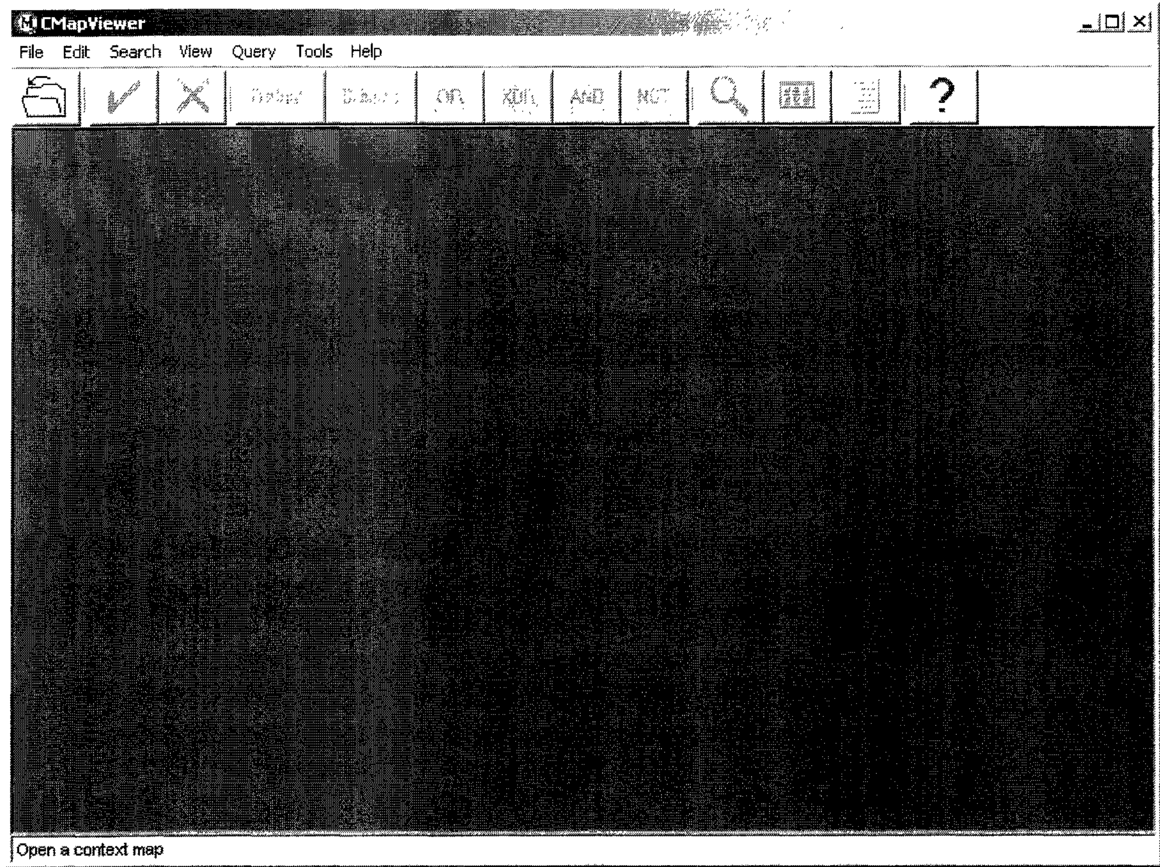


Figure 6-2: The Main Window of CMapViewer Application

6.3.3 Subsystem Architecture

6.3.3.1 Navigation Subsystem

The responsibilities of navigation subsystem are described below. For more details refer to the Message Sequence Charts for the flow of operations in Figure 6-3.

- When user opens an XML-based context map, all the XML metadata will be used to build the navigation tree.
- When user selects a node on the navigation tree, the corresponding row (atom) in the result table will be highlighted.

- When user performs the Select or Unselect operation on a selected tree node, the corresponding atom in the result table will also be marked or unmarked, respectively.
- When user performs the Query operation, the navigation tree will be rebuilt based on the query result of the selected atom(s).
- When user performs the Find operation, the found atom will be highlighted on the navigation tree.
- When user performs the Go To operation, the corresponding application will be brought up based on the selected node (atom) type.

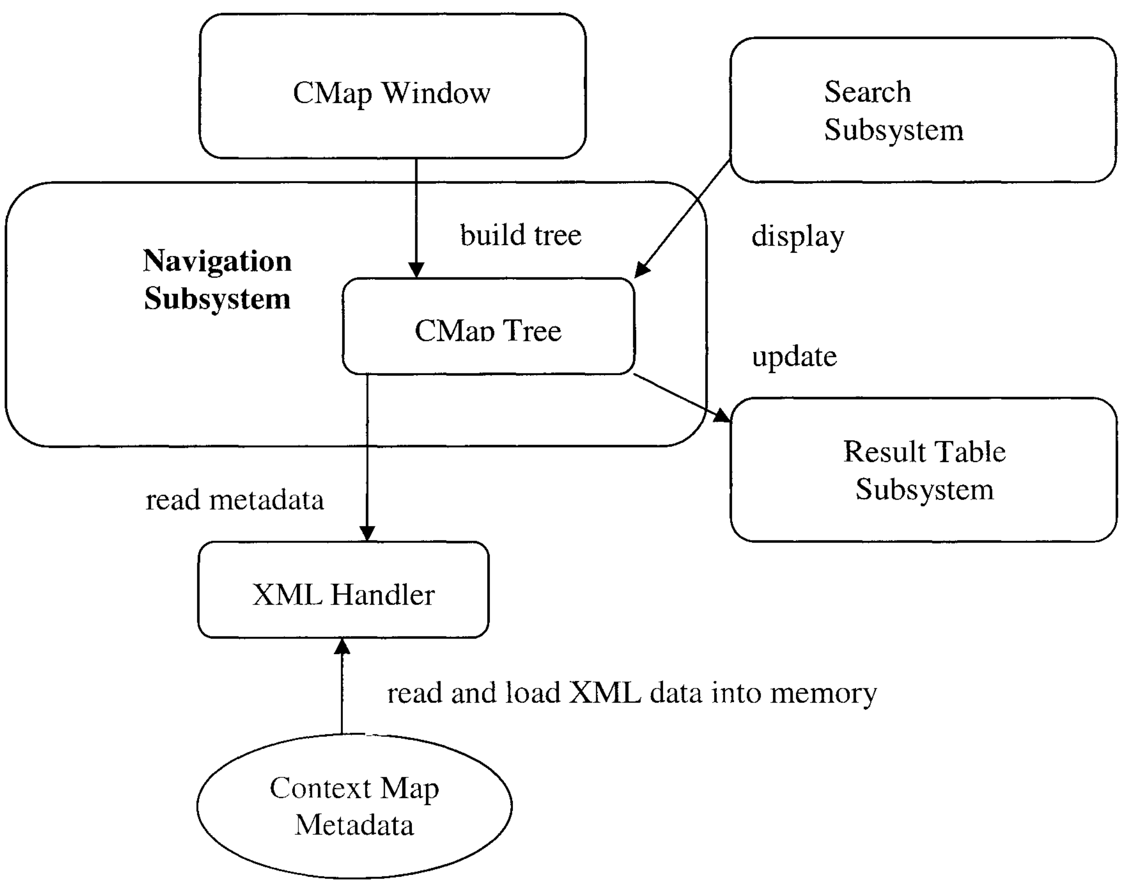


Figure 6-3: Message Sequence Chart for Navigation Subsystem

6.3.3.2 Search Subsystem

The main responsibility of search subsystem is to create the Find window when user performs the search operation on the main window. It then handles the user actions from the Find window. When an atom is found, it informs Navigation Subsystem to indicate the found atom on the navigation tree. Figure 6-4 shows the Message Sequence Charts for the flow of operations for search subsystem.

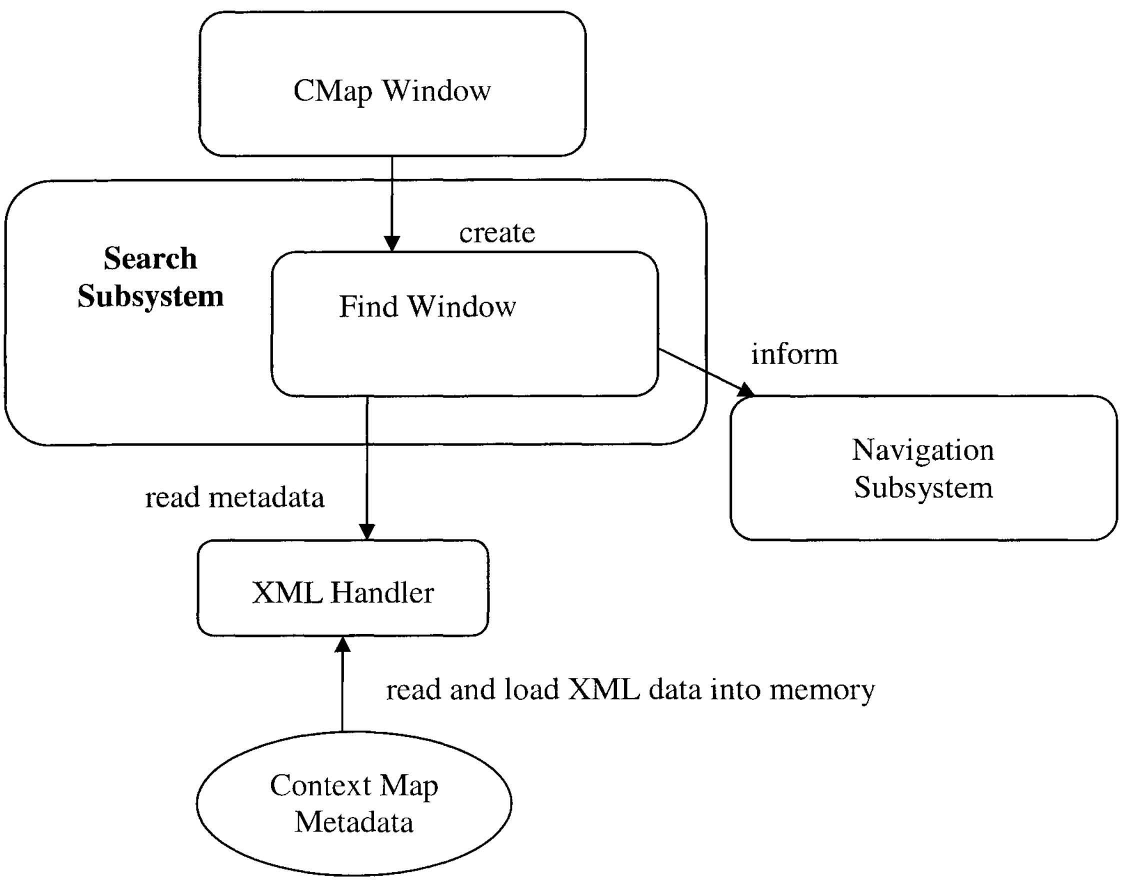


Figure 6-4: Message Sequence Chart for Search Subsystem

6.3.3.3 Result Table Subsystem

The responsibilities of result table subsystem are described below. For more details refer to the Message Sequence Chars for the flow of operations in Figure 6-5.

- When user opens an XML-based context map, all the XML metadata will be used to create a result table to display the whole context map view.
- When user performs the Select or Unselect operation on a selected row in the result table, the selected row (atom) in the table will be marked.
- When user performs the Query operation, the result table will be updated based on the query result of selected atom(s).
- It handles user actions from the result table such as table row selection or table cell navigation.

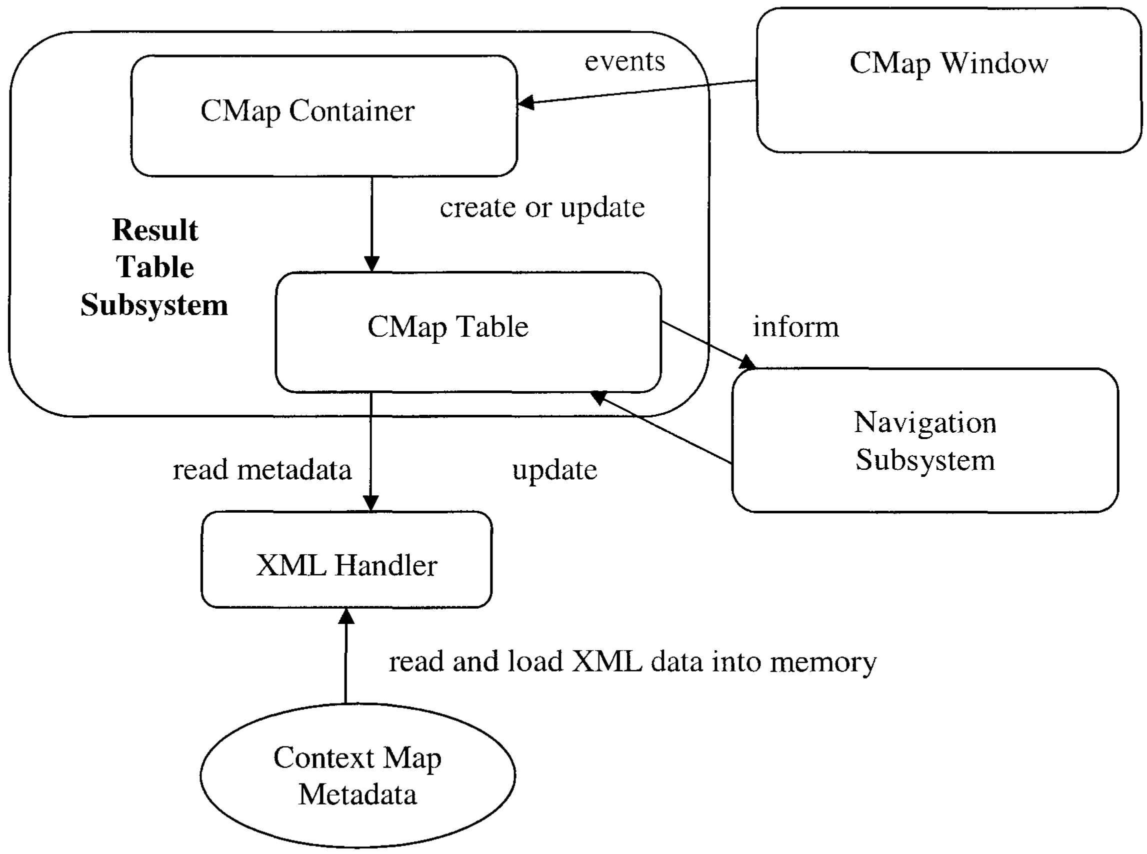


Figure 6-5: Message Sequence Chart for Result Table Subsystem

6.3.3.4 Report Subsystem

The responsibility of report table subsystem is to bring up the IE browser to report the query result in HTML format. For more details refer to the Message Sequence Chars for the flow of operations in Figure 6-6.

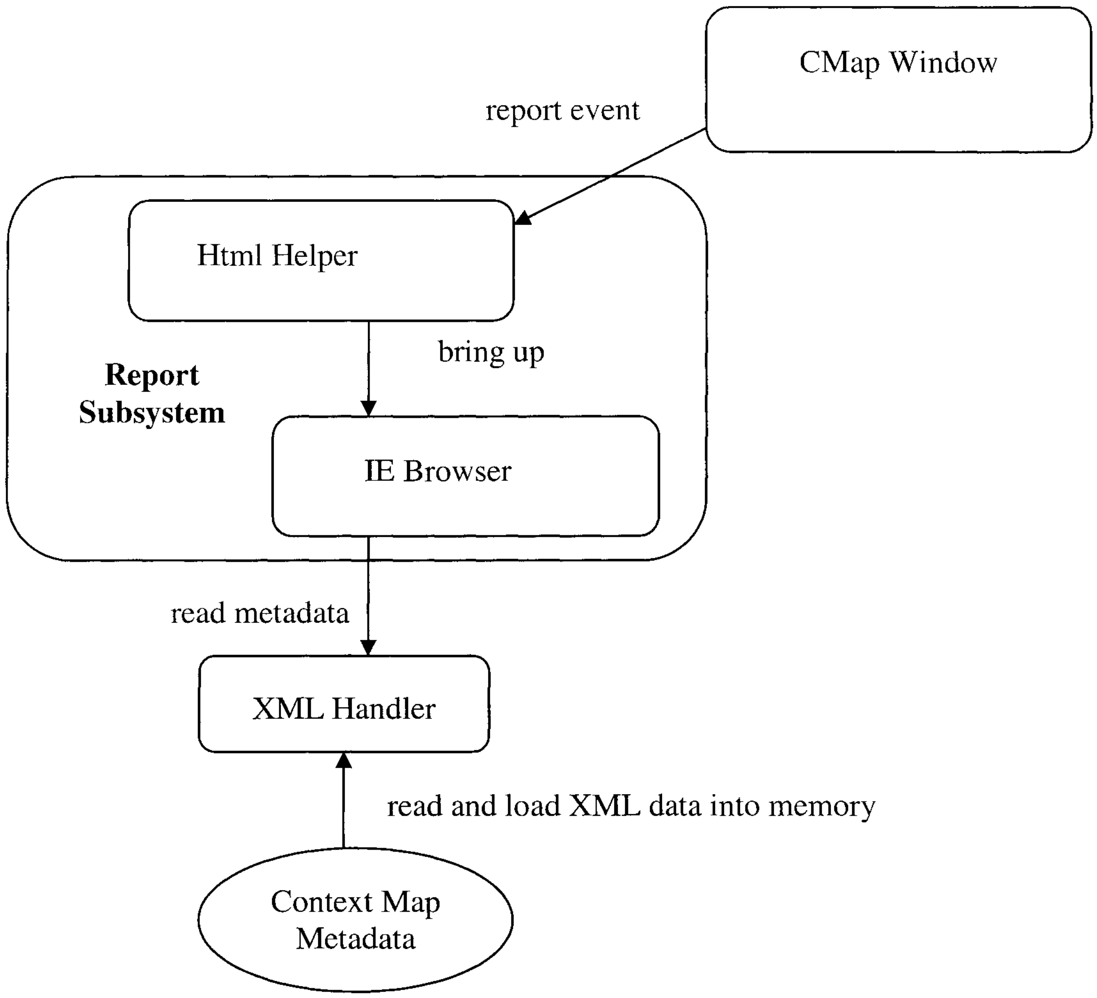


Figure 6-6: Message Sequence Chart for Report Subsystem

6.3.3.5 CMap Window Subsystem

The main responsibilities of cmap window subsystem are described below. For more details refer to the Message Sequence Charts for the flow of operations in Figure 6-7.

- It creates all menus, toolbar, status bar, slider bar, title, etc.

- When user chooses to open a context map XML file, it handles the user actions from the open window and sends corresponding events to the navigation and result table subsystems.
- When user performs the Select or Unselect operation on the selected tree node or table row, it sends the corresponding event to the navigation and result table subsystems.
- When user performs different Query operations, the corresponding events will be sent to the navigation and result table subsystems.
- When user performs the Search operation, it sends the corresponding event to the search subsystem.
- When user performs the Go To operation one selected tree node, it sends the corresponding event to the navigation subsystem.
- When user performs Report operation, it sends the corresponding event to the report subsystem.
- When user performs Help operation, it brings up the Help Topics Browser.

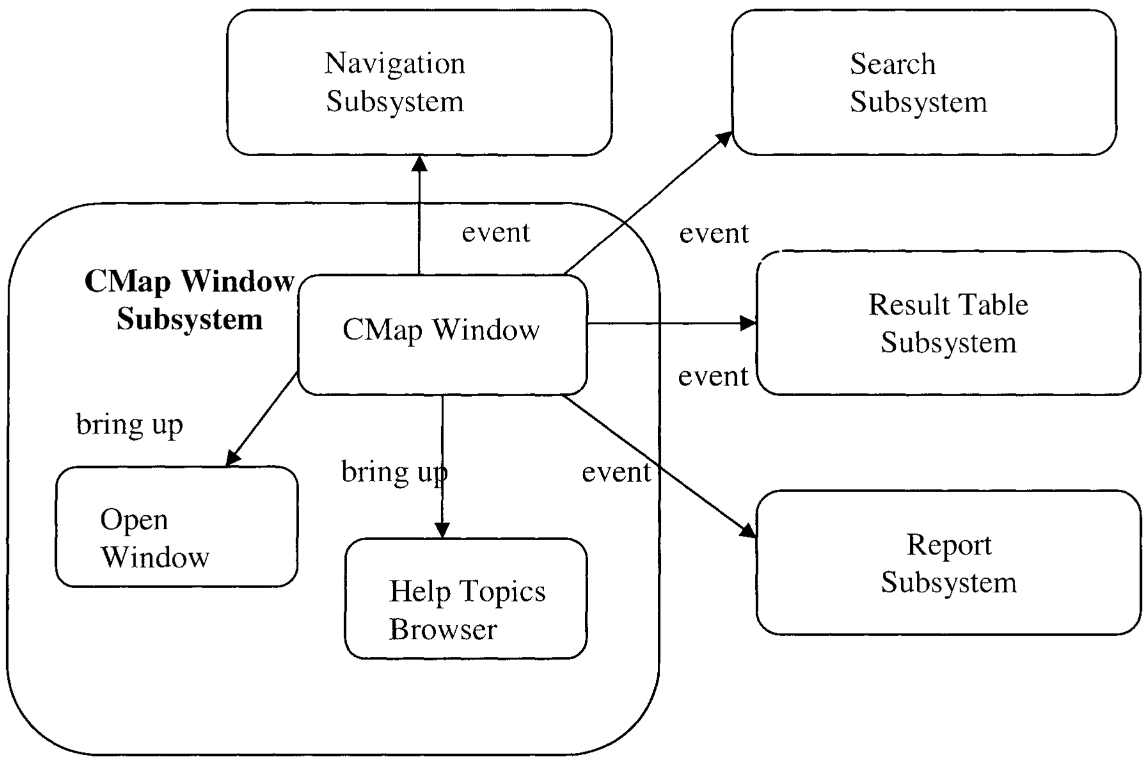


Figure 6-7: Message Sequence Chart for CMap Window Subsystem

6.3.3.6 Context Map Metadata Handling Subsystem

The main responsibility of context map metadata handling subsystem is to read and load the context map metadata into memory when the application starts up. It is the most important subsystem in the CMapViewer application as you saw in pervious described subsystems.

As we saw in Chapter 4, all context map data written to XML files are formatted according to the DTD defined for XML-based context maps. XML handler will read

XML files and convert each XML tag to a running Java object. The parser (cmap.utils.CMapXMLHandler) will convert each tag to an instance of the corresponding Java metadata object. It looks up tag name, creates an instance of the object based on a property file mapping, creates a list of attributes values based on tag attributes, and initializes object with values. The CMapXMLHandler class will be invoked to transform the XML data into an instance of its corresponding Java object. The primary static method in CMapXMLHandler class is the readMetaData method. The other subsystems will use the objects to access data at runtime. Each object is a singleton and is based off a common template. Accessing to one instance gives access to all and they can also be accessed statically as you see in the example below:

Example 6-1: A Part of Atoms.java

```
public static Atoms getByKeys(String given_atom_index)
{
    String key = KEYLETTER + "_" + given_atom_index;
    return (Atoms) classInstancesTable.get(key);
}
```

6.4 Major Features

The CMapViewer application provides an easy, intuitive and powerful solution for manipulating XML-based context maps. The major features of the CMapViewer application described in this section are:

- Pure Java application
- Uses familiar GUI paradigms
- Explorer style atom navigation
- Easy atom search
- Easy query operation
- Reports in HTML format
- Provides a variety of help assistants

In this section, a context map named ACM Map which was developed by Jaworski will be used to demonstrate several major features of the CMapViewer application. The Association for Computing Machinery (ACM) is a group consisting of professionals and students worldwide with diverse backgrounds who share a common interest in computers. This context map represents the Computing Curricula 2001 Report which consists of a set of course descriptions intended to serve as models for institutions offering undergraduate degrees in computer science.

6.4.1 Pure Java Application

The CMapViewer application was developed using the Java programming language by which the software can immediately run on any Java-compatible computer. We can choose to deploy the CMapViewer application on whichever operating system makes the most sense to meet user's needs. For the future development of the context maps, this flexibility can make it much easier, i.e., transforming the CMapViewer application to a web-based or client-server application.

6.4.2 Uses Familiar GUI Paradigms

By conforming to common GUI standards, the CMapViewer application is a standard Windows application. It is easy for users to learn and use the application whenever they use the menu, dialog, toolbar and so on. For example, the following toolbar buttons with universally meaningful icons make it easy for users to understand what kind of operations they represent:

-  Open operation
-  Select operation
-  Unselect operation
- 

Default operation
- 

Schema operation
- 

OR operation
- 

XOR operation
- 

AND operation
- 

NOT operation
-  Search operation
-  Go To operation
-  Report operation

-  Help operation

6.4.3 Explorer Style Atom Navigation

The CMapViewer application is similar to the Windows Explorer in that it aids you to manipulate the context maps much more easily. It has a navigation pane (left) where you can determine the atom location that you are browsing by the tree node index and the atom type by the different tree node icons, respectively. While the context map view is shown on the right pane of the window. Figure 6-8 shows a snapshot of the CMapViewer application after the "ACM Map" is opened.

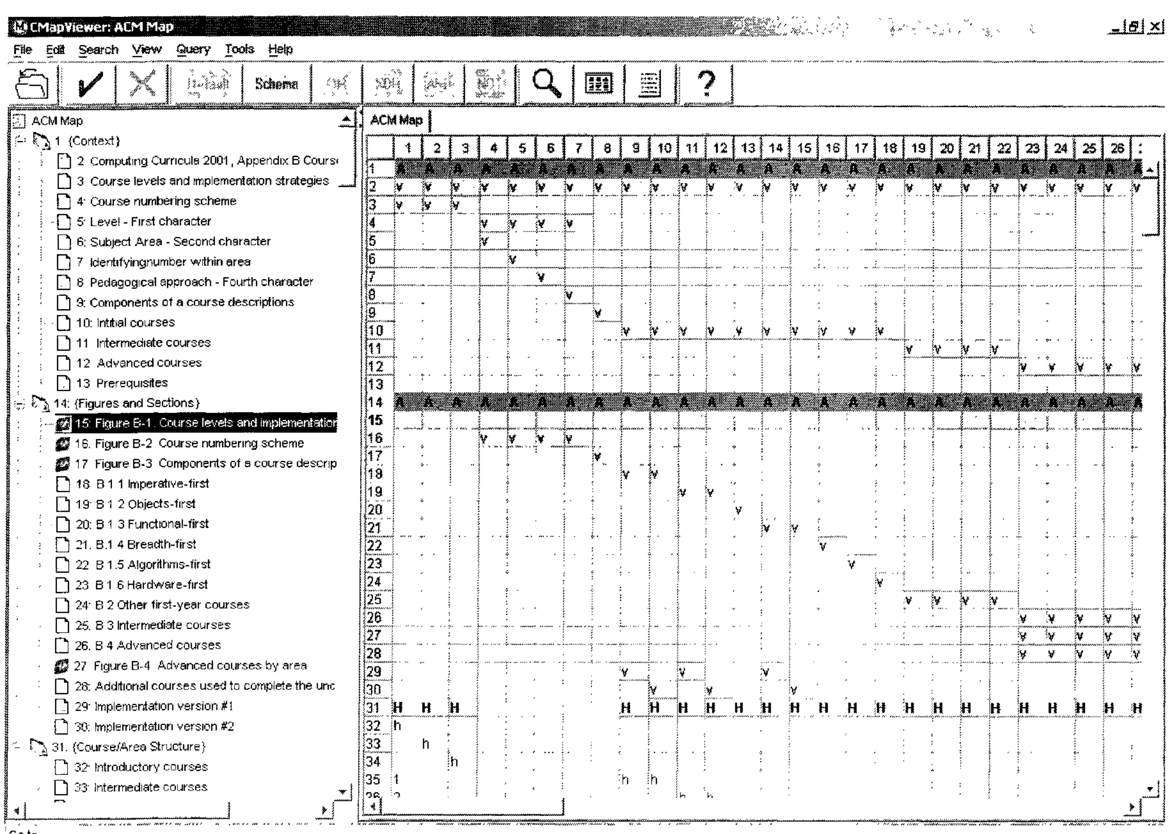


Figure 6-8: CMapViewer Application

As you can see from Figure 6-8, the tree view on the right side of allocation displays the hierarchy of context map atoms. It allows user to easily navigate the atoms. Different types of atoms are indicated by different tree node icons in the tree view. The different atom types represented by the icons are shown below:

-  context map type of atom
-  concept type of atom
-  text type of atom
-  image type of atom
-  link type of atom
-  email type of atom

Based on the different atom type icons, user can quickly know the type of each atom. If the selected atom in tree view is either a link or an email type (currently the image and context map types are not supported), user can directly view the link in browser or compose an email by performing the **Go To** operation or by double-clicking on selected tree node, respectively. For example, the atom in the tree view with index **15** (the highlighted tree node in Figure 6-8) is a link type, the web site page of this hyperlink destination will be opened in an IE browser when you double click on it as shown in Figure 6-9.

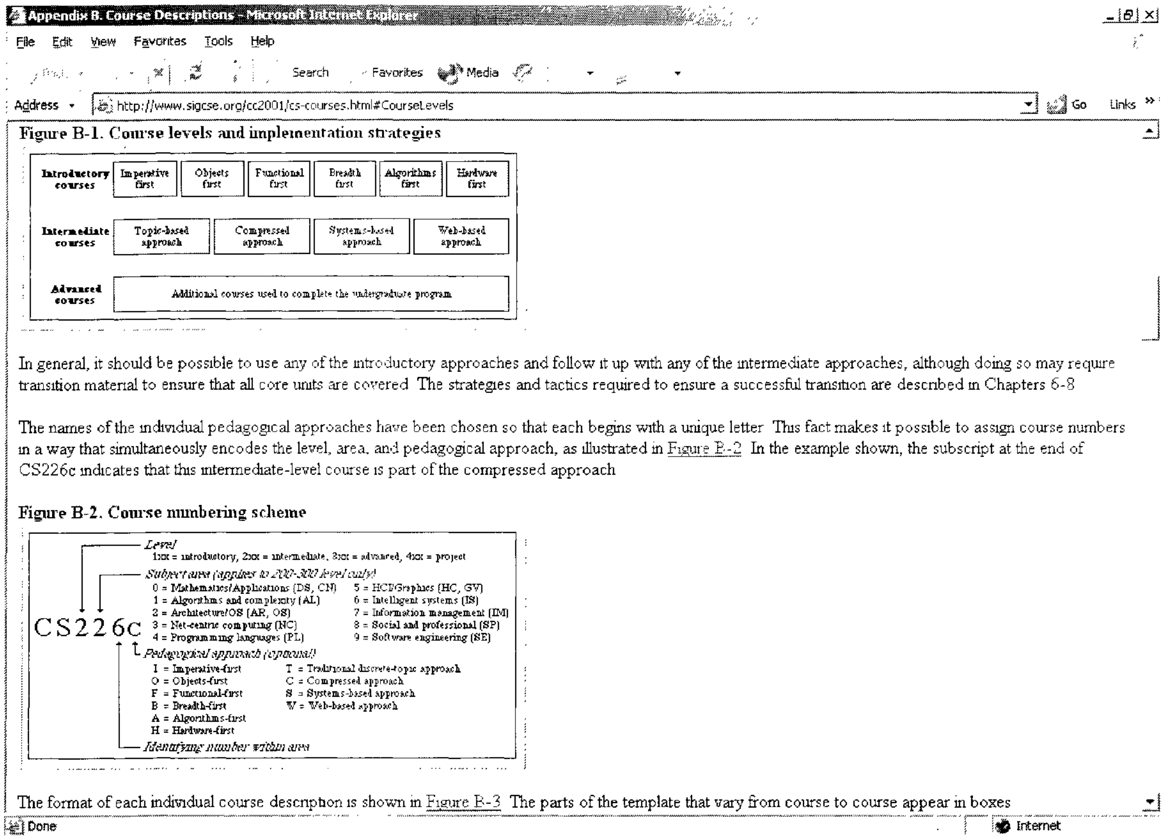


Figure 6-9: The Web Page of Atom 15 in the ACM Map

In addition to the atom type, the atom index in both the tree view and the result table also allow the user to easily access and locate atoms as shown in Figure 6-10. When an atom in the tree view is selected, the corresponding atom in the result table will also be highlighted. It is easy for user to map the atoms between the tree view and the map view when manipulating the context map.

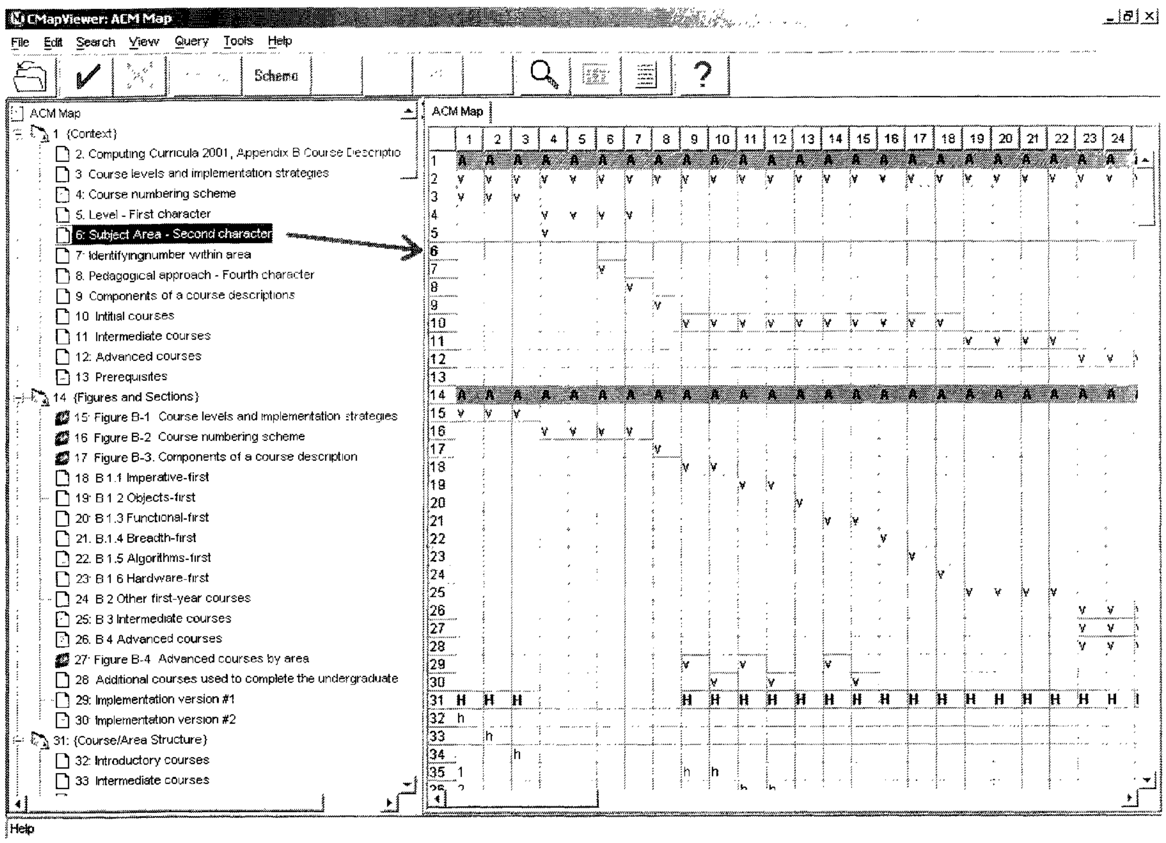


Figure 6-10: Atom Index Mapping in Both Tree View and Result Table

6.4.4 Easy Atom Search

The windows style searching function makes it very easy to find an atom in the tree view especially when the number of atoms is huge (i.e., the ACM Map has 371 atoms). It only requires the user to type in the searching text in the Find window and the found atom will be highlighted in tree view as shown in Figure 6-11. As you can see, the first found atom is "Advanced courses" because it contains the search criteria "advanced".

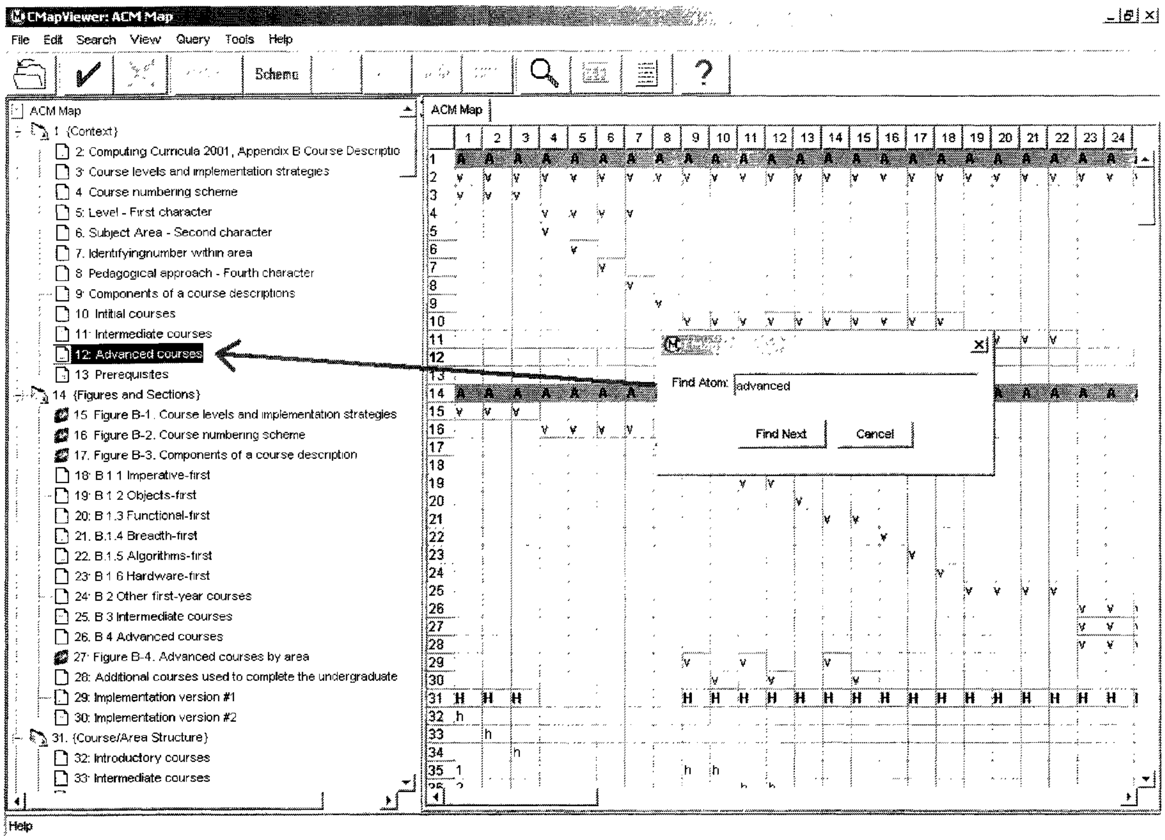


Figure 6-11: The Search Function in CMapViewer Application

6.4.5 Easy Query Operation

The **Default**, **Schema**, **OR**, **XOR**, **AND** and **NOT** query operations provided by the CMapViewer application make it a powerful tool to manipulate the formatted knowledge. By default, a context map is in the **Default** view when it is opened just as shown in Figure 6-8 for the ACM Map. When you click on the **Schema** button from the toolbar, the schema view of the ACM Map will be presented as shown in Figure 6-12. However, you can go back to **Default** view under any circumstances.

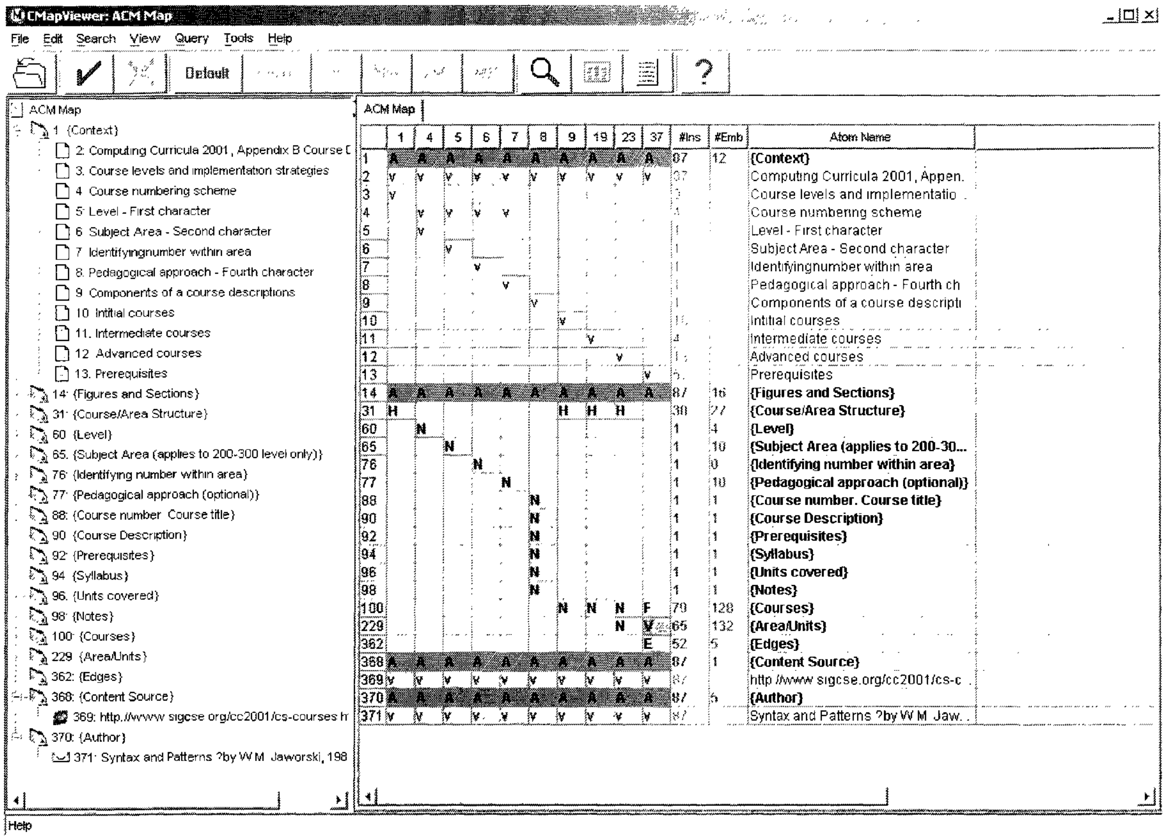


Figure 6-12: The Schema View of the ACM Map

With direct selection of atoms on the navigation tree or in the result table, the query operation is a very easy task and may be as simple as shown in the steps below for the OR query operation on one selected atom:

1. Select the atom with index 15 as shown in Figure 6-13, the selected atom will be highlighted with row index number in result table in red color and in bold font. You can select this atom either in the tree view or from the result table.
2. After clicking on OR operation button, the query result is displayed as shown in Figure 6-14. You can continue other query operations on the selected atom. The query result will be reflected in both the tree view and the result table.

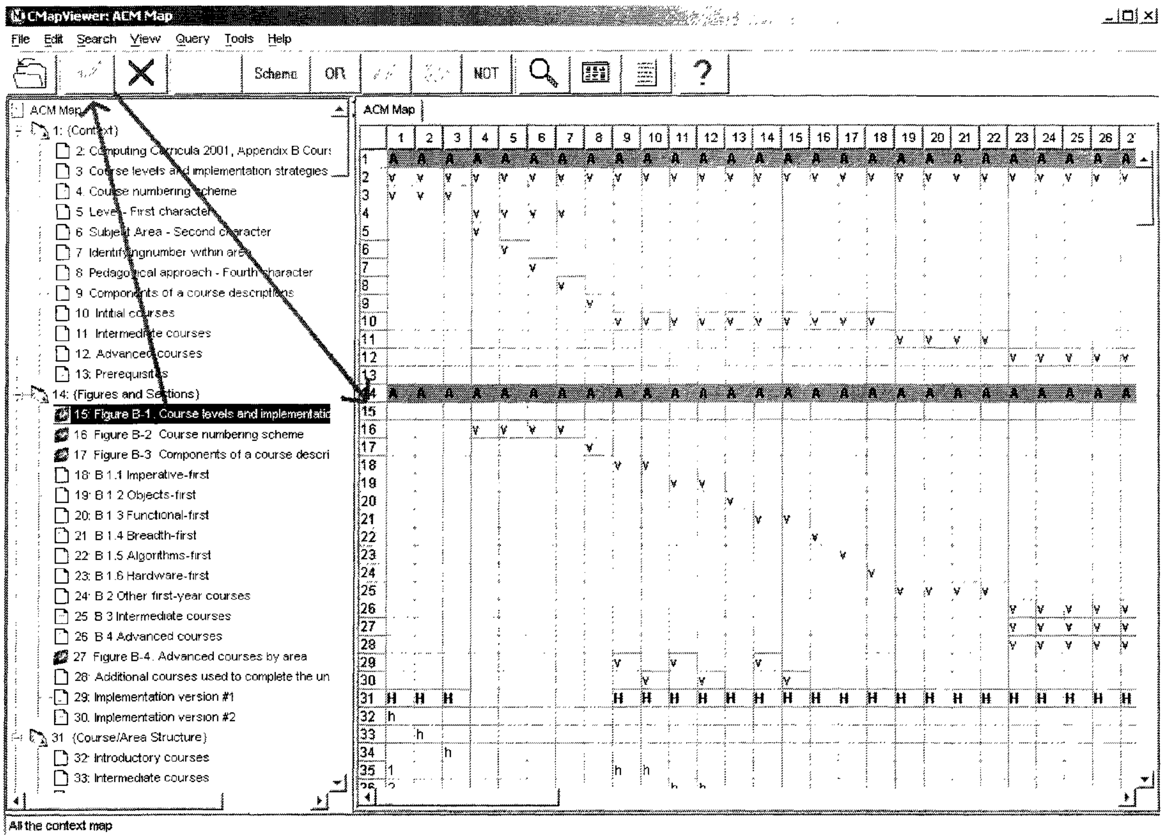


Figure 6-13: The Selection of Atom 15

As we implemented according to the requirements, the menu items or toolbar buttons are enabled/disabled based on whether the command does apply to the current atom selection or not. While some menu items or toolbar buttons may not require any selection. As shown in Figure 6-13, due to the atom index 15 is already marked as a selected tree node, the Select toolbar button is disabled, but user can unselect it because the Unselect button is now in enabled state. Also the NOT query button is enabled because it requires only one atom selection but XOR and AND query buttons are disabled because they require the selection of at least two atoms. By implementing this way, one can prevent user from doing any wrong operations and make the query operation a much easier task.

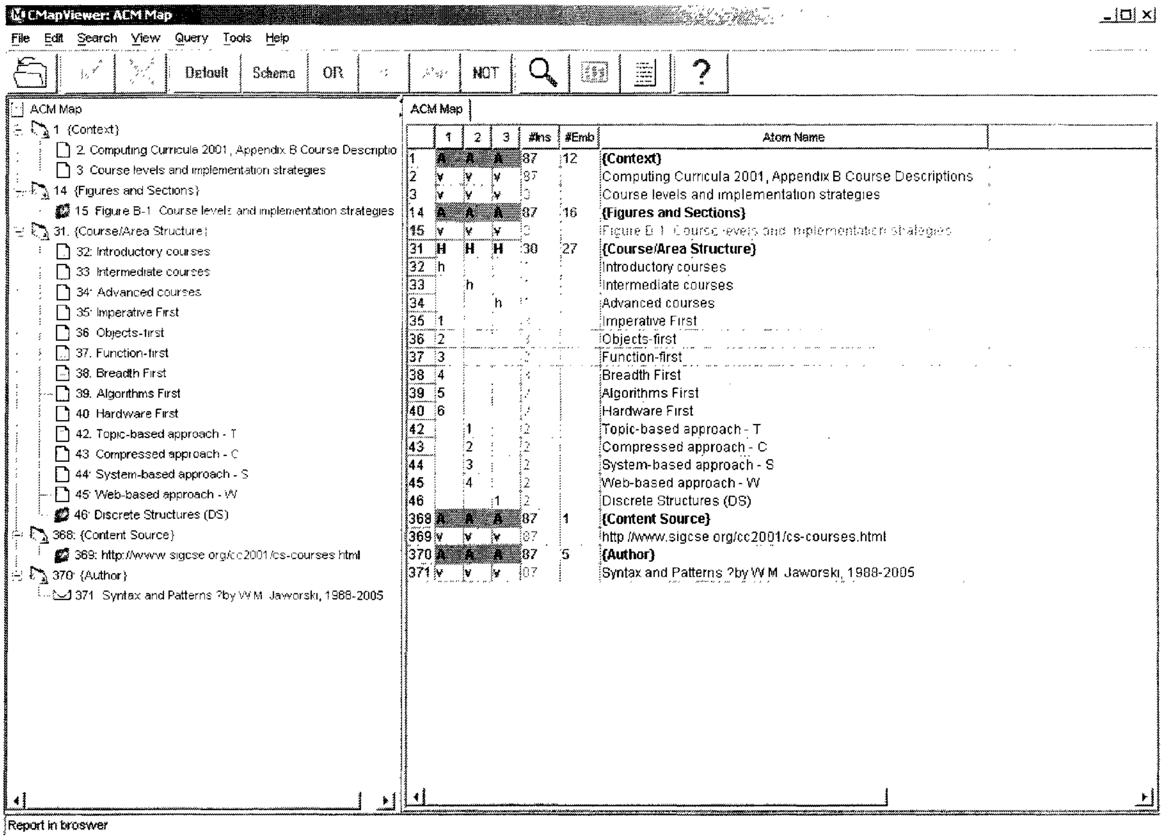


Figure 6-14: OR Query Operation Result on Atom 15

6.4.6 Reports in HTML Format

The CMapViewer application provides a new service to report the context map query result in a HTML-format. The HTML format report makes it easy to backup or print. Figure 6-15 is a HTML format of the query result for Figure 6-14.

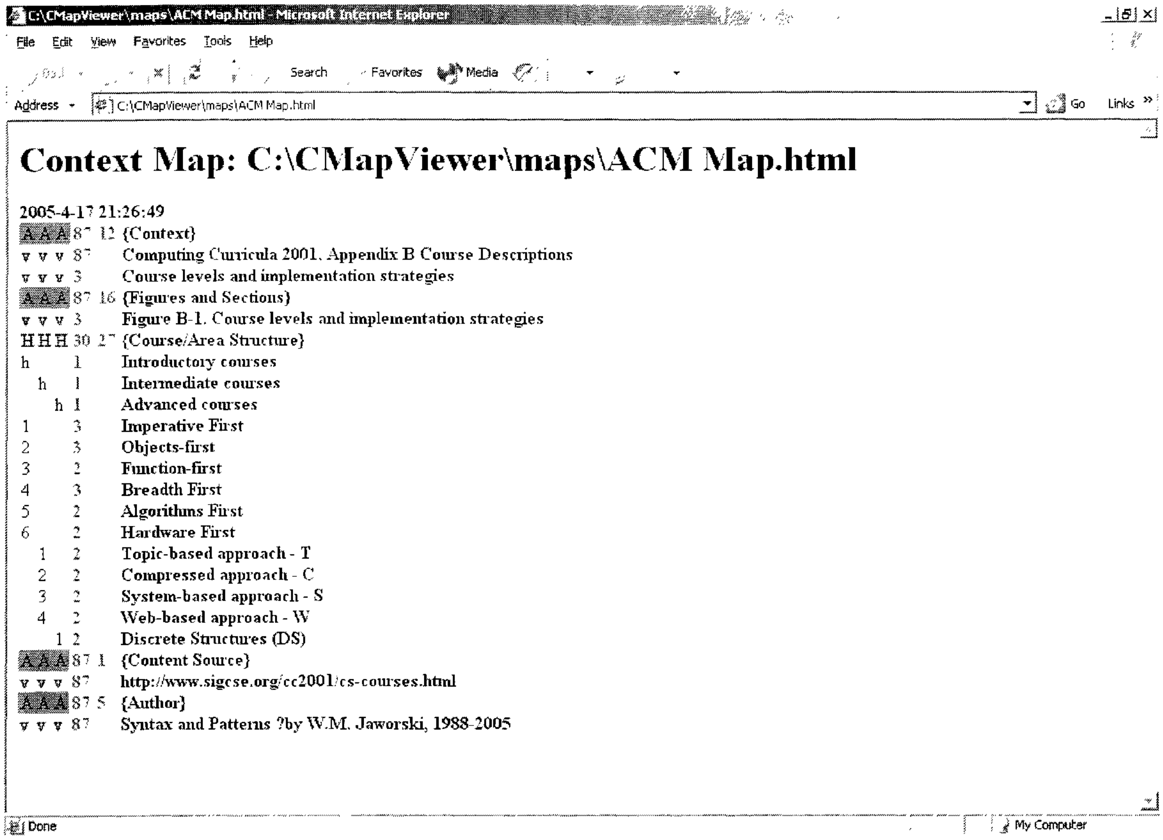


Figure 6-15: HTML Format of the OR Query Result on Atom 15

6.4.7 Provides a Variety of Help Assistants

When user places the mouse pointer over any menu item or toolbar button, a tool tip appears. And the text description of the operation displays on the status bar. However, the most important help assistant is the online help system which was developed by using Sun's JavaHelp solution. JavaHelp is similar to Microsoft's HTML Help system and its viewer is a tri-paned window. One pane includes controls, such as menus and a toolbar. Another includes table of contents, index, and full-text search tabs. The remaining pane is used to display help topics. With this help system, user can quickly to get the information on the context maps and to learn how to use the application. Figure 6-16 shows a snapshot of the JavaHelp browser for CMapViewer application.

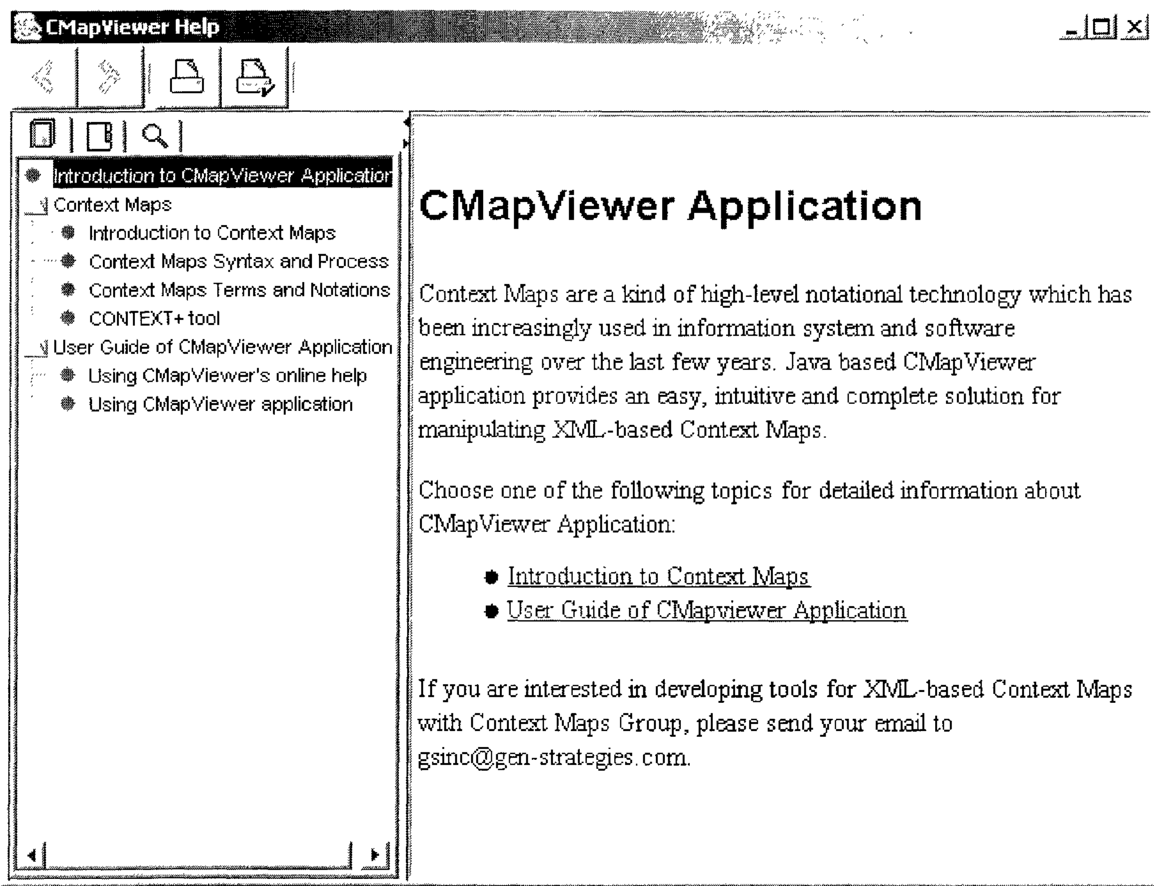


Figure 6-16: JavaHelp Browser for CMapViewer Application

Chapter 7 : Conclusions and Recommendations

7.1 Conclusions

The feasibility of using XML as the document container for the context maps was investigated for the introduction of XML-based context maps. In addition, a new software application was developed as a tool to read and manipulate the XML-based context maps. The conclusions obtained from this study are as follows:

1) With the considerations in mind of a number of short-comings of the current document format for the context maps, the XML-based context maps format was introduced because the merits of XML format include that it is an international standard and the XML-based context maps hold the promise of separating content, structure, semantics and presentation as an electronic document. In this respect standardization initiatives will ensure not only any applications using the XML-based context maps can be implemented across platforms but will also help more public accessibility of the context maps information.
2) An appropriate tool was developed to allow users to transform the current context map documents into XML-based formats. This service demonstrated the successful implementation of solutions to the interoperability of XML-based context maps.
3) A new application named CMapViewer was developed to view and manipulate the XML-based context maps. This Java-based application demonstrates an easy, intuitive and powerful solution for manipulating the formatted knowledge with following features:

- Pure Java application
- Uses familiar GUI paradigms
- Explorer style atom navigation
- Easy atom search
- Easy query operation
- Reports in HTML format
- Provides a variety of help assistants

Moreover, it has greatly improved the potential for developing any web-based applications to meet our ultimate goal which is to create and manipulate well formatted knowledge on the web.

7.2 Recommendations

Recommendations for future the development of the context maps and limitations of this study are described as follows:

1) In order to meet the requirement that the context map's creation and manipulation can be available on the web, converting the standalone CMapViewer application to a web-based or client-server application is a necessity.
2) The force behind the development of the CMapViewer application is to demonstrate the manipulation functionalities on XML-based context maps. It is felt that the abilities of context map creation and validation need to be provided as a complete package solution for the XML-based context maps.

3) The CMapViewer application has the limitation that it currently does not support the manipulation on multiple context maps. This feature needs to be added into the application as an enhancement.

References

[1] W.M. Jaworski, "Representing System Schemata and Templates with jMaps", NOTATE'96, George Washington University, Washington, D.C., May 23-25, 1996

[2] W.M. Jaworski, "jMaps: Conceptual Spreadsheets for Data and Knowledge", Warehousing, 1995.

[3] W.M. Jaworski, "System Analysis and Design in the Classroom: InfoMAPs Teaching Factory", Modeling and Simulation Conference, Pittsburgh, Pa.,May 3-4, 1990.

[4] W.M. Jaworski, "Michailidis A. A., Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", NATW94, University of Massachusetts - Lowell, Massachusetts, June 1994.

[5] W.M. Jaworski, "Conceptual Spreadsheets for Data and Knowledge Warehousing", ATW95 - USA 1995, University of New Hampshire, Durham, New Hampshire, May 31 - June 1, 1995.

[6] W.M. Jaworski, "Cooperative Engineering Issues by Examples: Mapping of Mil498 and NSDIR with jMaps", ATW96-USA 1996, Electronic Systems Center, Hanscom Air Force Base, August 6-9, 1996.

[7] W.M. Jaworski, "Representing Processes, Schemata and Templates with jMaps", Expanded version of the paper presented at Conference on Notational Engineering (a.k.a. NOTATE96), The George Washington University, Washington, DC., May 23-25, 1996.

[8] W.M. Jaworski, Michailidis A. A., "Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", ATW '94, University of Massachusetts - Lowell, Lowell, Massachusetts, June 1994.

[9] W.M. Jaworski, "InfoMaps: Conceptual Spreadsheets for Data and Knowledge Warehousing", ATW '95, University of New Hampshire, Durham, New Hampshire, June 1995.

[10] W.M. Jaworski, et al. "The ABL/W4 methodology for system modeling", System Research Journal 4(1), 23-37, 1987.

[11] W.M. Jaworski, et al. "Representing processes, schemata and templates with jMaps", Semiotica 125(1/3), 229-47, 1999.

[12] W.M. Jaworski, "Representing System Schemata and Templates with jMaps", NOTATE'96, George Washington University, Washington, D.C., May 23-25, 1996.

[13] J.F. Sowa, "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley, Reading, MA 1984.

[14] J.F. Sowa, J.A. Zachman, "Extending and formalizing the framework for information systems architecture", IBM Systems Journal, Vol. 21, NO. 3, 1992.

[15] Conceptual Graphs: <http://www.jfsowa.com/cg/>

[16] M. Chein and M.-L Mugnier, "Conceptual Graphs: Fundamental Notions". Revue d'Intelligence Artificielle, 6(4):365-406, 1992.

[17] Peter Chen, "The Entity Relationship Model - Toward a Unified View of Data ", ACM Transactions on Database Systems, Vol.1, No. 1, pp 9-36, March 1976.

[18] C. Batini, S. Ceri, S.B. Navathe, "Conceptual Database Design: An Entity-Relationship Approach", Addison-Wesley, 1991.

[19] IBM, UML Resource Centre: <http://www.ibm.com/software/rational/uml/>.

[20] OMG (Object Management Group): <http://www.omg.org/uml>.

[21] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1998.

[22] A. Kleppe, J. Warmer and W. Bast. "MDA Explained: The Model DrivenArchitecture.: Practice and Promise", Addisson-Wesley, 2003.

[23] Object Management Group. MDA Guide Version 1.0.1. OMG document, omg/2003-06-01, 2003.

[24] Business Process Modeling Notation (BPMN): <http://www.bpmn.org/>.

[25] Minghu Han, "Associated Data Model and Context Maps", Major Report for Master Degree, Department of Computer Science, Concordia University, 2001.

[26] Kang Zhou, "CONTEXT+: Development Environment for 3P-able Context Maps", Major Report for Master Degree, Department of Computer Science, Concordia University, 2002.

[27] Xu Guo Zhu, "Due Diligence Strategy With Context Maps", Major Report for Master Degree, Department of Computer Science, Concordia University, 2002.

[28] Jing Bai, "Knowledge Mining With Context Maps: Converting OPEN Process methodology to a 3P-able Format", Major Report for Master Degree, Department of Computer Science, Concordia University, 2002.

[29] Puping Peng, "Web Mining With jMap Technology", Major Report for Master Degree, Department of Computer Science, Concordia University, 2002.

[30] Yaozhong Chen, "Application of the Context Map: Modeling the Rational Unified Process and Rational ClearQuest", Major Report for Master Degree, Department of Computer Science, Concordia University, 2002.

[31] Cay Horstmann and Gary Cornell, "Core Java™ 2, Volume I--Fundamentals", Published by Prentice Hall PTR, 2002.

[32] Cay Horstmann and Gary Cornell, "Core Java™ 2, Volume II--Advanced Features", Published by Prentice Hall PTR, 2002.

[33] Stephen Stelting and Olav Maassen, "Applied Java Patterns", Published by Prentice Hall PTR, 2002.

[34] Charles F. Goldfarb and Paul Prescod, "Charles F. Goldfarb's XML Handbook™", Published by Prentice Hall PTR, 2002.

Appendix A: An XML Example

XML-based Context Map file: Inception.cm.xml


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ContextMap SYSTEM "ContextMap.dtd">
<ContextMap>
<CMapDoc>
<Location locatedAt="http://localhost/docs/"/>
<RevisionTable>
<Revisions version="" date="01/12/05" author="" desc="Template"/>
</RevisionTable>
<Reference document_name=" " location=" " date=" " author=" "/>
</CMapDoc>
<ContextMapType Name="Template" cmap_type_name="Inception Phase"
cmap_atom_numbers="17" cmap_relationship_numbers="14" Copyright=" ">
<Description> </Description>
<Atoms atom_index="1" atom_name="{Context View}" is_a_schema="1"
atom_type="0" location="" roles_cardinality="14"
set_members_cardinality="1">
<Relationships atom_index="1">
<Relationship atom_index="1" index="1" label="A"/>
<Relationship atom_index="1" index="2" label="A"/>
<Relationship atom_index="1" index="3" label="A"/>
<Relationship atom_index="1" index="4" label="A"/>
<Relationship atom_index="1" index="5" label="A"/>
<Relationship atom_index="1" index="6" label="A"/>
<Relationship atom_index="1" index="7" label="A"/>
<Relationship atom_index="1" index="8" label="A"/>
<Relationship atom_index="1" index="9" label="A"/>
<Relationship atom_index="1" index="10" label="A"/>
<Relationship atom_index="1" index="11" label="A"/>
<Relationship atom_index="1" index="12" label="A"/>
<Relationship atom_index="1" index="13" label="A"/>
<Relationship atom_index="1" index="14" label="A"/>
</Relationships>
</Atoms>
<Atoms atom_index="2" atom_name="Unified Process" is_a_schema="0"
atom_type="1" location="" roles_cardinality="14"
set_members_cardinality="">
<Relationships atom_index="2">
<Relationship atom_index="2" index="1" label="v"/>
<Relationship atom_index="2" index="2" label="v"/>
<Relationship atom_index="2" index="3" label="v"/>
<Relationship atom_index="2" index="4" label="v"/>
<Relationship atom_index="2" index="5" label="v"/>
<Relationship atom_index="2" index="6" label="v"/>
<Relationship atom_index="2" index="7" label="v"/>
<Relationship atom_index="2" index="8" label="v"/>
<Relationship atom_index="2" index="9" label="v"/>
<Relationship atom_index="2" index="10" label="v"/>
<Relationship atom_index="2" index="11" label="v"/>
```

```
&lt;Relationship atom_index="2" index="12" label="v"/&gt;
&lt;Relationship atom_index="2" index="13" label="v"/&gt;
&lt;Relationship atom_index="2" index="14" label="v"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="3" atom_name="{Phase}" is_a_schema="1" atom_type="0"
location="" roles_cardinality="14" set_members_cardinality="1">
&lt;Relationships atom_index="3"&gt;
&lt;Relationship atom_index="3" index="1" label="A"/&gt;
&lt;Relationship atom_index="3" index="2" label="A"/&gt;
&lt;Relationship atom_index="3" index="3" label="A"/&gt;
&lt;Relationship atom_index="3" index="4" label="A"/&gt;
&lt;Relationship atom_index="3" index="5" label="A"/&gt;
&lt;Relationship atom_index="3" index="6" label="A"/&gt;
&lt;Relationship atom_index="3" index="7" label="A"/&gt;
&lt;Relationship atom_index="3" index="8" label="A"/&gt;
&lt;Relationship atom_index="3" index="9" label="A"/&gt;
&lt;Relationship atom_index="3" index="10" label="A"/&gt;
&lt;Relationship atom_index="3" index="11" label="A"/&gt;
&lt;Relationship atom_index="3" index="12" label="A"/&gt;
&lt;Relationship atom_index="3" index="13" label="A"/&gt;
&lt;Relationship atom_index="3" index="14" label="A"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="4" atom_name="Inception" is_a_schema="0"
atom_type="1" location="" roles_cardinality="14"
set_members_cardinality="">
&lt;Relationships atom_index="4"&gt;
&lt;Relationship atom_index="4" index="1" label="v"/&gt;
&lt;Relationship atom_index="4" index="2" label="v"/&gt;
&lt;Relationship atom_index="4" index="3" label="v"/&gt;
&lt;Relationship atom_index="4" index="4" label="v"/&gt;
&lt;Relationship atom_index="4" index="5" label="v"/&gt;
&lt;Relationship atom_index="4" index="6" label="v"/&gt;
&lt;Relationship atom_index="4" index="7" label="v"/&gt;
&lt;Relationship atom_index="4" index="8" label="v"/&gt;
&lt;Relationship atom_index="4" index="9" label="v"/&gt;
&lt;Relationship atom_index="4" index="10" label="v"/&gt;
&lt;Relationship atom_index="4" index="11" label="v"/&gt;
&lt;Relationship atom_index="4" index="12" label="v"/&gt;
&lt;Relationship atom_index="4" index="13" label="v"/&gt;
&lt;Relationship atom_index="4" index="14" label="v"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="5" atom_name="{Task}" is_a_schema="1" atom_type="0"
location="" roles_cardinality="7" set_members_cardinality="0">
&lt;Relationships atom_index="5"&gt;
&lt;Relationship atom_index="5" index="1" label=""/&gt;
&lt;Relationship atom_index="5" index="2" label=""/&gt;
&lt;Relationship atom_index="5" index="3" label=""/&gt;
&lt;Relationship atom_index="5" index="4" label=""/&gt;
&lt;Relationship atom_index="5" index="5" label=""/&gt;
&lt;Relationship atom_index="5" index="6" label=""/&gt;
&lt;Relationship atom_index="5" index="7" label=""/&gt;
&lt;Relationship atom_index="5" index="8" label="S"/&gt;
&lt;Relationship atom_index="5" index="9" label="S"/&gt;
&lt;Relationship atom_index="5" index="10" label="S"/&gt;
```

```
<Relationship atom_index="5" index="11" label="S"/>
<Relationship atom_index="5" index="12" label="S"/>
<Relationship atom_index="5" index="13" label="S"/>
<Relationship atom_index="5" index="14" label="S"/>
</Relationships>
</Atoms>
<Atoms atom_index="6" atom_name="&lt;task&gt;" is_a_schema="0"
atom_type="1" location="" roles_cardinality="7"
set_members_cardinality="">
<Relationships atom_index="6">
<Relationship atom_index="6" index="1" label=""/>
<Relationship atom_index="6" index="2" label=""/>
<Relationship atom_index="6" index="3" label=""/>
<Relationship atom_index="6" index="4" label=""/>
<Relationship atom_index="6" index="5" label=""/>
<Relationship atom_index="6" index="6" label=""/>
<Relationship atom_index="6" index="7" label=""/>
<Relationship atom_index="6" index="8" label="v"/>
<Relationship atom_index="6" index="9" label="v"/>
<Relationship atom_index="6" index="10" label="v"/>
<Relationship atom_index="6" index="11" label="v"/>
<Relationship atom_index="6" index="12" label="v"/>
<Relationship atom_index="6" index="13" label="v"/>
<Relationship atom_index="6" index="14" label="v"/>
</Relationships>
</Atoms>
<Atoms atom_index="7" atom_name="{Workflow}" is_a_schema="1"
atom_type="0" location="" roles_cardinality="14"
set_members_cardinality="7">
<Relationships atom_index="7">
<Relationship atom_index="7" index="1" label="L"/>
<Relationship atom_index="7" index="2" label="L"/>
<Relationship atom_index="7" index="3" label="L"/>
<Relationship atom_index="7" index="4" label="L"/>
<Relationship atom_index="7" index="5" label="L"/>
<Relationship atom_index="7" index="6" label="L"/>
<Relationship atom_index="7" index="7" label="L"/>
<Relationship atom_index="7" index="8" label="L"/>
<Relationship atom_index="7" index="9" label="L"/>
<Relationship atom_index="7" index="10" label="L"/>
<Relationship atom_index="7" index="11" label="L"/>
<Relationship atom_index="7" index="12" label="L"/>
<Relationship atom_index="7" index="13" label="L"/>
<Relationship atom_index="7" index="14" label="L"/>
</Relationships>
</Atoms>
<Atoms atom_index="8" atom_name="Iteration Planning" is_a_schema="0"
atom_type="1" location="" roles_cardinality="3"
set_members_cardinality="">
<Relationships atom_index="8">
<Relationship atom_index="8" index="1" label="f"/>
<Relationship atom_index="8" index="2" label=""/>
<Relationship atom_index="8" index="3" label=""/>
<Relationship atom_index="8" index="4" label=""/>
<Relationship atom_index="8" index="5" label=""/>
<Relationship atom_index="8" index="6" label=""/>
<Relationship atom_index="8" index="7" label="t"/>
```

```
<Relationship atom_index="8" index="8" label="1"/>
<Relationship atom_index="8" index="9" label=""/>
<Relationship atom_index="8" index="10" label=""/>
<Relationship atom_index="8" index="11" label=""/>
<Relationship atom_index="8" index="12" label=""/>
<Relationship atom_index="8" index="13" label=""/>
<Relationship atom_index="8" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="9" atom_name="Requirements" is_a_schema="0"
atom_type="1" location="" roles_cardinality="3"
set_members_cardinality="">
<Relationships atom_index="9">
<Relationship atom_index="9" index="1" label="t"/>
<Relationship atom_index="9" index="2" label="f"/>
<Relationship atom_index="9" index="3" label=""/>
<Relationship atom_index="9" index="4" label=""/>
<Relationship atom_index="9" index="5" label=""/>
<Relationship atom_index="9" index="6" label=""/>
<Relationship atom_index="9" index="7" label=""/>
<Relationship atom_index="9" index="8" label=""/>
<Relationship atom_index="9" index="9" label="1"/>
<Relationship atom_index="9" index="10" label=""/>
<Relationship atom_index="9" index="11" label=""/>
<Relationship atom_index="9" index="12" label=""/>
<Relationship atom_index="9" index="13" label=""/>
<Relationship atom_index="9" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="10" atom_name="Analysis" is_a_schema="0"
atom_type="1" location="" roles_cardinality="3"
set_members_cardinality="">
<Relationships atom_index="10">
<Relationship atom_index="10" index="1" label=""/>
<Relationship atom_index="10" index="2" label="t"/>
<Relationship atom_index="10" index="3" label="f"/>
<Relationship atom_index="10" index="4" label=""/>
<Relationship atom_index="10" index="5" label=""/>
<Relationship atom_index="10" index="6" label=""/>
<Relationship atom_index="10" index="7" label=""/>
<Relationship atom_index="10" index="8" label=""/>
<Relationship atom_index="10" index="9" label=""/>
<Relationship atom_index="10" index="10" label="1"/>
<Relationship atom_index="10" index="11" label=""/>
<Relationship atom_index="10" index="12" label=""/>
<Relationship atom_index="10" index="13" label=""/>
<Relationship atom_index="10" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="11" atom_name="Design" is_a_schema="0" atom_type="1"
location="" roles_cardinality="3" set_members_cardinality="">
<Relationships atom_index="11">
<Relationship atom_index="11" index="1" label=""/>
<Relationship atom_index="11" index="2" label=""/>
<Relationship atom_index="11" index="3" label="t"/>
<Relationship atom_index="11" index="4" label="f"/>
<Relationship atom_index="11" index="5" label=""/>
```

```
<Relationship atom_index="11" index="6" label=""/>
<Relationship atom_index="11" index="7" label=""/>
<Relationship atom_index="11" index="8" label=""/>
<Relationship atom_index="11" index="9" label=""/>
<Relationship atom_index="11" index="10" label=""/>
<Relationship atom_index="11" index="11" label="l"/>
<Relationship atom_index="11" index="12" label=""/>
<Relationship atom_index="11" index="13" label=""/>
<Relationship atom_index="11" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="12" atom_name="Implementation" is_a_schema="0"
atom_type="1" location="" roles_cardinality="3"
set_members_cardinality="">
<Relationships atom_index="12">
<Relationship atom_index="12" index="1" label=""/>
<Relationship atom_index="12" index="2" label=""/>
<Relationship atom_index="12" index="3" label=""/>
<Relationship atom_index="12" index="4" label="t"/>
<Relationship atom_index="12" index="5" label="f"/>
<Relationship atom_index="12" index="6" label=""/>
<Relationship atom_index="12" index="7" label=""/>
<Relationship atom_index="12" index="8" label=""/>
<Relationship atom_index="12" index="9" label=""/>
<Relationship atom_index="12" index="10" label=""/>
<Relationship atom_index="12" index="11" label=""/>
<Relationship atom_index="12" index="12" label="l"/>
<Relationship atom_index="12" index="13" label=""/>
<Relationship atom_index="12" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="13" atom_name="Test" is_a_schema="0" atom_type="1"
location="" roles_cardinality="3" set_members_cardinality="">
<Relationships atom_index="13">
<Relationship atom_index="13" index="1" label=""/>
<Relationship atom_index="13" index="2" label=""/>
<Relationship atom_index="13" index="3" label=""/>
<Relationship atom_index="13" index="4" label=""/>
<Relationship atom_index="13" index="5" label="t"/>
<Relationship atom_index="13" index="6" label="f"/>
<Relationship atom_index="13" index="7" label=""/>
<Relationship atom_index="13" index="8" label=""/>
<Relationship atom_index="13" index="9" label=""/>
<Relationship atom_index="13" index="10" label=""/>
<Relationship atom_index="13" index="11" label=""/>
<Relationship atom_index="13" index="12" label=""/>
<Relationship atom_index="13" index="13" label="l"/>
<Relationship atom_index="13" index="14" label=""/>
</Relationships>
</Atoms>
<Atoms atom_index="14" atom_name="Iteration Assessment" is_a_schema="0"
atom_type="1" location="" roles_cardinality="3"
set_members_cardinality="">
<Relationships atom_index="14">
<Relationship atom_index="14" index="1" label=""/>
<Relationship atom_index="14" index="2" label=""/>
<Relationship atom_index="14" index="3" label=""/>
```

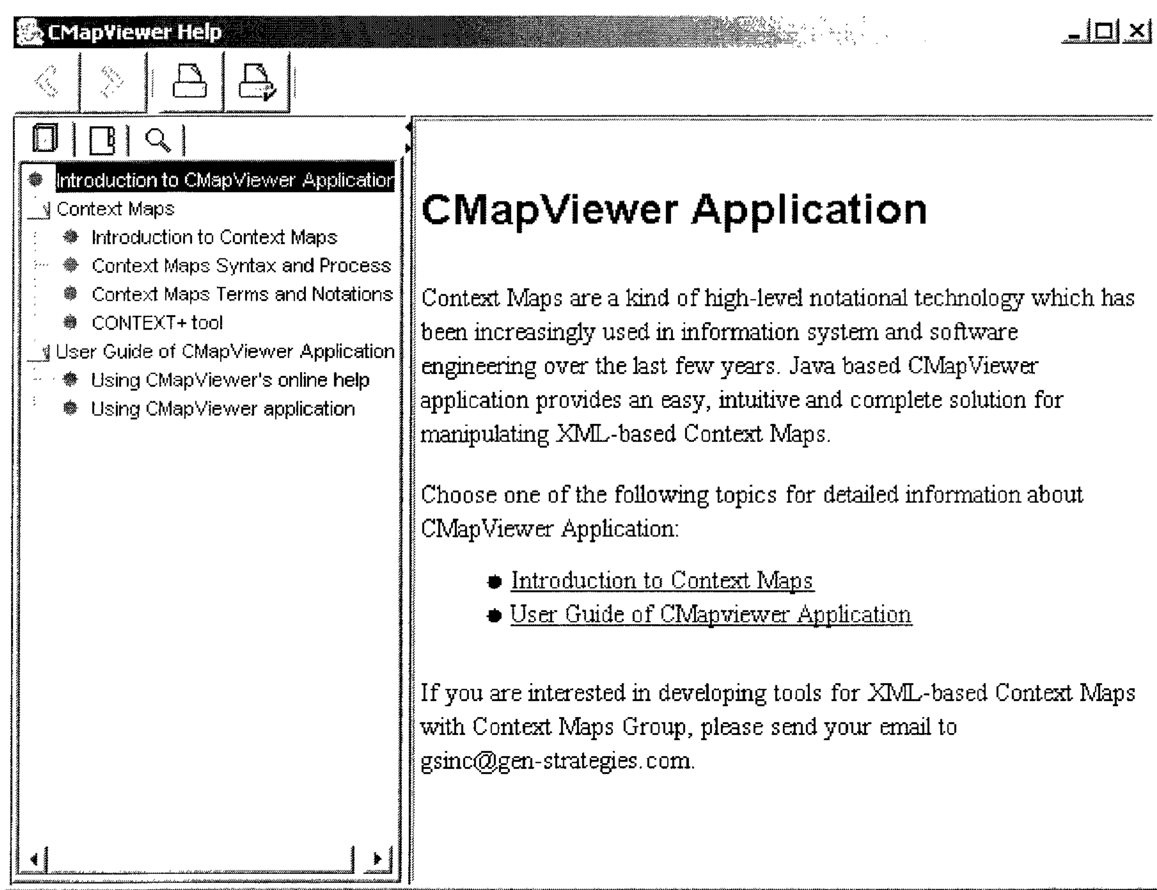
```
&lt;Relationship atom_index="14" index="4" label=""/&gt;
&lt;Relationship atom_index="14" index="5" label=""/&gt;
&lt;Relationship atom_index="14" index="6" label="t"/&gt;
&lt;Relationship atom_index="14" index="7" label="f"/&gt;
&lt;Relationship atom_index="14" index="8" label=""/&gt;
&lt;Relationship atom_index="14" index="9" label=""/&gt;
&lt;Relationship atom_index="14" index="10" label=""/&gt;
&lt;Relationship atom_index="14" index="11" label=""/&gt;
&lt;Relationship atom_index="14" index="12" label=""/&gt;
&lt;Relationship atom_index="14" index="13" label=""/&gt;
&lt;Relationship atom_index="14" index="14" label="1"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="15" atom_name="{Author &amp; Copyright}"
is_a_schema="1" atom_type="0" location="" roles_cardinality="14"
set_members_cardinality="2">
&lt;Relationships atom_index="15"&gt;
&lt;Relationship atom_index="15" index="1" label="A"/&gt;
&lt;Relationship atom_index="15" index="2" label="A"/&gt;
&lt;Relationship atom_index="15" index="3" label="A"/&gt;
&lt;Relationship atom_index="15" index="4" label="A"/&gt;
&lt;Relationship atom_index="15" index="5" label="A"/&gt;
&lt;Relationship atom_index="15" index="6" label="A"/&gt;
&lt;Relationship atom_index="15" index="7" label="A"/&gt;
&lt;Relationship atom_index="15" index="8" label="A"/&gt;
&lt;Relationship atom_index="15" index="9" label="A"/&gt;
&lt;Relationship atom_index="15" index="10" label="A"/&gt;
&lt;Relationship atom_index="15" index="11" label="A"/&gt;
&lt;Relationship atom_index="15" index="12" label="A"/&gt;
&lt;Relationship atom_index="15" index="13" label="A"/&gt;
&lt;Relationship atom_index="15" index="14" label="A"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="16" atom_name="Context Maps @by Qiao Li, 2004"
is_a_schema="1" atom_type="1" location="" roles_cardinality="14"
set_members_cardinality="">
&lt;Relationships atom_index="16"&gt;
&lt;Relationship atom_index="16" index="1" label="v"/&gt;
&lt;Relationship atom_index="16" index="2" label="v"/&gt;
&lt;Relationship atom_index="16" index="3" label="v"/&gt;
&lt;Relationship atom_index="16" index="4" label="v"/&gt;
&lt;Relationship atom_index="16" index="5" label="v"/&gt;
&lt;Relationship atom_index="16" index="6" label="v"/&gt;
&lt;Relationship atom_index="16" index="7" label="v"/&gt;
&lt;Relationship atom_index="16" index="8" label="v"/&gt;
&lt;Relationship atom_index="16" index="9" label="v"/&gt;
&lt;Relationship atom_index="16" index="10" label="v"/&gt;
&lt;Relationship atom_index="16" index="11" label="v"/&gt;
&lt;Relationship atom_index="16" index="12" label="v"/&gt;
&lt;Relationship atom_index="16" index="13" label="v"/&gt;
&lt;Relationship atom_index="16" index="14" label="v"/&gt;
&lt;/Relationships&gt;
&lt;/Atoms&gt;
<Atoms atom_index="17" atom_name="Syntax and Patterns @by W.M. Jaworski,
1988-2002" is_a_schema="1" atom_type="4" location="mailto:wmj@gen-
strategies.com" roles_cardinality="14" set_members_cardinality="">
&lt;Relationships atom_index="17"&gt;
```

```
<Relationship atom_index="17" index="1" label="v"/>
<Relationship atom_index="17" index="2" label="v"/>
<Relationship atom_index="17" index="3" label="v"/>
<Relationship atom_index="17" index="4" label="v"/>
<Relationship atom_index="17" index="5" label="v"/>
<Relationship atom_index="17" index="6" label="v"/>
<Relationship atom_index="17" index="7" label="v"/>
<Relationship atom_index="17" index="8" label="v"/>
<Relationship atom_index="17" index="9" label="v"/>
<Relationship atom_index="17" index="10" label="v"/>
<Relationship atom_index="17" index="11" label="v"/>
<Relationship atom_index="17" index="12" label="v"/>
<Relationship atom_index="17" index="13" label="v"/>
<Relationship atom_index="17" index="14" label="v"/>
</Relationships>
</Atoms>
</ContextMapType>
</ContextMap>
```

Appendix B: User Guide of the CMapViewer Application

Using CMapViewer's online help

CMapViewer displays online help topics in the Help Viewer.



How to get Help?

There are several ways to get help on all topics:

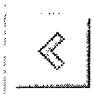
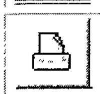
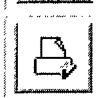
1 Choose Help|Help Topics from the CMapViewer main menu to open the Help Viewer.

2 Click the Help button displayed on the CMapViewer toolbar, or press `F1`.

The main parts of the Help Viewer

The CMapViewer Help Viewer includes:

- The main menu.
- The Contents page, which displays an expandable Table of Contents for all books.
- The Index page, which shows the index entries for all books.
- The Find page, which allows you to enter search words for all books.
- The content pane, which displays the text of the selected topic.
- The following buttons:

Button	Name	Description
	Back	Goes to the previous topic in the history list.
	Forward	Goes to the next topic in the history list.
	Print	Prints the current topic.
	Page Setup	Page setup for printing topic.

Using the Table of Contents

To choose a topic from the Table of Contents:

1. Click the Contents tab in the top left of the Help Viewer.
2. Double-click a book icon to expand the topics in that book.
3. Click an item in the Table of Contents to view the topic in the content pane.

Using the Index

To look up a topic in the Index:

1 Click the Index tab in the top left of the Help Viewer.

2 Type the topic in the text field. As you start typing, the index scrolls, doing an incremental search on the index entries to find the closest match.

3 Double-click the highlighted index topic or press Enter to view the content. If there is more than one topic for the selected index entry, the Index pane splits in two and displays a topic list in the lower portion of the pane. Click the topic you want to display.

Using full-text search

To find text in all documentations:

1 Click the Find tab in the top left of the Help Viewer.

2 Type the search words in the text field. As you type, the search scrolls to find the closest match.

Printing documents

To print documents, open the documentation in the Help Viewer and select File|Print.

Using CMapViewer application

Installation instructions

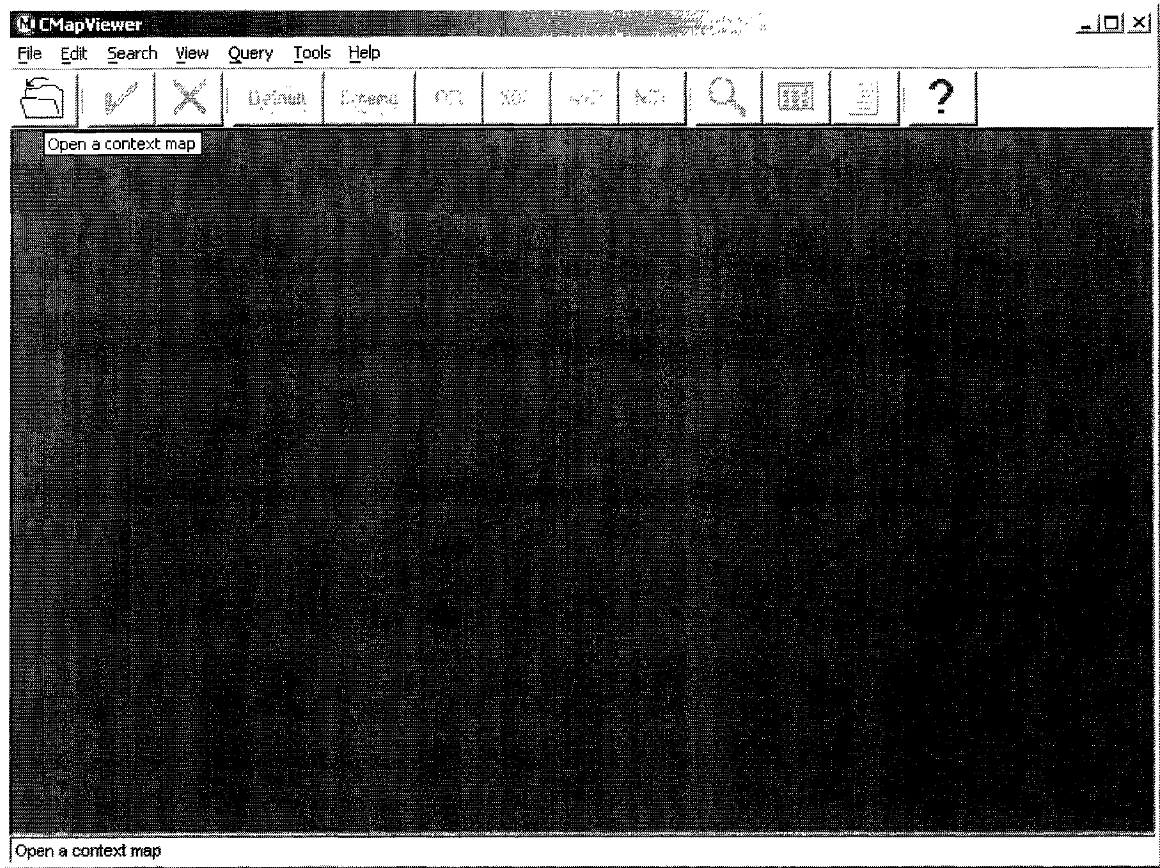
To install the CMapViewer application, you should follow the following simple steps:

1. Unzip the program (cmapviewer.zip) to a location on your hard drive (e.g., D:\cmap).

2. Update the JAVA_BIN parameter in the cmap_run.bat file. You need a jdk1.3 or jdk1.4 installed on your PC.

Launch the application

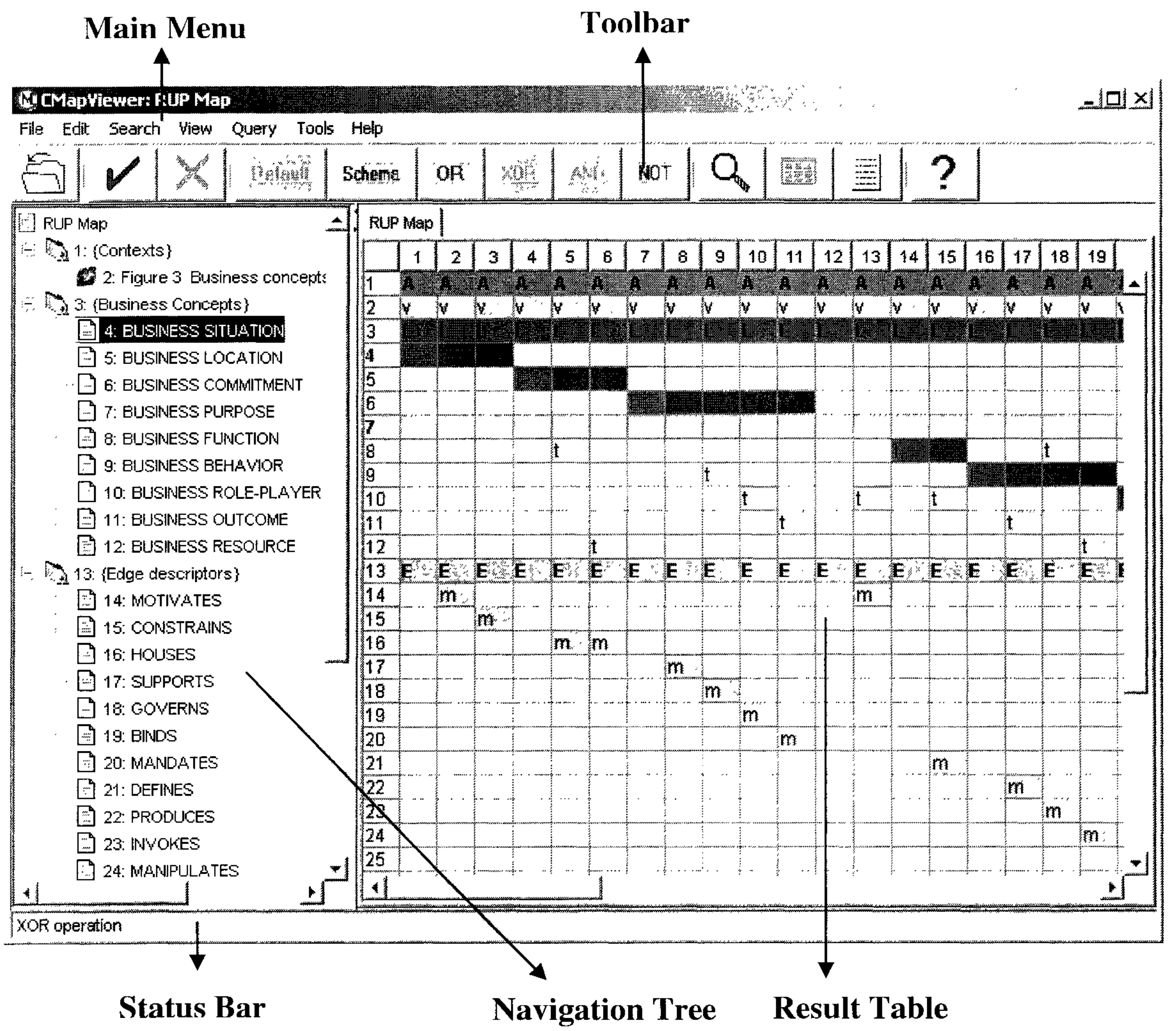
Run the installed program by double-clicking on the cmap_run.bat file. The following is a screen shot of the main window of the CMapViewer application.



Working in CMapViewer main window

CMapViewer uses one window to perform most of the manipulation operations on a context map: selecting, navigating, querying and other operations.

Main window



CMapViewer menus

The main menu is at the top of the CMapViewer main window. Below are the explanations of each menu in the CMapViewer application.

- **File**: The File menu contains commands for opening files and exiting CMapViewer application.

- **Open**: Displays the Open dialog box where you can browse to select the file you want to open.
- **Exit**: Closes the CMapViewer application window.

- **Edit**: The Edit menu contains commands for selecting and unselecting atoms.
  - **Select**: Selects either one tree node or one table row. The selected atom will be marked by displaying the row index in red and bold font in the result table.
  - **Unselect**: Unselect one selected atom. The mark of the selected atom will disappear.
  - **Unselect All**: Unselect all selected atoms. The marks of all selected atoms will disappear.
- **Search**: The Search menu contains one command for finding atom in the tree view.
  - **Find**: Displays the Find dialog box. Using this dialog box to specify the atom you want to locate.
- **View**: The View menu contains commands for viewing toolbar and status bar.
  - **Show Toolbar**: Shows or hides the Toolbar by checking or unchecking it.
  - **Show Status Bar**: Shows or hides the Status Bar by checking or unchecking it.

- **Query**: The Query menu contains commands for doing query operations on the context map.
    - **Default**: Displays the whole context map.
    - **Schema**: Displays the schema of the context map.
    - **OR**: Performs the OR query operation on the selected atom(s) which requires the selection of one or more atoms.
    - **XOR**: Performs the XOR query operation on the selected atoms which requires the selection of more than one atom.
    - **AND**: Performs the AND query operation on the selected atoms which requires the selection of more than one atom.
    - **NOT**: Performs the NOT query operation on the selected atom which requires the selection of only one atom.
- **Tools**: The Tools menu contains commands for displaying addition tools on manipulating the context map.
    - **Go To**: Brings up an IE browser or an email editor if the selected atom is a link or an email type, respectively.
    - **Report**: Brings up an IE browser to display the query result in HTML format.
- **Help**: Commands on the Help menu access the CMapViewer application online documentation set and display the CMapViewer About box.

- o **Help Topics**: Displays the CMapViewer's online documentation in the Help Viewer.

- o **About…**: Displays the About CMapViewer dialog box that contains copyright and version information.

CMapViewer Toolbar

The main toolbar is displayed at the top of the main window under the menu bar. It is composed of toolbar buttons in order by functionality: **Open**, **Select**, **Unselect**, **Deault**, **Schema**, **OR**, **XOR**, **AND**, **NOT**, **Find**, **GO TO**, **Report**, and **Help**. You can also view the toolbar button name and a brief description of the button by moving your cursor over the button. The button name appears below the button, and a brief description of the button appears in the main status bar. The toolbar provides shortcut buttons for the following menu commands:

Icon	Menu Equivalent	Description
	File|Open	Displays the Open dialog box where you can browse to select the file you want to open.
	Edit|Select	Selects one atom.
	Edit|Unselect	Unselects one atom.
Default	Query|Default	Displays the whole context map.
Schema	Query|Schema	Displays the schema of the context map.

OR	Query|OR	Performs the OR query operation on the selected atom(s).
XOR	Query|XOR	Performs the XOR query operation on the selected atoms.
AND	Query|AND	Performs the AND query operation on the selected atoms.
NOT	Query|NOT	Performs the NOT query operation on the selected atom.
	Search|Find	Displays the Find dialog box. Using this dialog box to specify atom you want to locate.
	Tools|GO TO	Brings up an IE browser or an email editor if the selected atom is a link or email type, respectively.
	Tools|Report	Brings up an IE browser to display the query result in HTML format.
?	Help|Help Topics	Displays the CMapViewer's online documentation in the Help Viewer.

CMapViewer Status bar

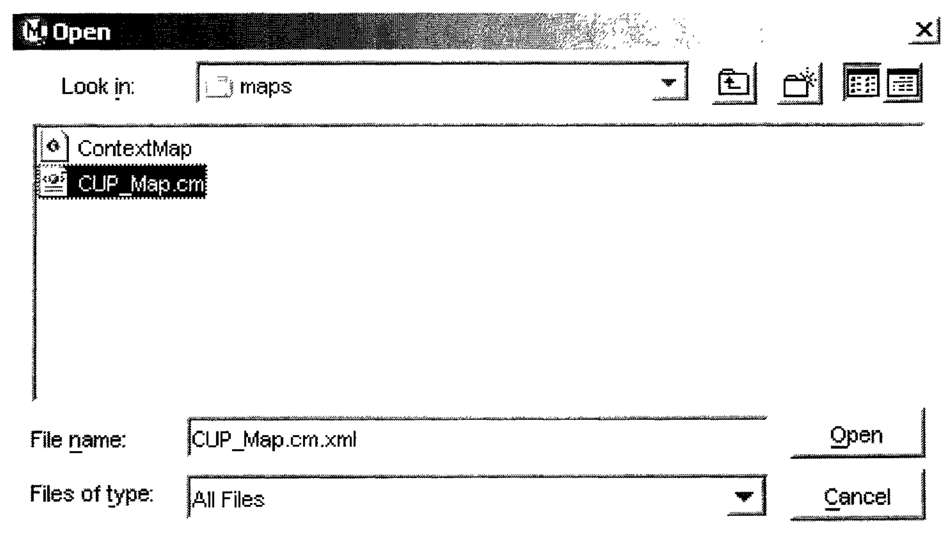
The status bar is displayed at the bottom of the main window and keeps you updated on any operations and their descriptions.

Getting started with CMapViewer application

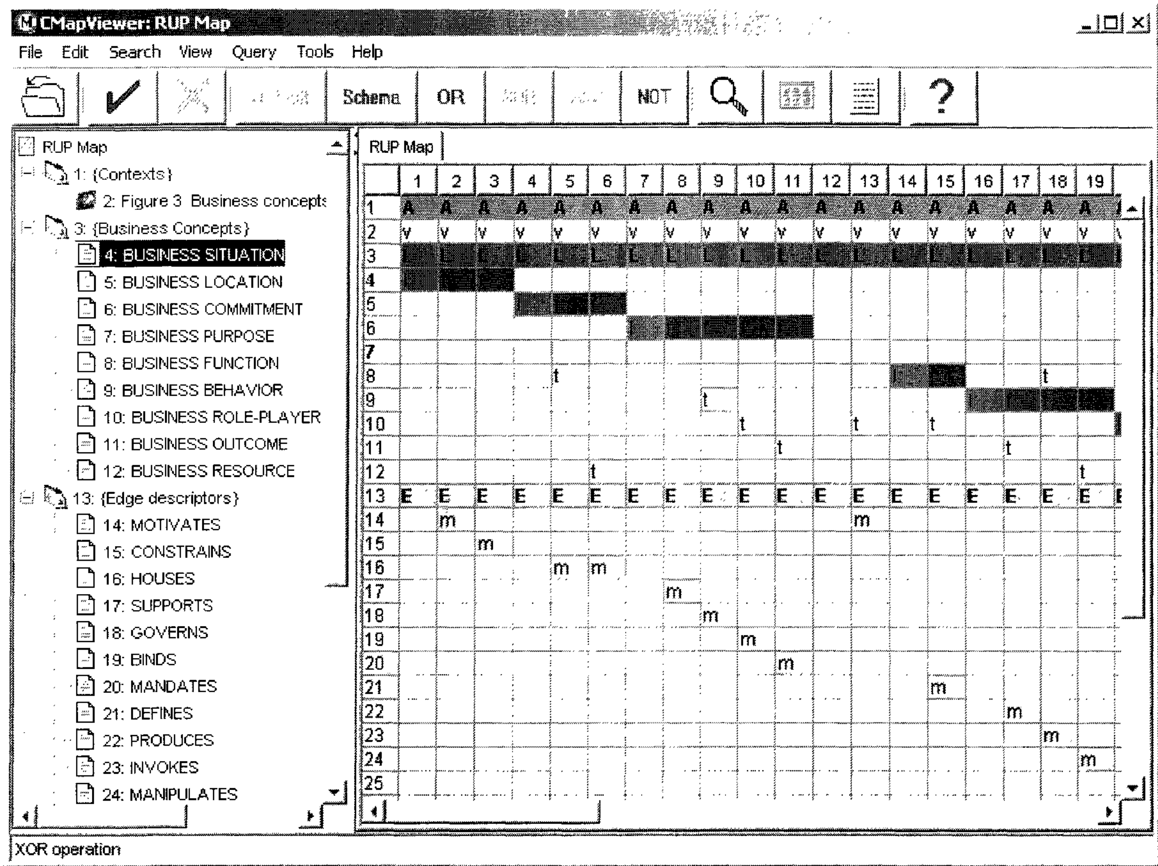
Open a context map xml file

1 In your CMapViewer application, click **File**, and then click **Open**.

2 Use the below **File Selection** dialog box to locate the context map xml files and open the file type recognized by the CMapViewer. Note: The context map xml files with extension *.cm.xml*.



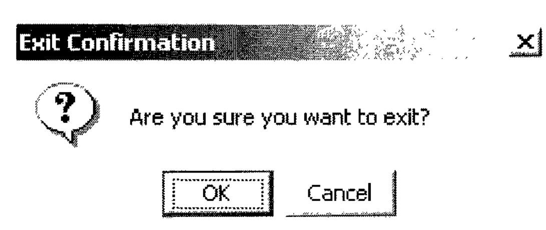
3 Below is the screen shot of the CUP Map after the CUP_Map.cm.xml file is opened.



Exit CMapViewer application

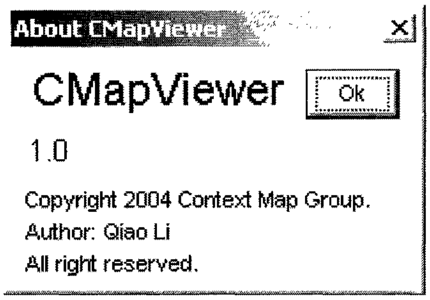
1 In your CMapViewer application, click **File**, and then click **Exit**.

2 Click **OK** on the Exit Confirmation dialog box to exit application.



Access About information

1 In your CMapViewer application, click **Help**, and then click **About…**.



Context Map manipulation

Atom selection

1 Use the mouse to select either a tree node on the tree view or a table row in the result table.

2 Click **Edit** menu, and then click **Select**. Or click **Select** toolbar button.

Atom unselection

To unselect one selected atom:

1 Use the mouse to select either a tree node or a table row which is marked as the selected atom.

2 Click **Edit** menu, and then click **Unselect**. Or click **Unselect** toolbar button.

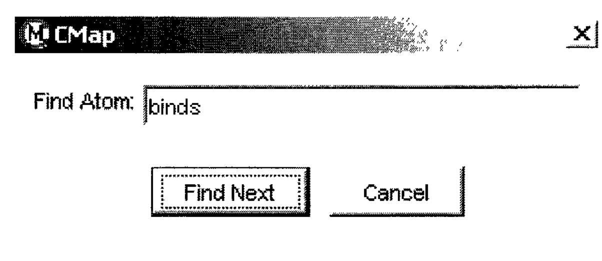
To unselect all selected atoms

1 Click **Edit** menu, and then click **Unselect All**.

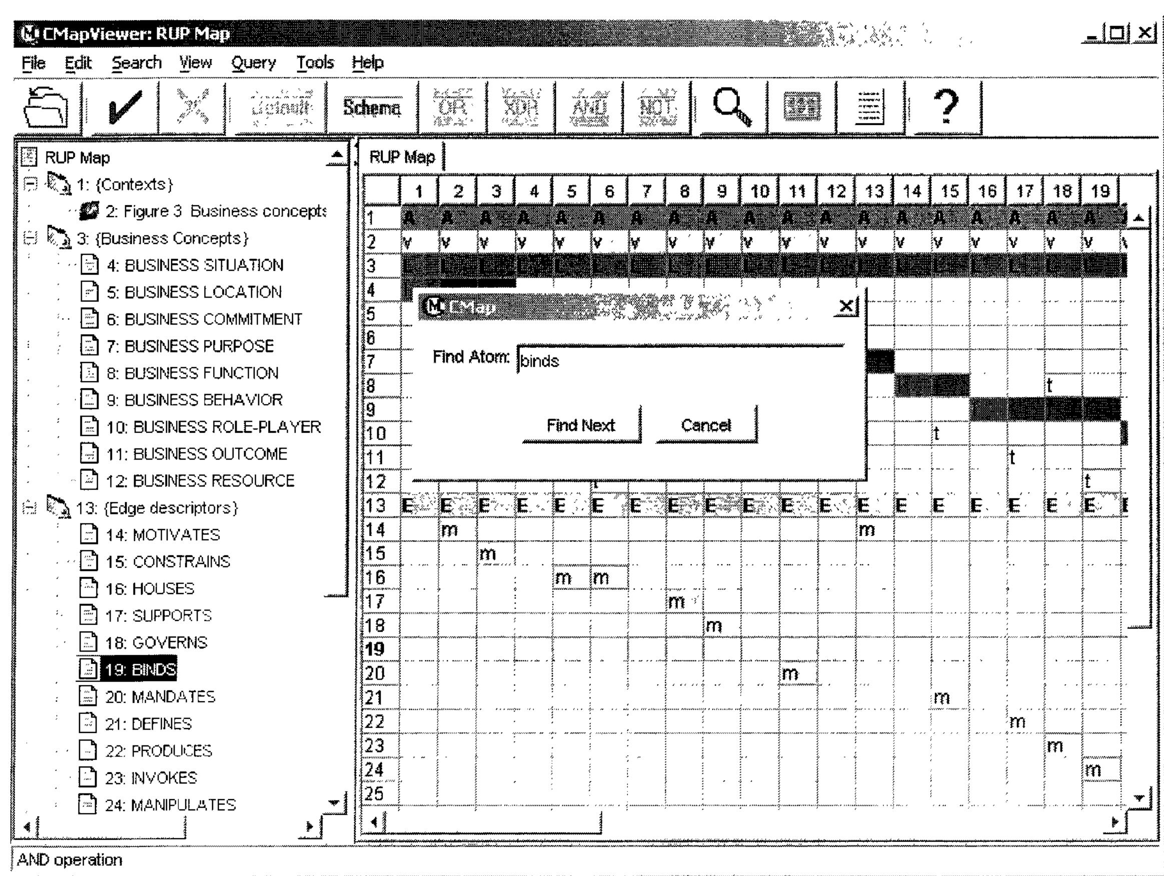
Find an atom

1 Click **Search** menu, and then click **Find**. Or click **Find** toolbar button.

2 Type in the searching text in the below **Find** window.



3 Click **Find Next** button. The found atom is indicated as shown in below example.



Default query operation

To view the whole context map:

1 Click **Query** menu, and then click **Default**. Or click **Default** toolbar button.

Schema query operation

To view the schema of the context map:

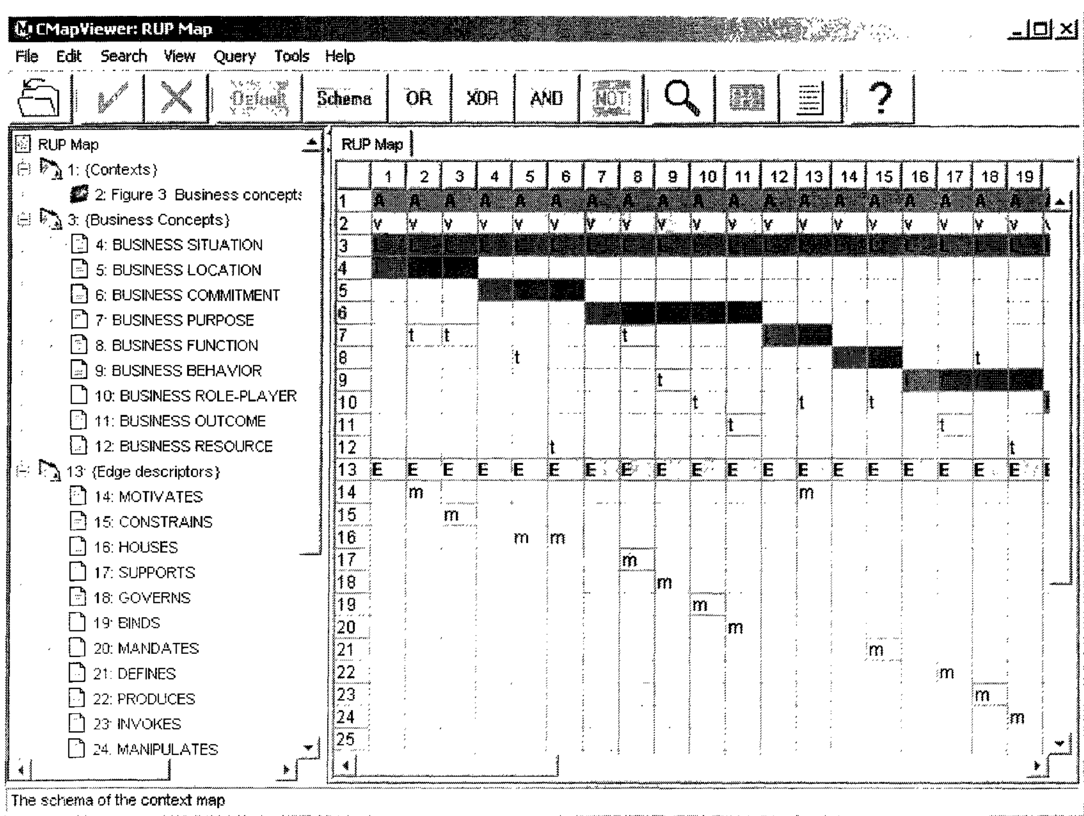
1 Click **Query** menu, and then click **Schema**. Or click **Schema** toolbar button.

Note: All pervious atom selections will be lost.

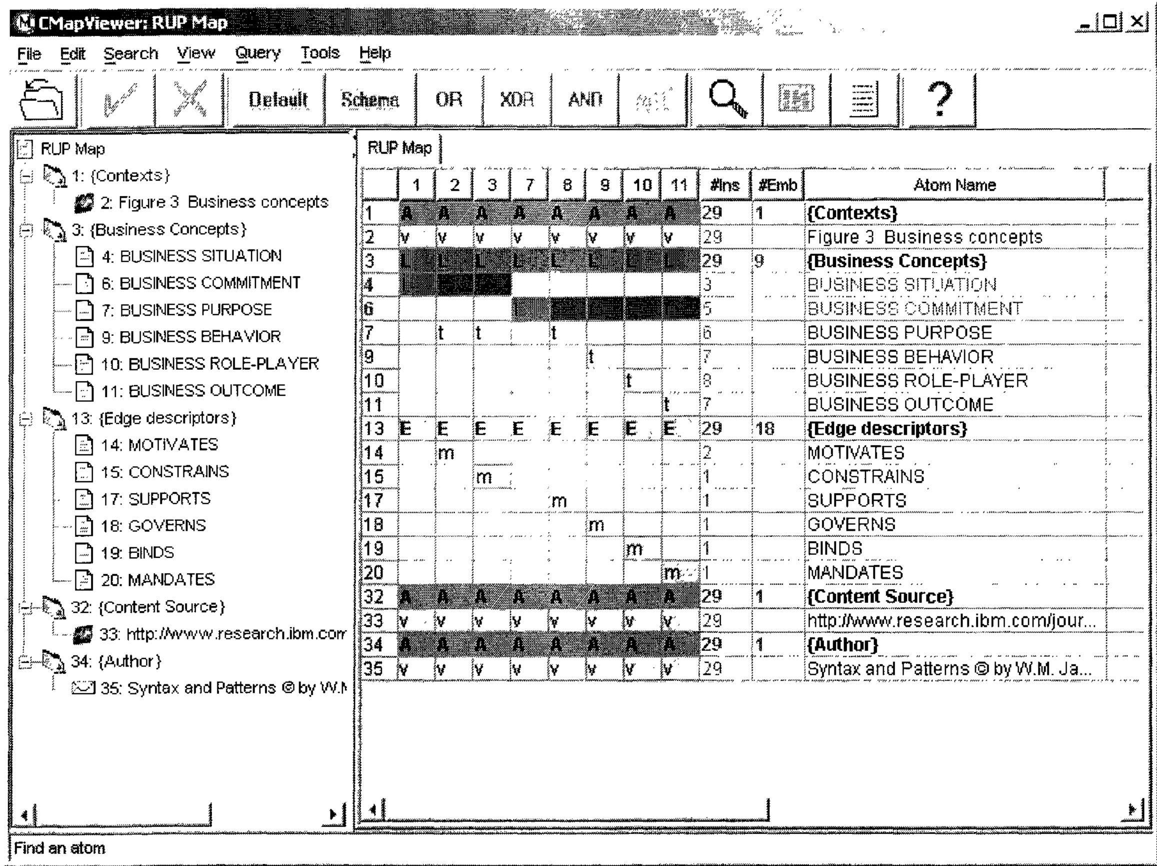
OR query operation

To perform the OR query on the selected atoms:

1 Select the atom(s) you want to query. The below example shows the selection of atom **4** and **6**.



2 Click **Query** menu, and then click **OR**. Or click **OR** toolbar button. Below example is the query result on the atom **4** and **6**.



XOR query operation

To perform the XOR query on the selected atoms:

1 Select the atoms you want to query.

2 Click **Query** menu, and then click **XOR**. Or click **XOR** toolbar button.

AND query operation

To perform the AND query on the selected atoms:

1 Select the atoms you want to query.

2 Click **Query** menu, and then click **AND**. Or click **AND** toolbar button.

NOT query operation

To perform the NOT query on the selected atom:

1 Select one atom you want to query.

2 Click **Query** menu, and then click **NOT**. Or click **NOT** toolbar button.

GO TO operation

1 Use mouse to select one atom which is either a link or an email type. Below are the atom types represented by the icons:

Icon	Atom Type
	context map
	concept
	text
	image
	link
	email

2 Click **Tools** menu, and then click **GO TO**. Or click **GO TO** toolbar button.

If you select the atom on the tree view, you can also double-click it to perform the **GO TO** operation.

Report in HTML

To report the query operation result in HTML format:

1 Do one query operation.

2 Click **Tools** menu, and then click **Report**. Or click **Report** toolbar button. Below is an example of the query operation result in the HTML format.