CONTEXT+: Development Environment for 3P-able Context Maps

Kang Zhou

A MAJOR REPORT

IN

THE DEPARTMENT

Of

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

July 2002

0-612-72952-4

Abstract

CONTEXT+: Development Environment for 3P-able Context Maps

Kang Zhou

This report introduces a kind of high-level notational technology, "Context Maps", first introduced by Dr. Wojciech. M. Jaworski, that can be used in information system and software engineering. In practice, with map applications multiplying, in order to use this technology better, the development of efficient visualization CONTEXT+ tool is necessary and important. This report describes the process of developing and applying CONTEXT+ tool for 3P-able Context Maps in detail. All programs are developed with Visual Basic and the Context Maps are displayed within MS Excel spreadsheet. By applying and processing the various applications, the functionality and attributes of CONTEXT+ tool are completely demonstrated in this report. The main objective is to supply a set of advanced, easy to learning and powerful tool for 3P-able Context Maps. The purpose of this project is to illustrate that Context Maps is not only a methodology and formalized notation for representing the knowledge, but also a powerful tool that can be used directly to manipulate on the formatted knowledge.

Acknowledgements

I wish to thank all those who made the final realization of this dissertation possible. It is not possible to mention all their names, however I would like to express my special gratitude to the following contributors.

I am greatly indebted to my supervisor, Professor Wojciech. M. Jaworski, who encouraged my interest in the knowledge representation field, for his technical advice, generous attention, constant help and critical remarks throughout this work. He patiently guided to me with his knowledge about information systems and encouraged me in tackling various difficult issues.

I am pleased to express my gratitude to Professor Peter Grogono for accepting to be my co-supervisor. He gave me many helpful comments on the report.

My sincere thanks are extended to Professor Olga Ormandjieva, who is an examiner for the report, for her valuable suggestions and support.

My thanks also go to Halina Monkiewicz, the Graduate Program Secretary, for her support and collaboration.

I would like to thank a group of students, ShengTian Yang, YanHong Li, Li Xuan, Yi Chen, MinHua Chen and Zhu Lin Lu, for helping me to convert and verify the source code into ContextMaps.

Finally, my special thanks are also due to the faculties and staffs in the Computer Science Department at Concordia University, who provided extensive support during my master program's study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Context Maps, originally called jMap (jointed map), were first introduced by Dr. W.M. Jaworski in 1999. The technology was initially developed for recovering and refining knowledge from legacy systems. By using the concept of spreadsheet structure, it is feasible to describe and process conceptual information. It is easy to integrate different information into one consistent map using Context maps notation, so it is a kind of high-level notation technology. The use of Context Maps notation 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.

Until today, Context Maps can deal with large amounts of data and represent knowledge assets in 3P-able format. The "3P-able" means Plug-able, Process-able, and Pattern search-able. And "large amounts" indicates data grow exponentially. In order to enhance efficiency and satisfy the new technical requirement, it becomes very necessary to develop the special purpose tools applied in Context Maps. By considering the basic structure of 3P-able format, it is possible for us to supply a set of advanced, easy to learn, and powerful tools for Context Maps to manipulate the formatted knowledge. The CONTEXTt+ tool is developed under this environment. The full package contains basic functionalities (some of which was developed by predecessors (Varacalli [1998] and Stoyanov [1999])) and new functionalities. All of the work was directly instructed by Dr. W. M. Jaworski. He provides theoretical constructs for CONTEXT+ tool.

1.2 Objective of Study

This report introduces a notational technology named Context Maps. The application environment of Context Maps is Microsoft Excel. Microsoft Excel is a spreadsheet program that has been widely used for data storage and manipulation. It allows user easily to analyse data and mine data by using its standard toolkit. However, query in Context Maps requires visualization of increasingly large data sets in multiple dimensions. With the data multiplying, the Excel inherent toolkit is not satisfactory for scientific research, especially for Context Maps studies. Therefore, efficient visualization tools within this environment are necessary.

The objective of this project is to develop CONTEXT+ tool working in 3P-able Context Maps, which can allow users to deal with Context Maps very efficiently and easily. For users of Context Maps, the CONTEXT+ tool increases the efficiency of data manipulation and visualization, and allows exploration of large data sets that would not be done by hand.

The application was written in Visual Basic with emphasis on using Micro Office application, the MS Excel spreadsheet is effective used in this project, and the user-friendly graphical interface (GUI) is utilized to guide users through application steps. Learning CONTEXT+ tool is simple and straightforward.

1.3 Research Procedure

The research work for this report was supervised by Professor Wojciech M. Jaworski. It was started in January 2002. The Context Maps operation environment study has been involved in the development of tools. For CONTEXT+ tool, the predecessor's developed result is absolutely an essential reference and as a component part.

The important steps in the development of this project are:

1) Map out the knowledge of the target, analyze the requirements of this project, and list all functions to be developed.
2) Study the Context Maps, and try to convert the relative model into a spreadsheet with Context Maps notation, focussing on the understanding the basic expression of this new technology in Excel.
3) Get familiar with predecessor's program, and understand the structure of the old code.
4) Design project, develop program for CONTEXT+, source coding in MS Excel by using Visual Basic.
5) Integrate the program, test all functions and adjust CONTEXT+.
6) Clean up the source code, and convert all source code and the user manual into the Context Maps.
7) A final project package will contain a full description of manual, sample Excel file and source code.

8) Make a conclusion for this research work and provide recommendations for future works.

Before finish this project, the report is a very important portion. In this report, the first Chapter introduces the background, objective of study, and the procedure of project.

Context Maps information is introduced in Chapter 2, it includes Context Maps paradigm, technology, syntax, notation, and tools of Context Maps.

Context Maps tool development is main part in whole project. Chapter 3 describes the main goal of CONTEXT+ development, practical application and focus on the features of the CONTEXT+ tool introduction

After CONTEXT+ developed, how to use this CONTEXT+ is very critical to users. Chapter 4 depicts the operation environment, installation, required files, and General Constraints.

In order to illustrating the usage of ContextMap and implementation of the source code, in chapter 5, an efficient means it indroduced to represent the source code and user manual: ContextMaps format decumentation

Chapter 6 provides an evaluative conclusions and recommendations for future work.

In addition, there is a full description of user manual for CONTEXT+ tool. Appendix A focusses on the user manual for tool functionality in detail and provide user an operation solution.

Appendix B provides main detail source code which is written by VB.

Chapter 2

Context Maps: A Notational Technology

2.1 Context Maps Overview

In the computer software industry, there are many technologies and methodologies for specifying, constructing and visualizing software-intensive system. For example, Unified Modeling Language (UML) is a tool with standard notation for expressing a system's blueprint.

Context Maps is advanced, powerful and easy to implement method for information representation. It can be used for representing architectures, structures, and reusable templates of information systems. Context Maps notation allows efficient recovery and modeling of generic schemata for processes, objects and views in these systems.

Concretely, Context Maps is a formal representing method for information system with a set of predefined formal notation. It consists of an unlimited number of context tuples that are generic associations of set members cast in roles. In the extended spreadsheet, a column of roles and the related set members define context tuples. Graphically, a context tuple is represented by a compounded edge and the connected compounded nodes. A directed edge object consists of tail object, middle object and head object. While context tuples represent system behaviors, processes, tasks, procedures and programs, the aggregation of the context tuples forms Context Maps. The Context Maps allows modeling, mining and evaluating context, processes and view of information system with generic templates and domain experts.

2.2 Context Maps Paradigm and Technology

The website ***www.gen-strategies.com*** built by Dr. Wojciech M. Jaworski contains much useful information about Context maps including the evolution of Context maps. Historically, this technology was initially developed as a means of recovering and refining knowledge from legacy system. During the late 1970s and early 1980s, it was named as Array Based Language based on conceptual graphs introduced by J. F. Sowa. In the late 1980s, it was renamed as ABL/W4 (W4 indicates the meaning of what, when, where and which). In the early 1990s, by considering existing notations and methodologies, Professor Jaworski first introduced the concept of Context Maps and named this technology as Context Maps, namely Jointed Map. Until now, Context Maps can represent knowledge assets in 3P-able format (process-able, plug-able, and pattern-able).

Context Maps introduces the concept of creating style sheets to control knowledge based information access and navigation. It represents the relationship between different information nodes in a spreadsheet by vertical columns. In a technical sense, Context Maps describe the information set by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics.

In view of the special property of Context Maps, it is feasible and efficient to describe and process conceptual information by using the popular concept of spreadsheet structure. And by applying the logic query tool with spreadsheet structure, the specific information that users expect to search from the map can be rapidly acquired. So Context Maps is a collection of different information connected together in a logical means and its technology is very powerful.

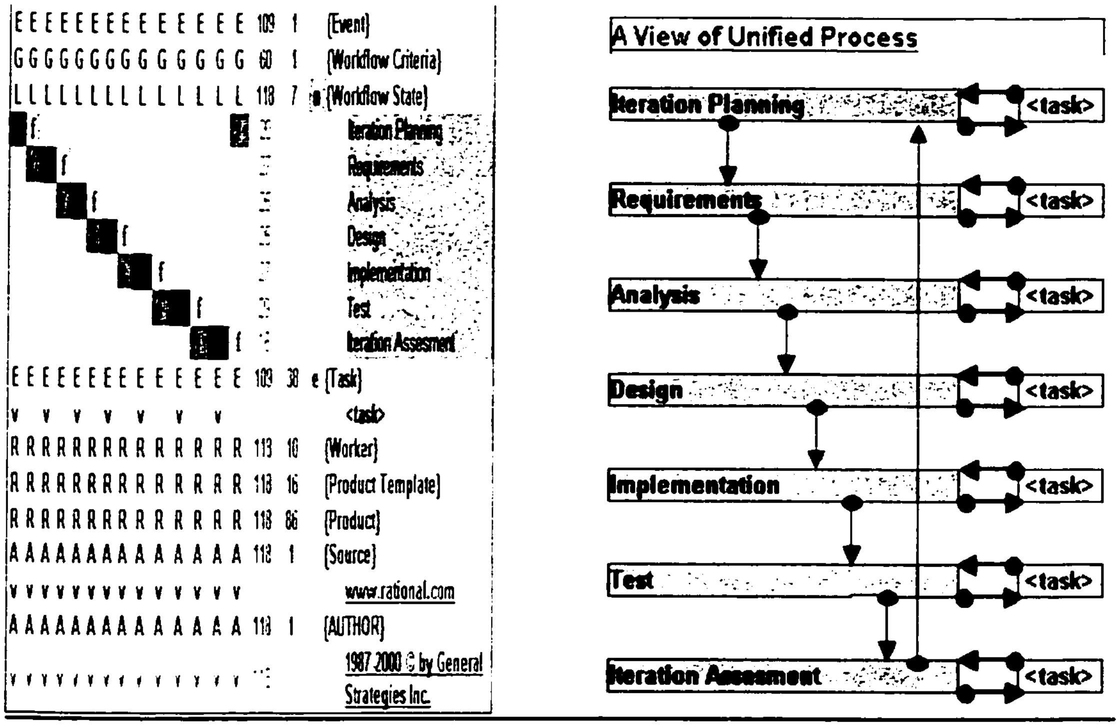
In practice, using this new technology, program source code can also be expressed clearly and readably by Context Maps notation in spreadsheet. In this report, some typical samples will be demonstrated.

2.3 Context Maps Syntax and Process

The syntax of Context Maps is based on the Relationship-Oriented paradigm, defined by relating Sets (concepts) and Set Members. In Context Maps, the relationships are represented by kTuples (i.e. vertical columns in map). The kTuple consists of Set, Set member and Role Tuples. This construct is the fundamental structure defined by the concepts and instances related by roles.

The relating mechanism is implemented by allocating roles to sets in schema and their instance to set components in map. Compared to diagrams, maps are very compact, and offering a rich context within limited space of a computer screen. Maps are created or edited within an organized electronic sheet (MS Excel spreadsheet) that assures efficient manipulation of relationships (columns) and heavy reuse of components (rows).

Figure2-1 (source from http://www.gen-strategies.com/images/Inception.htm built by Dr. W.M.Jaworski) is a sample that demonstrates how to represent the state machine of "Inception" phase in Unified Process to Context Maps. In this phase, each stage can be transferred to its subsequent stage after achieving all the tasks involved in this stage. The left figure illustrates the **Context Maps View** of workflow in "inception" phase, and the right one illustrates the **Graph View** of workflow in "inception" phase.



Context Maps View **Graph View**

Figure2-1 state machine of "Inception" phase

Compared to **Context Maps View** with **Graph View** in the figure2-1, Sets (Iteration Planning, Requirements, Analysis, Design, Implementation, Test, Iteration Assessment) can be represented as "Set members" in rightmost column under Set name {Workflow State}, and arrows can be represented as "Set Member Roles" in column 1-14 marked with lower case letters "f" "t" "l". The terminology and symbol of Context Maps are introduced in 2.3 Context Maps Terms and Constraints and 2.5 Context Maps Notation in detail.

In **Context Maps View**, the schema-level view provides information about Context Maps structure and size. This map contains some sets namely {Event}, {Workflow Criteria}, {Workflow State}, and {Task}. There are some Set Roles namely 'E', 'G' and 'L' and four Member Roles namely 'v', 'l', 'f' and 't'. Set Role 'E' was allocated at {Event} and {Task}. {Task} allows clustering of columns with Member Role 'v'. Set Role 'L' was allocated to {Workflow State} to allow Member Role 'l', 'f' and 't' allocating under it. Set Role 'G' was allocated to {Workflow Criteria}.

If users need to develop large Context Maps models, they can hide irrelevant columns and rows, editing visible cells and inserting new columns and new rows.

In general, Sets appear on the right of the map between bold curly brackets. They can be considered as a heading of a table column or a row. The Roles can be the actual contents listed in the table. Each column is to be read vertically using the syntax. For every "Value" at the spreadsheet, user can read up or down a column and across towards the right of the map to find which Role and Set Member the "Value" is referring to. The user can also obtain the schema of Context Maps by hiding set members and irrelevant columns and can get useful information by applying query tool.

2.4 Context Maps Terms and Constraints

The Context Maps terms used in this report include definitions, acronyms and abbreviations.

Figure2-1, **Context Maps View**, will be used to explain the Context Maps terminology.

- **Context Maps:** represent the relationship between different information Sets and provide functionality of arrays, graphs, relational tables, etc.
- **JMaps:** Abbreviation of Jointed map. The previous name of Context Maps.
- **CONTEXT+:** A set of tools, which was developed for processing Context Maps.
- **Context Tuple**: A generic association of set members cast in roles. In the extended spreadsheet a column of roles and the related set members define context tuple.
- **KTuple**: Abbreviation of Knowledge Tuple. It consists of Set, Set member and Role Tuples, has its own Schema and contains Identifier, Type and Descriptors.
- **Schema**: Work frame of Context Maps. The Context Maps integrate Concepts and Concept Instances with abstract architecture.
- **Set**: Between the bold {} in column17, such as {Event}, {Workflow State}, and {Task}.
- **Set Member**: The members below the bold {} in the most-right column, such as "Iteration Planning", "Requirements" and "Analysis".
- **Set Roles**: the upper case letters in spreadsheet cells, such as letter "E", "G", and "L".
- **Set Member Roles**: the lower case letters or digitals in spreadsheet cells, such as "f", "t", and "l".
- **Cardinality of Roles**: the column 15 counts the number of not empty Roles in every row.

- **Cardinality of Set Member**: the column 16 counts the amount of set members under each Set.
- **Atom:** Anything in a right-most column: set name, value under {Set}.
- **Spreadsheet:** An electronic page in MS Excel, it is used to store and process the data in Context Maps.

The Context Maps constraints:

The Context Maps program must work properly with Windows 95/98/NT, Win2000 and Microsoft Excel. Its development may be influenced by Microsoft Window 98 operating system implementing strategies and policies. MS Excel is constructed by spreadsheet that allows users to organize data, complete calculations, make decisions, graph data, and develop professional reports.

The syntax, schemata, maps, and styles of Context Maps are protected by copyright and trade secret law and may not be disclosed, used or produced in any manner, or for any purpose, except with written permission from Dr. Wojciech M. Jaworski.

2.5 Context Maps Notation

Context Maps is graphical technology for specifying, visualizing and modeling generic schemas with information systems. Context Maps notation is an essential element in this technology. It can be widely employed in many fields such as:

- Information system architecture
- Recovery and reuse of system patterns

- Evolving information systems
- Software evaluation and renewal
- Automation of system design
- Modeling of web sites and knowledge hubs
- Systems workstations

Context Maps notation can be illustrated as the following map:

■	■	■	■	■	■	■	■	■	■	■	■	■	■	**14**	**1**	**{Context Tuple Unique Id}**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	14		Id
■	■	■	■	■	■	■	■	■	■	■	■	■	■	**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}**
[illegible]	[illegible]													2		Is a
		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	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
■		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

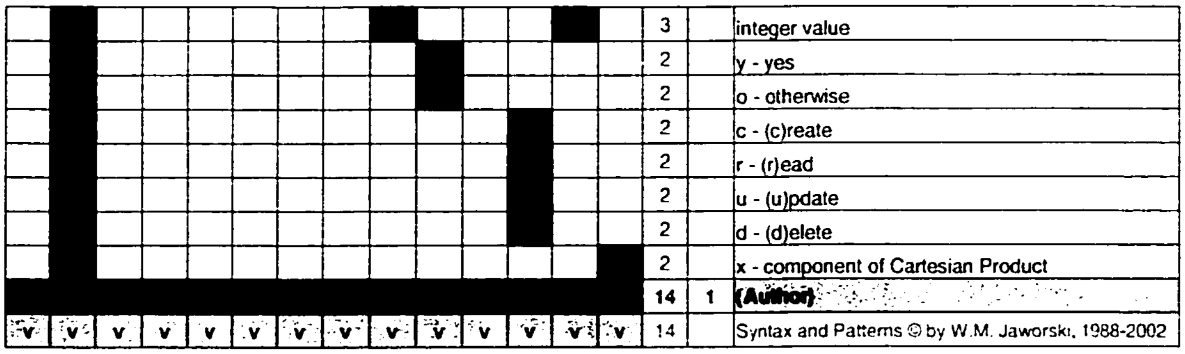

3
integer value
2
y - yes
2
o - otherwise
2
c - (c)reate
2
r - (r)ead
2
u - (u)pdate
2
d - (d)elete
2
x - component of Cartesian Product
14
1
{Author}
v v v v v v v v v v v v v v
14
Syntax and Patterns © by W.M. Jaworski, 1988-2002

Figure2-2 The Context Maps Notation

In practic, some symbols can stand for different meaning, and for special need, different users can define by themselves. The defined regulations are flexible, however easy to understanding is more important.

2.6 Context Maps Tools

The Context Maps tools named "CONTEXT+" have been developed for a specific purpose. As custom-made tools, they can work with Context Maps properly to retrieve the useful information and generate the corresponding maps. They are becoming the indispensible parts of Context Maps. The primary functions include:

- Show Schema
- Compute Cardinality
- Apply Color
- Help
- Mode Selection
- Query
- Output Format: Map or Graph

The "Query" is the most important function in "CONTEXT+" tool. It gives user a convenient and flexible way to manipulate and control the entries of Context Maps. For its detailed usefulness, benefits, development and implementation, the following chapter will discuss in depth.

Chapter 3

CONTEXT+: Development Environment for Context Maps

3.1 Main Goal

The application environment of Context Maps is Microsoft Excel. Microsoft Excel is a spreadsheet program that has been widely used for data storage and manipulation. It allows user easily to analyse data and mine data by using its standard toolkit. However, the Excel program is not tuned for scientific research, especially Context Maps studies. For example, data in Context Maps needs to have color applied based on different value in a big map, which could be thousands of rows and hundreds of columns, or retrieve small part of useful and meaningful data from big map and generate compounding maps, etc. The labor involved with such data exploration has become increasingly onerous. The standard toolkit in Microsoft Excel doesn't provide functionality to meet this goal. Also there are no such tools from third-parties to provide similar functionality. In this circumstance, the challenge comes out to develop an automatic tool which allows users to manipulate data efficiently. The objectives for developing such tool are:

1. To develop a user-friendly interface which provides user with a number of options to easily and efficiently process data in big maps.
2. To apply this tooi for processing some 3P-able format Maps, to calibrate the tool and hence validate its capabilities.

This report presents new tools for Context Maps, namely CONTEXT+. The program is designed using Visual Basic as an Excel macro program; it can be applied on any system with Excel 97 or higher. A user-friendly graphical interface (GUI) is utilized to guide users through application steps; the usage of this tool is simple and straightforward.

3.2 3P-able Format Introduction

With the development of Context Maps technology, Dr. Wojciech M. Jaworski [2002] introduced a new concept "3P-able". Today, Context Maps can deal with large amounts of data and represent knowledge assets in 3P-able format. 3P-able can be expressed as the following formula:

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

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

Context Maps is a collection of different knowledge connected together in a logical manner. Many scattered knowledge assets, concepts, and relationship among these concepts can be processed and integrated into one Context Map, so that a formatted and cleared view is presented in front of users. Concretely, Context Maps should be able to merge separate knowledge into one view in the horizontal and vertical. The "Join Maps" of CONTEXT+ was developed according to this requirement.

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

Context Maps technology originally supports this requirement. In order to speed up the efficiency of processing Maps, it is more flexible and more convenient to manipulate and view information of Context Maps with CONTEXT+ developed and employed.

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

  Context Maps is 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 Context Maps. By using the custom-make Query of CONTEXT+, it is convenient to get the specific knowledge that users expect to search from the map.

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

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

The order of KTuple elements is loss-less.

- K-Tuple is a member of one or more KViews.
- K-View is a member of one or more KHubs.
- K-Hub is a member of one or more KDomains.

3.3 Tool Features

The "CONTEXT+" tool mainly includes the following four functions:

1) Show Schema
2) Query
3) Compute Cardinality
4) Apply Color

"Query" is the most useful function in the "CONTEXT+" tool. It prompts a user-friendly interface, and gives users a convenient and flexible method to manipulate and control Context Maps. "Show Schema", "Compute Cardinality" and "Apply Color" are autonomic commands which provide producing a zoom-in map i.e. work frame of Context Maps, computing the number of non-empty cells in concept rows and the number of sub-concept value under this concept, and applying different color based on the values of cells functions individually. They also appear in the Query Form, which can be combined with other options, and provide the same functionality.

In the Query Form GUI, there are four sub-sections that are "Mode", "Query", "Output" and "Run". The following will describe the main features of them.

3.3.1 Query Usefulness in Context Maps

The Query Usefullness function provides user an efficient way to retrieve the useful and meaningful data from map. Normally, a signal map of Context Maps may have thousands of rows and hundreds of columns so that the whole sheet can't be displayed within the limited space of computer screen. Therefore, manually handling the data with such big map is very time consuming work and also errors are easily generated. In this case, Query provides user a tool to automatically finding and retrieving relevant data based on different operations.

In the "Query" section of Query Form, there are four options "AND", "XOR", "OR", "NOT" with another choice "By Color".

First of all, "AND" allows user to select at least two cells from different rows to generate corresponding map which contains all the not empty columns within these rows.

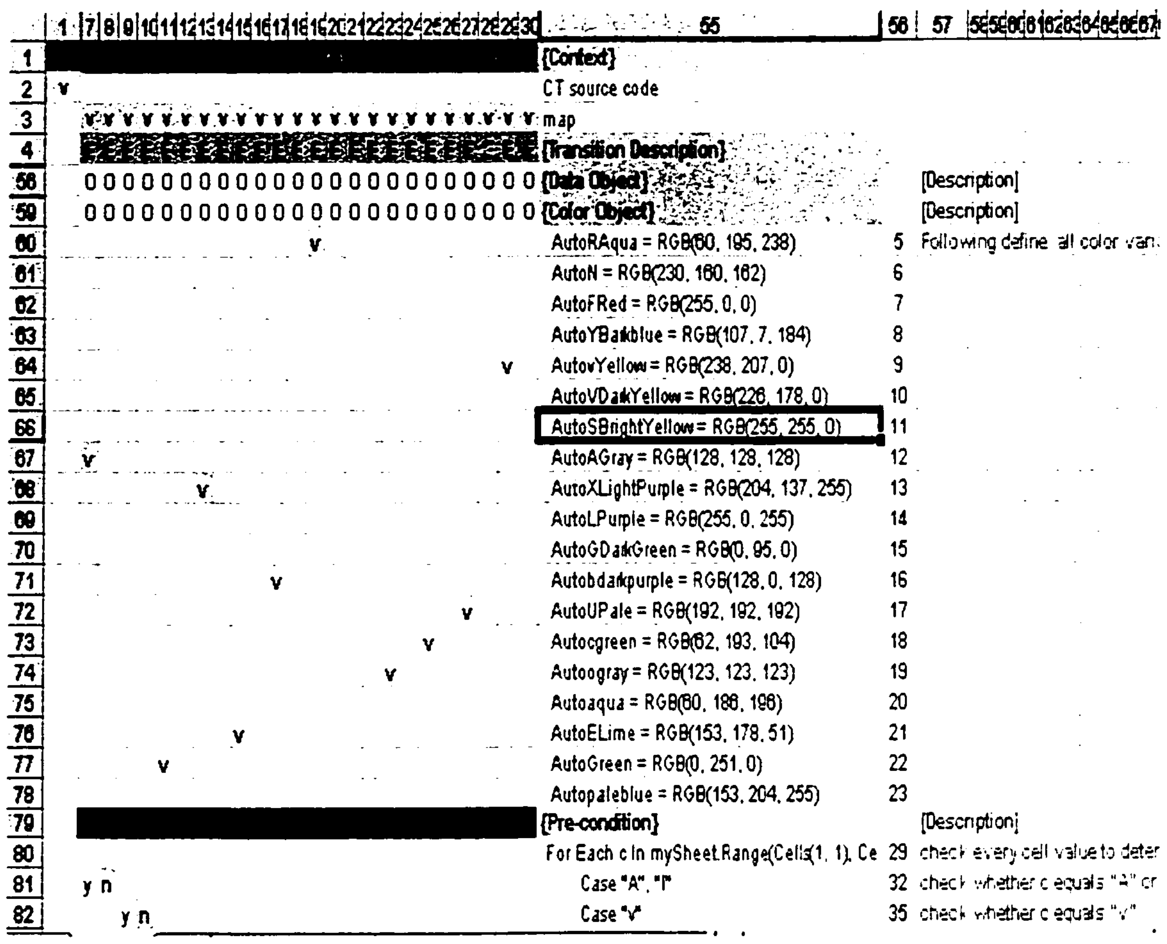
Second, "XOR" allows user to select at least two or multiple cells from different rows to generate corresponding map, which contains the exclusive not empty columns within these rows.

Third, "OR" allows user to select at least one or multiple cells from different rows to generate corresponding map which contains the each of the not empty columns from within these rows.

Finally, "NOT" allows user to select only one cell to retrieve negation columns and generate the corresponding map.

All these operations provide different ways to efficiently retrieve a small map with useful and meaningful data from a big map. Thus user can easily get the useful information or process the map.

The following, Figure3-1, is an example for showing the Context Maps.

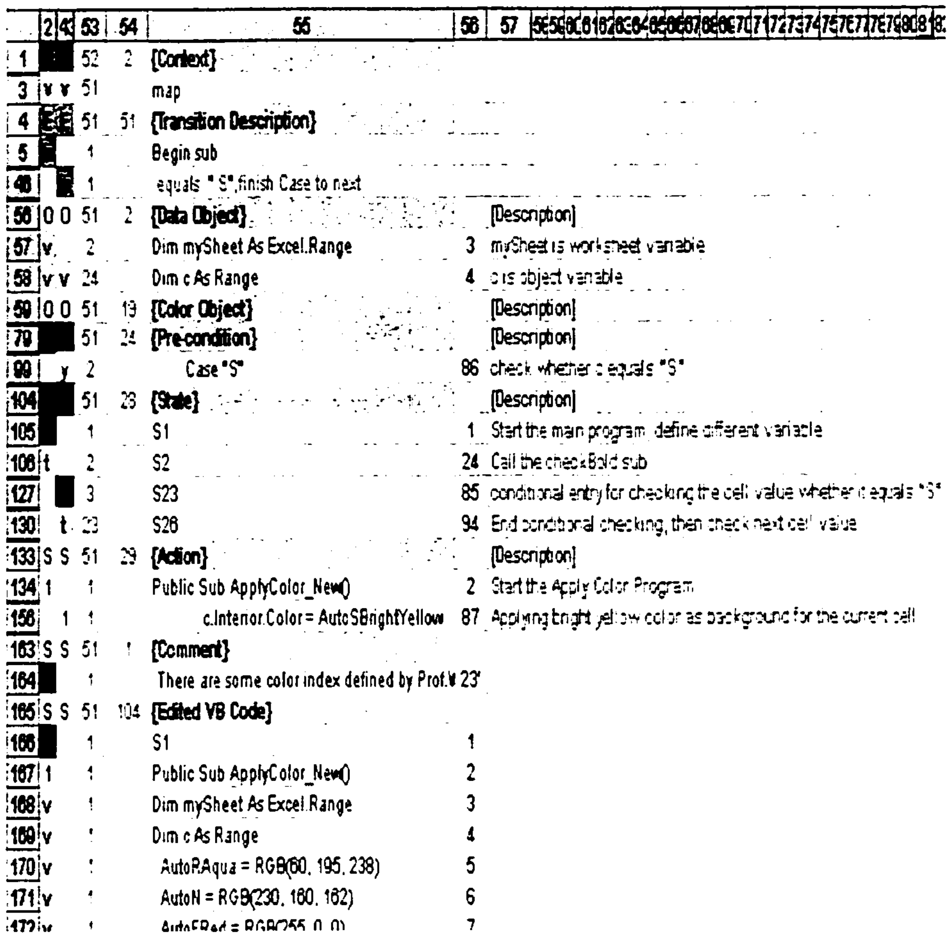


	1 7 8 9 10 … 30	55	56	57
1		{Context}		
2	v	CT source code		
3	v v v v v v v v v v v v v v v v v v v v v v v v v	map		
4		{Transition Description}		
56	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	{Data Object}		[Description]
59	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	{Color Object}		[Description]
60	v	AutoRAqua = RGB(60, 195, 238)	5	Following define all color vari
61		AutoN = RGB(230, 160, 162)	6	
62		AutoFRed = RGB(255, 0, 0)	7	
63		AutoYBarkblue = RGB(107, 7, 184)	8	
64	v	AutovYellow = RGB(238, 207, 0)	9	
65		AutoVDarkYellow = RGB(226, 178, 0)	10	
66		AutoSBrightYellow = RGB(255, 255, 0)	11	
67	v	AutoAGray = RGB(128, 128, 128)	12	
68	v	AutoXLightPurple = RGB(204, 137, 255)	13	
69		AutoLPurple = RGB(255, 0, 255)	14	
70		AutoGDarkGreen = RGB(0, 95, 0)	15	
71	v	Autobdarkpurple = RGB(128, 0, 128)	16	
72	v	AutoUPale = RGB(192, 192, 192)	17	
73	v	Autocgreen = RGB(62, 193, 104)	18	
74	v	Autoogray = RGB(123, 123, 123)	19	
75		Autoaqua = RGB(60, 188, 198)	20	
76	v	AutoELime = RGB(153, 178, 51)	21	
77	v	AutoGreen = RGB(0, 251, 0)	22	
78		Autopaleblue = RGB(153, 204, 255)	23	
79		{Pre-condition}		[Description]
80		For Each c In mySheet.Range(Cells(1, 1), Ce	29	check every cell value to deter
81	y n	Case "A", "I"	32	check whether c equals "A" or
82	y n	Case "V"	35	check whether c equals "v"

Figure3-1 Real Example of Context Maps

This Figure is a small example of Context Maps. It contains around 458 rows and 56 columns (most of them are invisible in order to display on this report). Even though it is not a very big Context Maps, the whole sheet can't be showed on the computer screen.. In order to display this, we have to group the rows and columns together. Now let's try the Query function. If the user selects cell R66:C55 and run the "OR" operation, this process can get the relevant data with "AutoSBrightYellow" variable and save the corresponding data into a small map.

The result of running "OR" on cell R66:C55 is showing below:



	2	43	53	54	55	56	57
1	■	■	52	2	{Context}		
3	v	v	51		map		
4	E	E	51	51	{Transition Description}		
5	■		1		Begin sub		
48		■	1		equals " S",finish Case to next		
56	0	0	51	2	{Data Object}		[Description]
57	v,		2		Dim mySheet As Excel.Range	3	mySheet is worksheet variable
58	v	v	24		Dim c As Range	4	c is object variable
59	0	0	51	19	{Color Object}		[Description]
79	■	■	51	24	{Pre-condition}		[Description]
99		y	2		Case "S"	86	check whether c equals "S"
104	■	■	51	23	{State}		[Description]
105	■		1		S1	1	Start the main program, define different variable
108	t		2		S2	24	Call the checkBold sub
127		■	3		S23	85	conditional entry for checking the cell value whether it equals "S"
130		t	23		S26	94	End conditional checking, then check next cell value
133	S	S	51	29	{Action}		[Description]
134	1		1		Public Sub ApplyColor_New()	2	Start the Apply Color Program
156		1	1		c.Interior.Color= AutoSBrightYellow	87	Applying bright yellow color as background for the current cell
163	S	S	51	1	{Comment}		
164	■		1		There are some color index defined by Prof.¥ 23'		
165	S	S	51	104	{Edited VB Code}		
166	■		1		S1	1	
167	1		1		Public Sub ApplyColor_New()	2	
168	v		1		Dim mySheet As Excel.Range	3	
169	v		1		Dim c As Range	4	
170	v		1		AutoRAqua = RGB(60, 195, 238)	5	
171	v		1		AutoN = RGB(230, 160, 162)	6	
172	v		1		AutoFRed = RGB(255, 0, 0)	7	

Figure3-2: The Result Maps after Running "OR" Operation

This result map only contains no more than hundred rows and 5 columns. And all data in this result map are relevant to the selected 'AutoSBrightYellow'. The useful information user will get is as following:

{Pre-Condition}

Case "S"

{Action}

c.Interior.Color=AutoSBrightYellow

"By Color" is another important function in "Query" section. It is used for querying data and generating handy map with relevant data, which can provide user pre-defined information. This is made more efficient by pre-defining the customized rule to get the corresponding map. Every active sheet should have at least one predefined query in order to run this function. The predefined query is several cells with Red, Yellow, Green and Blue as background color in one column. In the predefined query, the Red means "AND" operation, Yellow means "XOR" operation, "Green" means "OR" operations and "Blue" means "NOT" operation. It also can work with standalone "AND", "XOR", "OR" and "NOT" operation, however it has to have at least two predefined query selected.

3.3.2 Mode Selection Usage

This function is an assistant function for "Query" function. Namely, it works with "Query" function to provide user more flexible and convenient way for querying data. Under "Mode" section, it allow user to select "Visible Map", "Select Sets", "Select Roles", "Select Maps" and "Join Maps".

"Visible Maps" allows user to specify whether querying data should be taken only from the visible part of the map or from the whole map, which includes hidden rows and columns. As stated before, mostly the Context Maps are too big to show on the computer screen so that it's not easy for user to process the data. Sometimes, some parts of the data are not useful for the particular operation. Then the user can hide those parts of data and let the operation work only on visible part within a map.

"Select Sets" allows the user to choose the Sets collection for querying data from the Context Maps. Sets are necessary and important roles in the Context Maps. "Select Sets" function provides a more flexible way to let the user select any part of Sets collection for querying data. All the elements under selected Sets will take part in querying data action. Others will be hidden.

"Select Roles" provides another flexible way for the user to select useful part of map for querying data. It allows the querying data faster and more efficient. Roles are basic element to consist of Context Maps. Some roles may not useful for particular operation. The user can select only necessary roles for querying data from the Context Maps.

"Select Maps" allows the user to work on multiple maps for querying data. Excel has a restriction for spreadsheet. It only allows containing 255 columns in one spreadsheet. However, the Context Maps are mostly very big in the real world so that it's hard to construct a Context Maps within only 255 columns. In term of this situation, Context Maps allow using several sheets to consist of one map. Therefore, querying data should enable user to select multiple maps to work on. All the sheets in the current active workbook are available for selection. And user should specify queried data in each selected maps before implementation.

"Join Maps" enable merging of multiple maps and saving the result into a new map. Querying will generate a small map with relevant data so that "Join Maps" can merge couple of these maps. It is a bit different from other operations in this section because it can work with either "Select Maps" or by itself. This operation has nothing to do with querying data. When it works with "Select Maps", it merges all the result maps from "Select Maps". Otherwise, the user has to select maps for merging. For merging maps, it compares the Set values and Set member values for all selected maps. If they are the same, it conjuncts all the roles from the multiple maps and saves the result into a new map. If they are not the same, it adds additional rows in relevant position of the new map.

3.3.3 Apply Color Usage

Context Maps consist of Sets, Set Members, Set Roles, Set Members Roles, Cardinality of Roles, and Cardinality of Set Members etc. different elements. This functionality provides automatic way to applying different background color based on the elements usage and value. This gives the user a clear vision of Context Maps and enable user easily and efficiently find useful information from the big map.

3.3.4 Context Map Help

In this application, there are two versions of "Help" files. One is Excel version that can be triggered from Query Form. The other is MS Word documentation of user manual. The Excel version supplys brief description for "QUERY" functions. The user manual provides a detailed description about how to use this program and give some examples for some functions.

3.4 Practical Applications

Using "CONTEXT+" tool in applications is more convenient for users to compare the similarities and differences in processing Context Maps. The powerful tool "CONTEXT+" can retrieve and express the relationship of original materials clearly. By simply selecting the specific components, "CONTEXT+" tool can analyze the relationships of each component within the process. In Context Maps representation, users can easily plug in another relationship or diagram notation in one map, even different kind of processes for comparison. "CONTEXT+" tool also supports these operations.

During the development stage, some practical applications were used as tests to measure flexibility and performance of the CONTEXT+; the most relevant ones are listed here.

- "DUE DILIGENCE STRATEGY WITH CONTEXT MAPS" major report by XU GUO ZHU.
- "KNOWLEDGE MINING WITH CONTEXT MAPS, CONVERTING OPEN PROCESS METHODOLOGY TO A 3P-ABLE FORMAT" major report by JING BAI.
- Self-consistency test with the examples in this report and part of source code.

Other applications will be part of the manual for the facilities or be part of other examples available to users.

Chapter 4

CONTEXT+: Installation

4.1 Required Files

The goal of this project is to build Context Maps tool under 3p-able format, which will allow users to deal with Context Maps very efficient. The program must run on computers equipped with MS Excel. The central element in the process of information manipulation is based on the Context Maps formal notation technology.

The fileContextMaps.xls contains some samples of input and output sheets to demonstrate the functionality of CONTEXT+. In order to use the Context Maps Tool, first of all, this file has to be opened and the user has to click enable macro message box, then the user can either open a new workbook or insert new sheet to work.

The CONTEXT+ menu buttons are pretty straightforward, and the dialogs will guide users through information-entry steps. If user needs any help to run the program, there are two versions of "Help" file. One is short version of Excel file, user can click Help button from Query Form. Another one is MS Word documentation of user manual, which will be included in this report.

4.2 Installation

Any system must run Microsoft Excel 97 (Office 97) or higher under Microsoft Windows operating system. The Programming environment is the Visual Basic Editor that comes with Excel. No stand-alone VB environment is needed.

The Context Maps file could be placed into any folder. For better file organization, it is suggested that a Context Maps folder can be created and all related files can be placed in this folder.

4.3 Development Tool

- **Excel**

  Microsoft Excel is a spreadsheet program that has been widely used for data Storage, data calculation, graphic or chart display and decision-making etc.

  The three major parts of Excel are:

1) Worksheets (which means the same as spreadsheet) that allow user to better calculate, manipulate, and analyze data such as numbers and text.
2) Charts, which pictorially represent data. Excel can draw a variety of two-dimensional and three-dimensional charts.
3) Databases, which manage data. For example, once users enter that data, they can search for specific data, and select data that meet the criteria.

- **Visual Basic**

  A Visual Basic Application can provide us with the means to accomplish a wide range of the programmatic functionalities. With VBA, we can create full-fledged custom applications in Microsoft Excel.

Visual Basic supports a set of objects that correspond directly to elements in Microsoft Excel. An object in Visual Basic can represent every element in Microsoft Excel, such as workbook, worksheet, range, cell, and so on. By creating procedures that control these objects we can automate tasks in Microsoft Excel.

4.4 Code Design

This program is written in Visual Basic. The source code of approximately 3000 lines is stored as different .frm, .bas and .cls files and is open to the public. The code can be viewed in the Visual Basic Editor in Excel.

The whole source code was grouped by 4 modules, 3 forms and 1 class. The modules and forms are organized and named by functionality. All modules are named as "Mxxxx". All forms are named as "frmxxxx". All classes are named as "Cxxxx".

Interested readers can be able look at individual modules and forms once they have access to this program. With Professor W.M. Jaworski's approval, the source code can be freely copied or modified in order that users can customize their own needs and add new functions.

4.5 General Constraints

This program is constrained only to run on Microsoft Windows operating system: Win 95/98, Win 2000 and Win NT. And also Microsoft Excel 97 or above is required to be installed. The user needs to know some basic function of Microsoft Excel.

Chapter 5

CONTEXT+: ContextMaps Documentation

Context Maps is a powerful method for representing systems architecture, structures and processes. Context Maps can incorporate instances, concepts, roles, knowledge tuples and views. It has simple semantics, which generate different views of the underlying knowledge for users. By using the Context Maps models technology, the information structure is rewritten from narratives into a knowledge frame, and create schema view of the Context Maps model. And CONTEXT+ tool gives user more convenient to retrieve and extract the relevant information from the mass and complicated diagrams.

The code written for this program consists of three Form interface, four Modules and one Class. The detail source code list will be found in the appendix of this paper.

The structure for the code is:

A. Form Interface

A.1 frmProjection: this Form is used as QUERY interface for user to select main operations. It will validate and save the user's selection.

A.2 frmSelectConcepts: this Form is used to display all the Concept Sets from the current active ContextMap, and save the selected Concept Sets in a collection.

A.3 frmSelectWorksheets: this Form is used to display all the Worksheets from the current active Workbook and save the selected Worksheets in a collection.

B. Modules

B.1 mApplyColor: this module has two subroutines which are ApplyColor and CheckBold. The ApplyColor is used for applying color based on the different value on the map. The CheckBold is used for changing the font to bold for the whole rows which are Concept Sets and set the gray as background color for the Concept Sets.

B.2 mAutoByColor: this module has eight subroutines. StartMultiQuery is used for validating the selection by calling ValBycolorParameter and implements the "AND", "XOR", "OR", "NOT' for predefined queries by calling AutoQueryByColor. PromptWorkSheets is used for getting all the worksheets from current active Workbook for user to select. SPAndHideCols, SPXorHideCols, SPOrideCols and SPNotHideCols are called by AutoQueryByColor for predefined queries.

B.3 mMerge: this module has three subroutines as its major parts, which are Merge, JoinMaps and findConceptSet. FindConceptSet is used for getting all the Concept Sets value and their row number from the map. Merge is used to add a new sheet and rename it as "Merge Result". JoinMaps is used to merge multiple selected sheets.

B.4 mProjections: this module is the main part of whole project. It has 16 subroutines, and the key subroutine is ProjectjMap. This subroutine is used for checking the selection from the Query Form and calling the different subs or functions to implement the corresponding operations. The rest subroutines are: InstanceBasedQueryProject, InteractiveProjection, QIBHideRows, QIBHideCols, QuerySPAndHideCols, QuerySPXorHideCols, QuerySPOrHideCols, QuerySPNotHideCols, ShowAll, SPHideRows, SPAndHideCols, SPOrHideCols, TestSPXorHideCols, TestSPNotHideCols, ValidateProjectionParameters. Mostly they all called by ProjectjMap. The relationship between them can be found through the ContextMap which is converted by the source code of this module.

C. Classes:

C.1 CsuspendVisualInterface: this class is used for creating the instance of the application and setting or getting the Timer or Status property for the application.

Context Maps can be used for representing architectures, structures, and relationship for information systems. It can also clearly demonstrate the consistency and completeness of the VB code and CONTEXT+ user manual. In order to illustrate the usage of ContextMap and implementation of the source code in this chapter, an efficient means will be indroduced to represent the source code and user manual: ContextMaps format decumentation. By this way, users can be able to understand the usefulness of ContextMaps and comprehend the workflow of source code and user manual.

The work of conversion is pretty hard and challenging. As everybody knows, constructing the schema of the ContextMap is critical and the most difficult part in the heavy mapping field. Professor W.M. Jaworski provides a good schema as table 1 (Jaworski [2002]) and instructions so that all these processes can be preformed smoothly.

And also a group of students help me to convert part of the maps and verify the maps. It contains 53 ContextMaps for VB source code and 19 ContextMaps for CONTEXT+ user manul. All these maps are very important in this report; however, all of them can't be showed here because of paper and space limitations. Here, only the map schema, top map and some small views will be displayed. The rest of them can be viewed on the website: http://groups.yahoo.com/group/ContextMaps/files/VBtoMAPS/.

■	■	■	■	2	2	**{Context}**
v				1		Original source code
	v	v	v	1		map
	v					<map view name>
		v				<map view name>
			v			<map view name>
	S	S	S	1	0	**{Map/code Purpose Description}**
	O	O	O	1	0	**{Data Objects}**
	E	E	E	1	0	**{Transition Description}**
	G	G	G	1	0	**{Pre-conditions}**
	■	■	■	1	0	**{States}**
	S	S	S	1	0	**{SubRoutines/Sub-Maps}**
	S	S	S	1	0	**{Actions}**
	U	U	U	1	0	**{Post-conditions}**
	S	S	S	1	0	**{Edited Source Code Statements}**
S	S	S	S	2	0	**{Source Code Statements}**
■	■	■	■	2	1	**{Content Source}**
■	■	■	■	2	1	**{Author}**
v	v	v	v	2		Syntax and Patterns © by W.M. Jaworski, 1988-2002
v	v	v	v			Map © by <map developer name>
v	v	v	v			Map verification © by <map verifier name>

Table 1: ContextMap Schema for mapping VB code

■	■	■	■	■	■	■	■	■	■	■	■	■	#	9	**{Context}**
v	v	v	v	v	v	v	v	v					#		Contains
									v	v	v	v	#		Executes

v													#		Fprojection
	v												#		frmSelectConcepts
		v										v	#		frmSelectWorksheet
			v						v				3		MApplyColor
				v									9		MAutoQueryByColor
					v	v	v			v			6		MMerge
								v			v		#		Mprojections
X	X	X	X	X	X	X	X	X	X	X	X	X	#	2	**{Edge Types}**
X	X	X	X	X	X	X	X	X					#		<<includes>>
									X	X	X	X	#		<<calls>>
													#	7	**{Entity Types}**
F		F											#	#	**{Public Property}**
			F	F					F	F	F		#	7	**{Public Sub}**
	F			F	F			F				F	#	#	**{Private Sub}**
													2	2	**{Public Function}**
													4	4	**{Private Function}**
						F	F						4	4	**{Old-Version Sub}**
													#	1	**{Content Source}**
v	v	v	v	v	v	v	v	v	v	v	v	v	#		Zhou Kang
													#	3	**{Author}**
v	v	v	v	v	v	v	v	v	v	v	v	v	#		Syntax and Patterns © by W.M. Jaworski, 1988-2002
v	v	v	v	v	v	v	v	v	v	v	v	v	#		Map © by W.M. Jaworski
v	v	v	v	v	v	v	v	v	v	v	v	v	#		Editing by WMJ

Table 2: Top-Map Schema and part of map

						6	4	**{Context}**
v	v	v	v	v	v	6		Public Sub Merge(joinSheets As Collection)
v						1		CT source code
	v					1		Private Sub JoinMaps(joinSheets)
		v	v	v	v			map
S		S	S	S	S	5	2	**{Map/code Purpose Description}**
v						1		The code is used for adding new sheet named "Merge Result" which is used to save the result of merging multiple maps
		v	v	v	v	4		The whole code divided into 4 States in map
		E	E	E	E	4	4	**{Transition Description}**
		m				1		Begin sub
			m			1		Browse to check each worksheets name under current active workbook

				m		1		Check whether the current worksheet exceeds the max count of the all the worksheets.Check if there's any worksheet named "Merge Result" already
					▨	1		If there is no worksheet named "Merge Result" in current active workbook, add a new sheet named the "Merge Result"
		O	O	O	O	4	5	**{Data Object}**
		v				1		Dim WS_Count As Integer
		v		u		2		Dim I As Integer
		v			u	2		Dim flgNewSheet As Boolean
			▨		▨	2		Dim wks As Worksheet
			u		u	2		Dim newWks As New Worksheet
		G	G	G	G	4	2	**{Pre-condition}**
			n	y	y	3		For I = 1 To ActiveWorkbook.Worksheets.count
				n	y	2		If ActiveWorkbook.Worksheets(I).Name = "Merge Result" Then
		■	■	■	■	4	3	**{State}**
		■				1		**Start Program**
		t	■	■	■	4		If "Merge Result" sheet already exists, delete it.Create a new sheet called "Merge Result"
			t		t	2		Call sub to start merging maps,End this program
	E	S	S	S	S	5	1	**{SubRoutines/Sub-Maps}**
	■		#		#	3		Call JoinMaps(joinSheets)
	N	S	S	S	S	5	17	**{Action}**
		1				1		Public Sub Merge(joinSheets As Collection)
		2				1		flgNewSheet = flase
					1	1		sheets("Merge Result").Select
					2	1		Application.CutCopyMode = False
					3	1		Application.DisplayAlerts = False
					4	1		ActiveWindow.SelectedSheets.Delete
					5	1		Exit For
				1		1		Next I
			1		6	2		sheets(joinSheets(1)).Select
			2		7	2		Set wks = ActiveSheet
			3		8	2		wks.UsedRange.SpecialCells(xlCellTypeVisible).Copy
			4		9	2		Set newWks = ActiveWorkbook.Worksheets.Add(after:=Worksheets(Worksheets.count))
			5		#	2		newWks.Name = "Merge Result"
			6		#	2		With newWks
			7		#	2		.Paste
			8		#	2		End With
	v		#		#	3		Call JoinMaps(joinSheets)

		S	S	S	S	4	1	**{Comment}**
			■		■	2		'Call JoinMaps
	N	S	S	S	S	5	29	**{Edited VB Code}**
S		S	S	S	S	5	29	**{VB Code}**
■	■	■	■	■	■	5	1	**{Content Source}**
v	v	v	v	v	v	5		Zhou Kang
■	■	■	■	■	■	5	33	**{Author}**
v	v	v	v	v	v	5		Syntax and Patterns © by W.M. Jaworski, 1988-2002
v	v	v	v	v	v	5		Map © by Zhou Kang
v	v	v	v	v	v	5		Map verification © by Li Xuan
v	v	v	v	v	v	6		**Map CORRECTION BY WMJ**

Table 3: ContextMap of Code Sub Merge

■	■	■	■	■	■	■	■	■	9	5	**{Context}**
v	v	v	v	v	v	v	v	v	9		Public Sub InteractiveProjection(ByRef wks As Worksheet, ByRef objSelected As Object)
v									1		CT source code
	v								1		properies(Fproject)
		v							1		**Public Function ProjectjMap(ByRef wks As Worksheet, ByRef obj As Object, Optional ByVal SaveView As Boolean = False, Optional Optional ByVal CheckVisibles As Boolean = False, Optional ByVal QueryColors As Boolean = False, Optional ByVal AutoByColors As ByVal PasteToNewSheet As Boolean = False, Optional ByVal PromptForConcepts As Boolean = False, _Optional ByVal JoinMaps As Boolean = False, Optional ByVal getCardinality As Boolean = False, Optional ByVal ApplyingColor As Optional ByVal ProjectionType As Long = pjtNORMAL, Optional ByVal SuppressUserInteraction As Boolean = False) As Boolean**
			v	v	v	v	v	v	6		map
			E	E	E	E	E	E	6	3	**{Transition Description}**
			m						1		go to the the program
				m					1		there is error system is resumed
							m		1		Execute the if clause actions
			O	O	O	O	O	O	6	1	**{Data Object}**
			v						1		Dim frmProjectionOptions As FProjection
	N		G	G	G	G	G	G	7	2	**{Pre-condition}**
				y	n				2		On Error Resume Next
	v					n	y		3		If Not .Canceled Then
			■	■	■	■	■	■	6	5	**{State}**
			■						1		begin the function
			t	■					3		if no error continue below functions
					t	■	■		3		if user don't want to cancel the execution then continue below functions
						t	t	■	3		the end of the function

				t				t	2		Resume the execution of the functions
	F	F	S	S	S	S	S	S	8	2	**{SubRoutines/Sub-Maps}**
		■					1		2		ProjectjMap ActiveSheet, Selection, .SaveView, .NewSheet, .PromptForConc .CheckVisible, .QueryColor, .AutoByColor, .SelectMap, .JoinMap, .Projecti onType
	■								1		ProjectjMap ActiveSheet, Selection, .SaveView, .NewSheet, .PromptForConc .CheckVisible, .QueryColor, .AutoByColor, .SelectMap, .JoinMap, .Projecti onType
	N	N	S	S	S	S	S	S	8	7	**{Action}**
			1						1		Public Sub InteractiveProjection(ByRef wks As Worksheet, ByRef objSelected As Object)
					1				1		Set frmProjectionOptions = New FProjection
					2				1		Load frmProjectionOptions
					3				1		With frmProjectionOptions
					4				1		.Show
	v	v					1		3		ProjectjMap ActiveSheet, Selection, .SaveView, .NewSheet, .PromptForConc .CheckVisible, .QueryColor, .AutoByColor, .SelectMap, .JoinMap, .Projecti onType
				1				1	2		Resume Next
			S	S	S	S	S	S	6	0	**{Comment}**
	N	N	S	S	S	S	S	S	8	19	**{Edited VB Code}**
S			S	S	S	S	S	S	7	15	**{VB Code}**
■	■	■	■	■	■	■	■	■	9	1	**{Content Source}**
v	v	v	v	v	v	v	v	v	9		Zhou Kang
■	■	■	■	■	■	■	■	■	9	3	**{Author}**
v	v	v	v	v	v	v	v	v	9		Syntax and Patterns © by W.M. Jaworski, 1988-2002
v	v	v	v	v	v	v	v	v	9		Map © by Minghua Chen
v	v	v	v	v	v	v	v	v	9		Map verification © by Chen, Yi
v	v	v	v	v	v	v	v	v	10		Map correction © by Zhou kang

Table 4: ContextMap of Code Sub InteractiveProjection

■	■	■	■	■	■	■	■	■	9	4	**{Context}**
v	v	v	v	v	v	v	v	v	9		Public Function PromptWorkSheets(ByRef colSelectedSheets As Collection) As Boolean
v									1		CT source code
	v								1		properties(SelectWorksheet)
		v	v	v	v	v	v	v	7		map
S		S	S	S	S	S	S	S	8	2	**{Map/code Purpose Description}**
v									1		**This map is used to represent the source code for selecting all the active worksheets in the current workbook**
		v	v	v	v	v	v	v	7		
		E	E	E	E	E	E	E	7	7	**{Transition Description}**
		O	O	O	O	O	O	O	7	2	**{Data Object}**
		v				u	r		3		Dim nSheetIndex As Long

		v		u		r	r		4		Dim colAllSheets As Collection
	N	G	G	G	G	G	G	G	8	3	**{Pre-condition}**
			y	n					2		On Error Resume Next
					n	y			2		For nSheetIndex = 1 To .sheets.count
	v						n	y	3		If Not SelectWorkSheet(frmSelectWorksheet.listSheets, frmSelectWorksheet.canceled, frmSelectWorksheet.SelectedSheets) T
		■	■	■	■	■	■	■	7	6	**{State}**
		■							1		**Begin function and define the variable used in this program**
		t	■	■					3		Initialize some variable
				t	■	■			3		Browse every worksheets in the current workbook, add their name to collection
					t		■	■	3		Get the selected maps from the available maps list
							t	t	2		End Function
			t						1		Resume Next
	F	S	S	S	S	S	S	S	8	1	**{SubRoutines/Sub-Maps}**
	■						v	v	3		If Not SelectWorkSheet(frmSelectWorksheet.listSheets, frmSelectWorksheet.canceled, frmSelectWorksheet.SelectedSheets) Then
		S	S	S	S	S	S	S	7	8	**{Action}**
		1							1		Public Function PromptWorkSheets(ByRef colSelectedSheets As Collection) As Boolean
				1					1		Set colAllSheets = New Collection
				2					1		With ActiveWorkbook
						1			1		colAllSheets.Add (.sheets.item(nSheetIndex).Name)
						2			1		Next nSheetIndex
					1				1		End With
								1	1		Exit Function
							1		1		PromptWorkSheets = True
		S	S	S	S	S	S	S	7	0	**{Comment}**
	N	S	S	S	S	S	S	S	8	20	**{Edited VB Code}**
S		S	S	S	S	S	S	S	8	15	**{VB Code}**
■	■	■	■	■	■	■	■	■	9	1	**{Content Source}**
v	v	v	v	v	v	v	v	v	9		Zhou Kang
■	■	■	■	■	■	■	■	■	9	3	**{Author}**
v	v	v	v	v	v	v	v	v	9		**Syntax and Patterns © by W.M. Jaworski, 1988-2002**
v	v	v	v	v	v	v	v	v	9		Map © by ZhouKang
v	v	v	v	v	v	v	v	v	9		Map verification © by Yang Shentian

Table 5: ContextMap of Code Sub PromptWorkSheets

■	■	■	■	■	■	■	■	8	2	**{Context}**
v	v	v	v	v	v	v	v			Private Sub CheckBold()
v								1		CT source code
	v	v	v	v	v	v	v	7		map
■	■	■	■	■	■	■	■	8	2	**{Map/code Purpose Description}**

v								1		This map is used to represent for checking concept set for the map
	v	v	v	v	v	v	v	7		
	E	E	E	E	E	E	E	7	5	**{Transition Description}**
	O	O	O	O	O	O	O	7	6	**{Data Object}**
	v						v	2		`Dim wks As Worksheet`
	v							1		`Dim rng As Range`
	v							1		`Dim RowCount As Long`
	v							1		`Dim ColCount As Long`
	v				v		v	3		`Dim rowIndex As Long`
	v					v		2		`Dim colIndex As Long`
	G	G	G	G	G	G	G	7	3	**{Pre-condition}**
			n	y				3		`For rowIndex = 1 To RowCount`
					n	y	y	3		`For colIndex = ColCount To 1 Step -1`
						n	y	2		`If IsConceptLabel(wks.Cells(rowIndex, colIndex).Value) Then`
	■	■	■	■	■	■	■	7	5	**{State}**
	■							1		`S1'`
	t	■						2		`S2'`
		t	■	■	t		t	5		`S3'`
				t	■	■	■	4		`S4'`
			t					1		`End Sub`
	S	S	S	S	S	S	S	7	10	**{Action}**
	1							1		`Private Sub CheckBold()`
	2							1		`Set wks = ActiveSheet`
	3							1		`With wks.Cells.SpecialCells(xlLastCell)`
	4							1		`RowCount = .Row`
	5							1		`ColCount = .Column`
		1						1		`End With`
							1	1		`wks.rows(rowIndex).Font.Bold = True`
							2	1		`Exit For`
						1		1		`Next colIndex`
					1		3	2		`Next rowIndex`
	S	S	S	S	S	S	S	7	1	**{Comment}**
	S	S	S	S	S	S	S	7	26	**{Edited VB Code}**
S	S	S	S	S	S	S	S	8	22	**{VB Code}**
■	■	■	■	■	■	■	■	8	1	**{Content Source}**
v	v	v	v	v	v	v	v	8		Zhou Kang
■	■	■	■	■	■	■	■	8	1	**{Author}**
v	v	v	v	v	v	v	v	8		Syntax and Patterns © by W.M. Jaworski, 1988-2002
v	v	v	v	v	v	v	v			Map © by Zhou Kang
v	v	v	v	v	v	v	v			**Map CORRECTED BY WMJ**

Table 6: ContextMap of Code Sub CheckBold

				12	1	**{cTuple Id}**		
9	#	#	#	12		id		
				12	3	**{Context}**		Prompt the Query Form
v	v	v	v	12		OO Templates		
				2		Processes		Perform ApplyColor Function
v	v	v	v	10		Products		
A	A	A	A	12	12	**{OO Template}**		Get the Help
				1		Project Process		
				1		Phase Process		
			v	1		ID: Interaction Diagram		
X	X	X	X	12	0	**{cTuple Type}**		

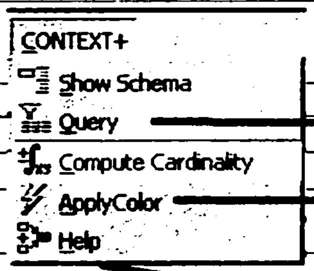


Table 7: ContextMap of User Manual "CONTEXT+ Toolbar Menu"

					12	1	**{cTuple Id}**		
8	9	10	11	12	12		id		
					12	3	**{Context}**		
v									
v									Display only Visible Maps
A									
									Perform Predefined Query
									Perform Select Sets
									Perform ApplyColor Function
v									Perform Select Maps
	v				1		CS: Class Specification		
		v			1		CCCD: Class-Centric Class Diagram		Perform JoinMaps Using:
			v		1		STD: State Transition		
				v	1		ID: Interaction Diagram		
X	X	X	X	X	12	0	**{cTuple Type}**		Perform Four Operations
M	M	M	M	M	12	0	**{Note}**		
			E	E	6	0	**{Event}**		
					1	0	**{Phase Transition Criteria}**		
					1	0	**{Review Criteria}**		
A	A	A	A	A	12	0	**{Phase/State}**		

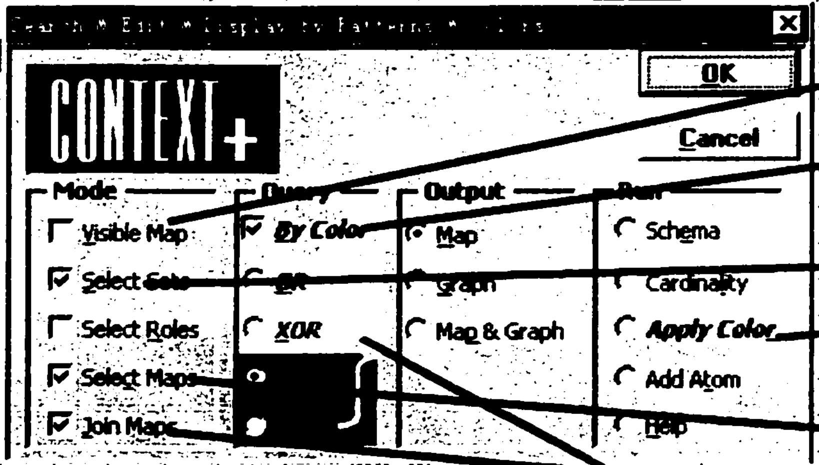


Table 8: ContextMap of User Manual "CONTEXT+ Query Form"

Chapter 6

Conclusion and Recommendation

6.1 General Conclusions

After studying this notation technology, many advantages of the Context Maps are displayed. The following conclusions are drawn from the study of this major report:

1. Context maps enable us to create virtual information maps for the knowledge-based system. Context Maps are a notational technology for representing systems architecture, structures, processes and reusable templates. The Context Maps notation allows easy recovery and modeling of generic schema for processes, objects and views of information systems.
2. Although Context maps already have a set of predefined notation, they also allow the user to create their extendable notations to meet new needs. In the meantime, there are always many different ways of representing the same knowledge. This property shows the flexibility of Context maps
3. Context Maps syntax is simple and robust. Context Maps models are 3P-able format, allow users to specify, query and control the model views. Different views are generated logically to be useful for compilers or end users.
4. By applying the CONTEXT+ tool, Contexts Maps are more efficient and convenient for users to retrieve and compare relevant information from the mass and complicated diagrams.

5. Using advanced query in CONTEXT+query tools, query on the high-level design schema is becoming possible. And directly querying the data allows query more easy and visible.
6. If the data of Context Maps reach the limit of each sheet in Excel, it can be stored in another sheet to extend size of Context Maps. And the CONTEXT+tool has such capability to process multiple sheets.
7. The CONTEXT+ program can also be treated as standard knowledge based information to be converted into Context Maps. The structure of CONTEXT+ program becomes normalized.

6.2 Recommendations for Future Works

After studying Context Maps technology, further work is required to improve this technology and application program. The following are recommended for the future improvement:

1. When processing a large Context Maps sheet, it takes much time to get results by using the query tools. It is necessary to modify the program of CONTEXT+ tools to speed up the query.
2. Some functions of CONTEXT+ will be developed or improved. In output section, Output Graph will be done. This feature will be very important for Context Maps.
3. The tools of Context Maps should be expensed to be more convenient and intelligent to support the 3P-able format. Design a more friendly user interface needs to be done in the future.

4. Inputting the data in the map is very boring and there are some redundancies existing. Provide a guideline for designing the schema, composing the map and inputting the data.
5. Transferring the knowledge to the map is still done by manually. Maybe there are some automatic transferring tool can be developed in the near future.
6. Refine the ambiguity concepts, elements or set members within Context Map; then, building a kind of reliable criteria guides further work.

Bibliography

A. Printed Materials

1) **Wojciech M. Jaworski**, "Comp 457/657 Course Notes", Concordia University, 2000.

2) **Grady Booch, James Rumbaugh, Ivar Jacobson**, "The UML User's Guide", Addison Wesley, 1998.

3) **Wojciech M. Jaworski**,"Conceptual Spreadsheets for Data and Knowledge", Warehousing, University of New Hampshire, Durham, May 31 - June 1, 1995

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

5) **Wojciech M. Jaworski** and "Michailidis A. A., Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", NATW94, University of Massachusetts - Lowell, Massachusetts, June 1994.

6) **Wojciech M. Jaworski,** "Conceptual Spreadsheets for Data and Knowledge Warehousing", ATW95 - USA 1995, University of New Hampshire, Durham, New Hampshire, May 31 - June 1, 1995.

7) **Wojciech M. Jaworski,** "Cooperative Engineering Issues by Examples: Mapping of Mil498 and NSDIR with *Context Maps*", ATW96-USA *1996*, Electronic Systems Center, Hanscom Air Force Base, August 6-9, 1996.

8) **Wojciech M. Jaworski,** "Representing Processes, Schemata and Templates with *Context Maps*", 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.

9) **Wojciech M. Jaworski, Michailidis A. A.**, "Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", ATW '94, University of Massachusetts - Lowell, Lowell, Massachusetts, June 1994.

10) **Wojciech M. Jaworski,** "InfoMaps: Conceptual Spreadsheets for Data and Knowledge Warehousing", ATW '95, University of New Hampshire, Durham, New Hampshire, June 1995.

11) **Wojciech M. Jaworski**, et al. "The ABL/W4 methodology for system modeling", System Research Journal 4(1), 23-37, 1987.

12) **Wojciech M. Jaworski**, et al. "Representing processes, schemata and templates with *Context Maps*", Semiotica 125(1/3), 229-47, 1999.

13) **Minghui Han,** "Associative Data Model And *Context Maps*", Major Report, Concordia University, 2001.

14) **Ian Sommerville**, "Software Engineering", Addison-Wesley, 5th edition, 1995.

B. Online Resources

1) **General Strategies Inc,** http://www.gen-strategies.com

2) **Context Maps,**

http://jan.ucc.nau.edu/~jwb2/research/ContextMaps/ContextMaps.html

3) **IHMC Concept Map Software,**

http://cmap.coginst.uwf.edu/download/cmapChoice.mgi

1) **Rational Unified Process,** http://www.rational.com

2) **Concordia University,** Thesis preparation and thesis examination regulations,

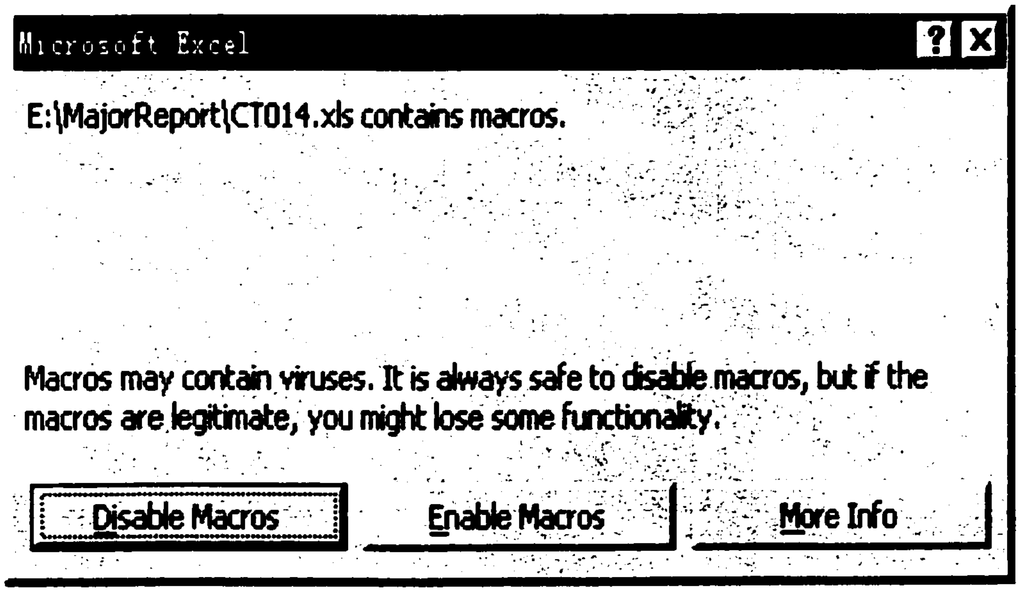
http://www-gradstudies.concordia.ca/SGS_WWW/publications.html

Appendix A---- CONTEXT+: User Manual

The CONTEXT+ application is written using VB language from Microsoft Office technologies, allowing it to run on the Microsoft platform. Make sure you check the system requirements (please refer to Chapter 4.2 Installation) before you run the application. The following will walk you through how to use this tool.

A.1 Start Program

1. Opening of the ContextTools.xls file will show the following popup dialog:



Microsoft Excel

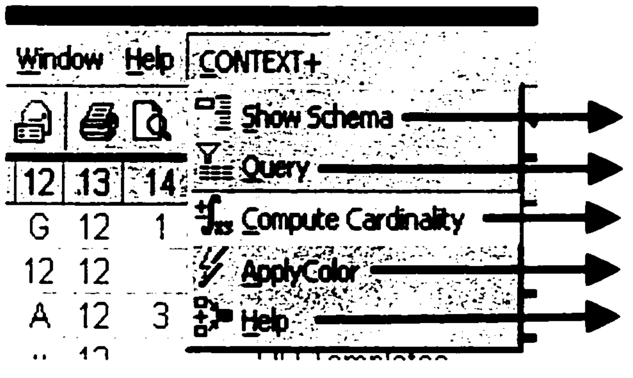
E:\MajorReport\CT014.xls contains macros.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

Figure A-1: MS Excel Standard Dialog Box

2. Clicking of the Enable Macros button will launch the tool and display new menu named CONTEXT+ in EXCEL toolbar.

3. As shown in Figure A-2, five sub-menus appear when user clicks the CONTEXT+.



1. Outout is oroduced as Zoom-in MAP
2. Display Query Form to process functions
3. MAP Cardinality is computed
4. Apply relevant colors to the Map ROLES
5. Display Help Spreadsheet with brief Description

Figure A-2: Sub-Menus of the CONTEXT+ Menu.

Note: some of sub- memus functions will also appear on the RUN part of the Query form shown in Figure A-8.

A.2 CONTEXT+ Tool Functionality

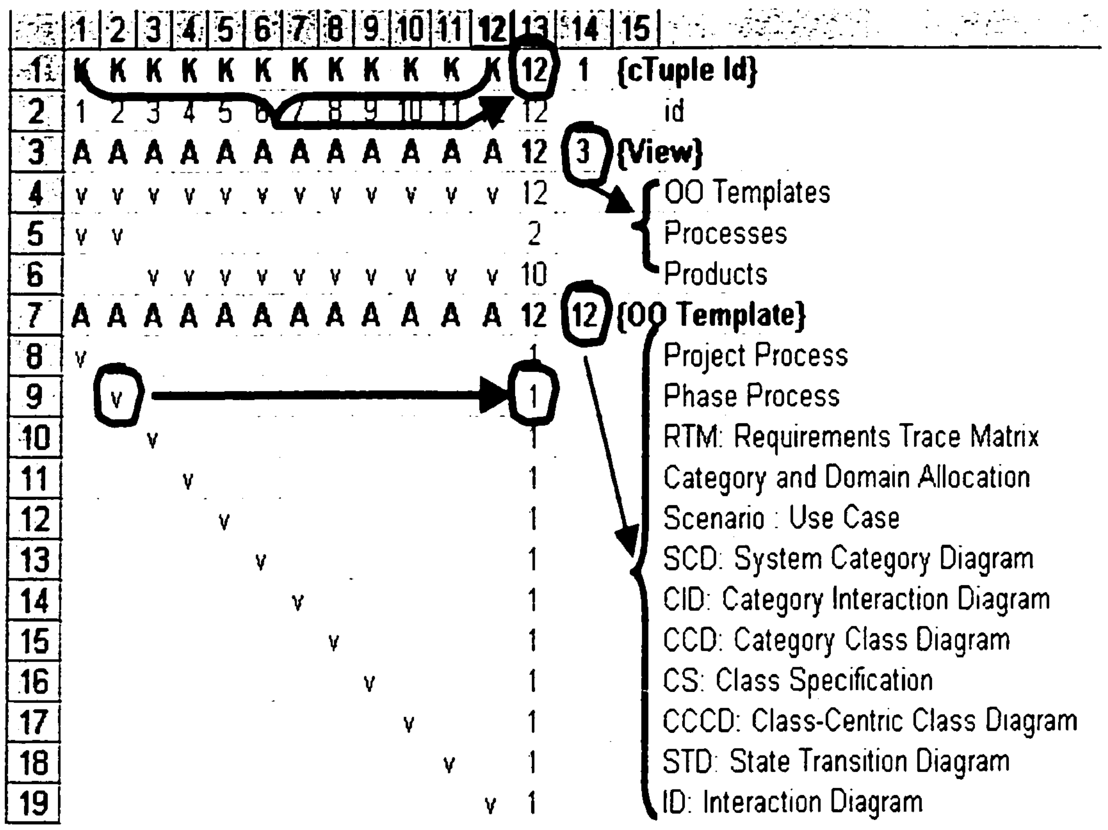
A.2.1 Show Schema

Show Schema — This button will produce Zoom-in Map, which is a work frame of Context Map.

A.2.2 Computer Cardinality

1. Using this function, the criteria for this button is there should be two columns reserved or added just preceding the Concept column for computer cardinality.
2. In Context Map, the right column added is used to compute the number of not empty roles in this row and left one is used to compute the number of sub-concept value under this concept.

3. After the option button Cardinality is selected from Query Form (Figure A-8) or Compute Cardinality is clicked from sub-menu bar in Context+, the program will implement computing function to position the results at these two columns.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	K	K	K	K	K	K	K	K	K	K	K	K	12	1	{cTuple Id}
2	1	2	3	4	5	6	7	8	9	10	11		12		id
3	A	A	A	A	A	A	A	A	A	A	A	A	12	3	{View}
4	v	v	v	v	v	v	v	v	v	v	v	v	12		OO Templates
5	v	v											2		Processes
6			v	v	v	v	v	v	v	v	v	v	10		Products
7	A	A	A	A	A	A	A	A	A	A	A	A	12	12	{OO Template}
8	v												1		Project Process
9		v											1		Phase Process
10			v										1		RTM: Requirements Trace Matrix
11				v									1		Category and Domain Allocation
12					v								1		Scenario : Use Case
13						v							1		SCD: System Category Diagram
14							v						1		CID: Category Interaction Diagram
15								v					1		CCD: Category Class Diagram
16									v				1		CS: Class Specification
17										v			1		CCCD: Class-Centric Class Diagram
18											v		1		STD: State Transition Diagram
19												v	1		ID: Interaction Diagram

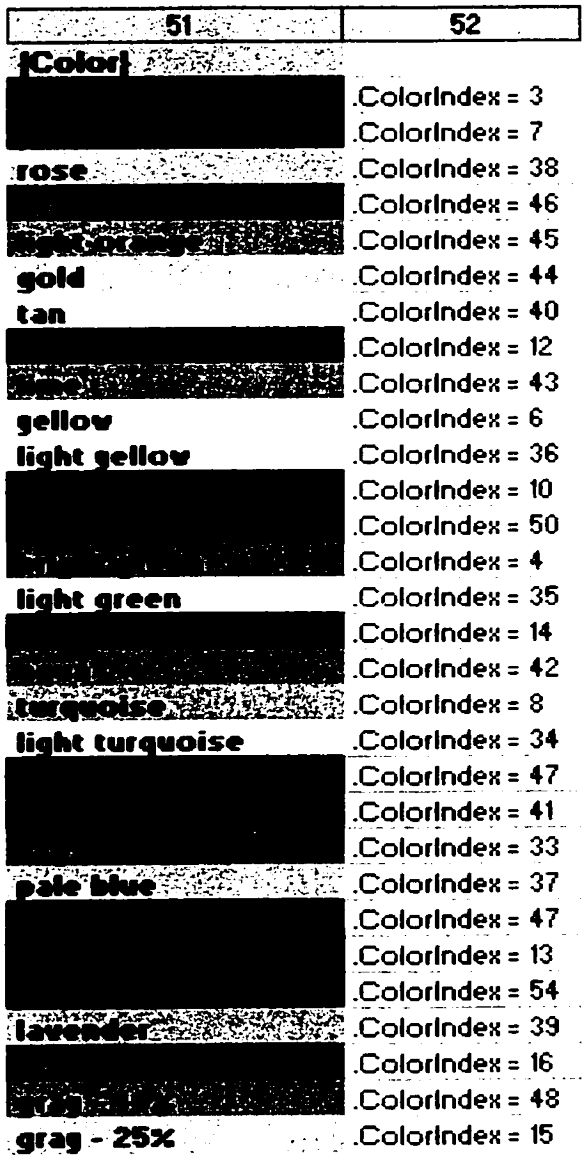
Figure A-3: Syntax for Cardinality

A.2.3 Apply Color

The Apply Color function is applying different color in each cell for the spreadsheet based on the value of the cells. The criteria is following:

1. First, the font of the whole row is bolded and the light gray color is used as background color only if this row has concept value.

2. Second, two columns before Concept is used for Cardinality function, and the number is colored Red.
3. The rest of the Context Maps will apply different color at the background of the cells based on the cell's value.
4. The detailed color code index is as following:



51	52
Color	
[illegible]	.ColorIndex = 3
[illegible]	.ColorIndex = 7
rose	.ColorIndex = 38
[illegible]	.ColorIndex = 46
[illegible]	.ColorIndex = 45
gold	.ColorIndex = 44
tan	.ColorIndex = 40
[illegible]	.ColorIndex = 12
[illegible]	.ColorIndex = 43
yellow	.ColorIndex = 6
light yellow	.ColorIndex = 36
[illegible]	.ColorIndex = 10
[illegible]	.ColorIndex = 50
[illegible]	.ColorIndex = 4
light green	.ColorIndex = 35
[illegible]	.ColorIndex = 14
[illegible]	.ColorIndex = 42
turquoise	.ColorIndex = 8
light turquoise	.ColorIndex = 34
[illegible]	.ColorIndex = 47
[illegible]	.ColorIndex = 41
[illegible]	.ColorIndex = 33
pale blue	.ColorIndex = 37
[illegible]	.ColorIndex = 47
[illegible]	.ColorIndex = 13
[illegible]	.ColorIndex = 54
lavender	.ColorIndex = 39
[illegible]	.ColorIndex = 16
[illegible]	.ColorIndex = 48
gray - 25%	.ColorIndex = 15

Figure A-4: The Color Index

5. The following lists the different color applied on different value of cells.

"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

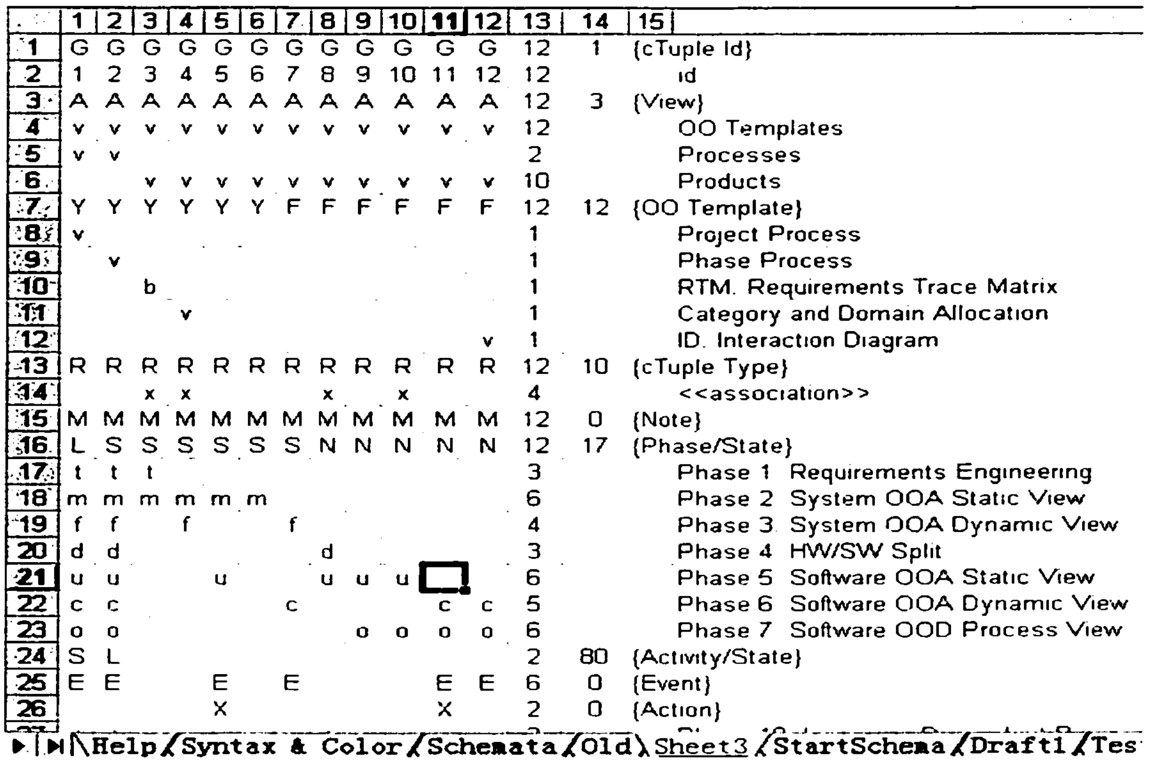
"G" --- Dark Green

"N" --- Rose

"S" --- Yellow

"t" --- Pale Blue

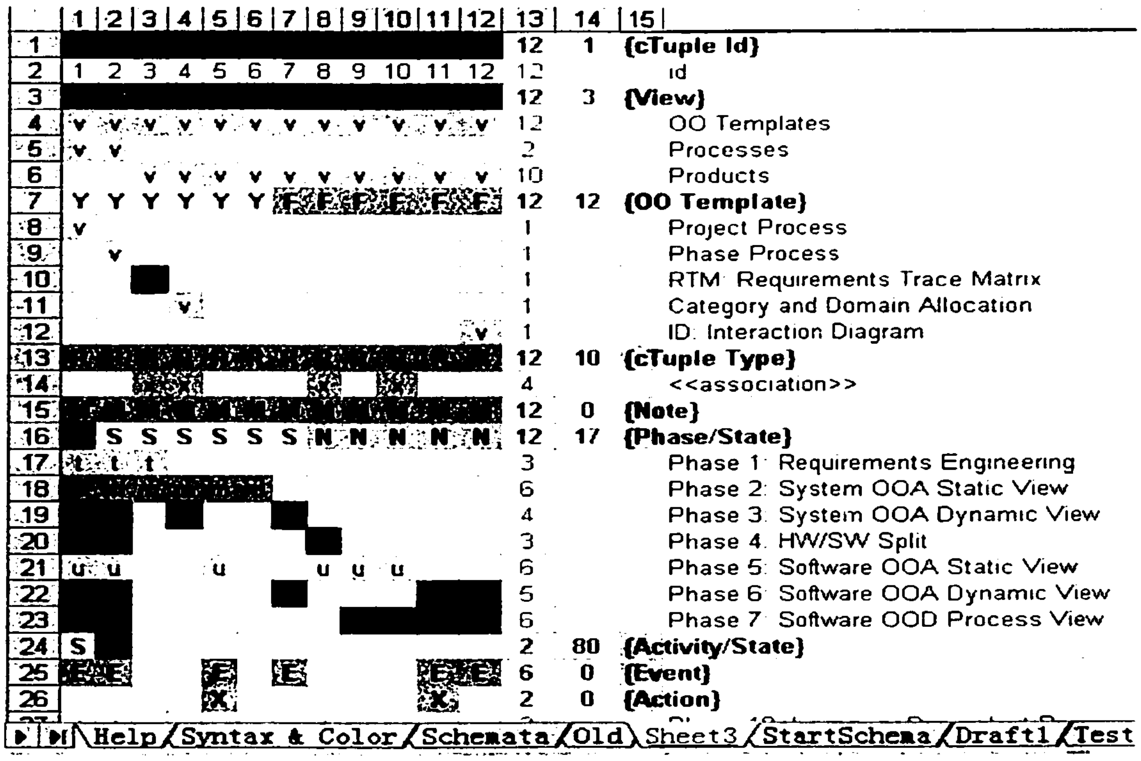
6. After the option button Apply Color is selected from Query Form (Figure A-8) or ApplyColor is clicked from sub-menu bar in CONTEXT+, "Apply Color" function will apply different color in the map. Figure A-5 is Input Test Map for Apply Color Function, and Figure A-6 is Output Result Map for Apply Color Function.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	G	G	G	G	G	G	G	G	G	G	G	G	12	1	{cTuple Id}
2	1	2	3	4	5	6	7	8	9	10	11	12	12		id
3	A	A	A	A	A	A	A	A	A	A	A	A	12	3	{View}
4	v	v	v	v	v	v	v	v	v	v	v	v	12		OO Templates
5	v	v											2		Processes
6			v	v	v	v	v	v	v	v	v	v	10		Products
7	Y	Y	Y	Y	Y	Y	F	F	F	F	F	F	12	12	{OO Template}
8	v												1		Project Process
9		v											1		Phase Process
10			b										1		RTM. Requirements Trace Matrix
11				v									1		Category and Domain Allocation
12												v	1		ID. Interaction Diagram
13	R	R	R	R	R	R	R	R	R	R	R	R	12	10	{cTuple Type}
14			x	x				x		x			4		<<association>>
15	M	M	M	M	M	M	M	M	M	M	M	M	12	0	{Note}
16	L	S	S	S	S	S	S	N	N	N	N	N	12	17	{Phase/State}
17	t	t	t										3		Phase 1 Requirements Engineering
18	m	m	m	m	m	m							6		Phase 2 System OOA Static View
19	f	f		f			f						4		Phase 3 System OOA Dynamic View
20	d	d						d					3		Phase 4 HW/SW Split
21	u	u			u			u	u	u	☐		6		Phase 5 Software OOA Static View
22	c	c					c				c	c	5		Phase 6 Software OOA Dynamic View
23	o	o							o	o	o	o	6		Phase 7 Software OOD Process View
24	S	L											2	80	{Activity/State}
25	E	E			E		E				E	E	6	0	{Event}
26					X						X		2	0	{Action}

\Help/Syntax & Color/Schemata/Old\Sheet3/StartSchema/Draft1/Tes

Figure A-5: Input Test Map for Apply Color Function



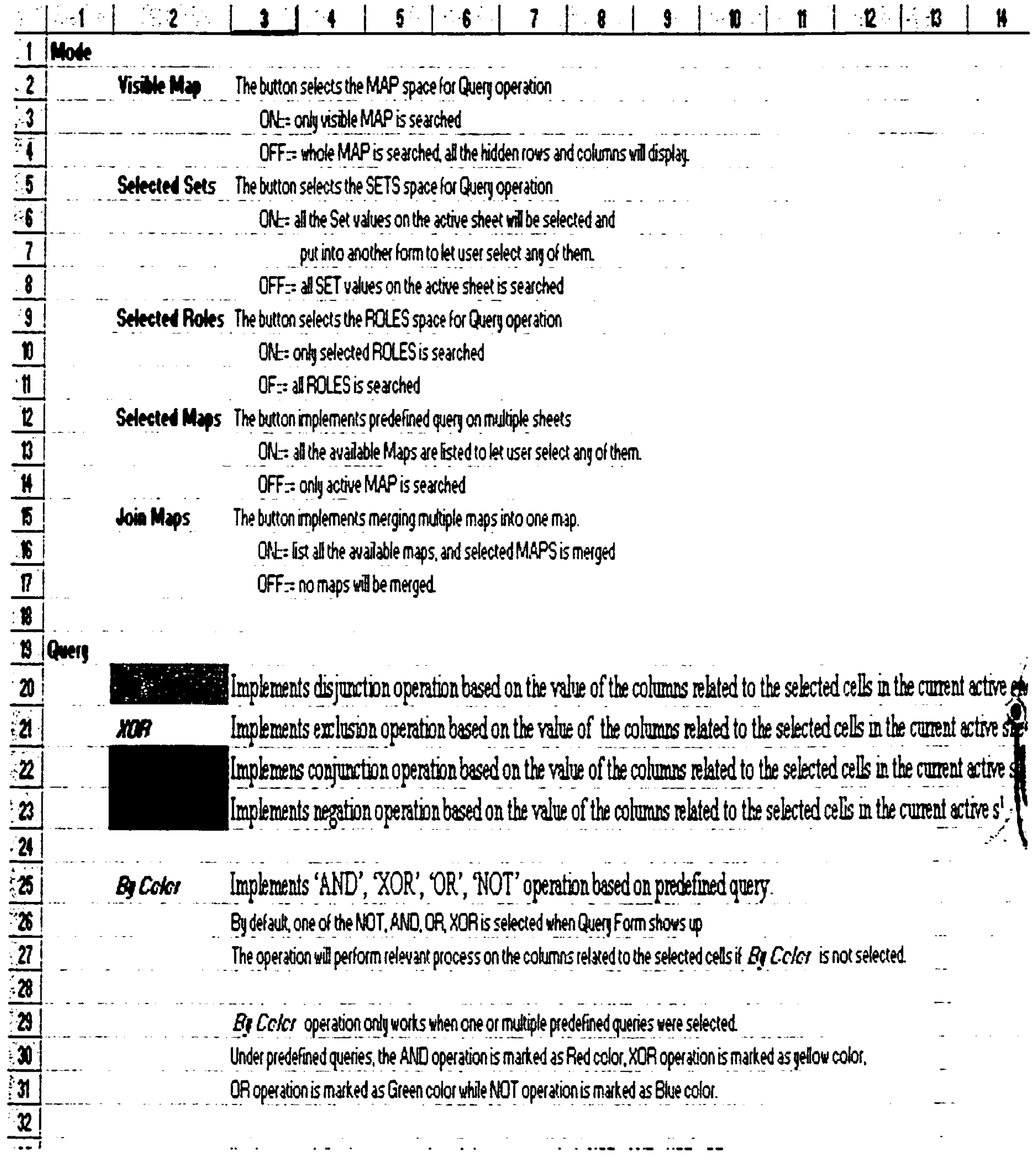
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	■	■	■	■	■	■	■	■	■	■	■	■	12	1	**{cTuple Id}**
2	1	2	3	4	5	6	7	8	9	10	11	12	12		id
3	■	■	■	■	■	■	■	■	■	■	■	■	12	3	**{View}**
4	v	v	v	v	v	v	v	v	v	v	v	v	12		OO Templates
5	v	v											2		Processes
6			v	v	v	v	v	v	v	v	v	v	10		Products
7	Y	Y	Y	Y	Y	Y	F	F	F	F	F	F	12	12	**{OO Template}**
8	v												1		Project Process
9		v											1		Phase Process
10			■										1		RTM Requirements Trace Matrix
11				v									1		Category and Domain Allocation
12												v	1		ID: Interaction Diagram
13	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	12	10	**{cTuple Type}**
14			▨	▨				▨		▨			4		<<association>>
15	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	12	0	**{Note}**
16	■	S	S	S	S	S	S	N	N	N	N	N	12	17	**{Phase/State}**
17	t	t	t										3		Phase 1: Requirements Engineering
18	▨	▨	▨	▨	▨	▨							6		Phase 2: System OOA Static View
19	■	■		■			■						4		Phase 3: System OOA Dynamic View
20	■	■						■					3		Phase 4: HW/SW Split
21	u	u			u			u	u	u			6		Phase 5: Software OOA Static View
22	■	■					■				■	■	5		Phase 6: Software OOA Dynamic View
23	■	■							■	■	■	■	6		Phase 7: Software OOD Process View
24	S	■											2	80	**{Activity/State}**
25	E	E			E		E				E	E	6	0	**{Event}**
26					X						X		2	0	**{Action}**

\Help/Syntax & Color/Schemata/Old\Sheet3/StartSchema/Draft1/Test

Figure A-6: Output Result Map for Apply Color Function

A.2.4 Help

When this option button ⭘ Help is clicked, the short version "Help" of Excel sheet will be displayed. This short version "Help" has brief description for each button on the Query Form.



Mode

Visible Map The button selects the MAP space for Query operation
ON:= only visible MAP is searched
OFF:= whole MAP is searched, all the hidden rows and columns will display.

Selected Sets The button selects the SETS space for Query operation
ON:= all the Set values on the active sheet will be selected and
put into another form to let user select any of them.
OFF:= all SET values on the active sheet is searched

Selected Roles The button selects the ROLES space for Query operation
ON:= only selected ROLES is searched
OF:= all ROLES is searched

Selected Maps The button implements predefined query on multiple sheets
ON:= all the available Maps are listed to let user select any of them.
OFF:= only active MAP is searched

Join Maps The button implements merging multiple maps into one map.
ON:= list all the available maps, and selected MAPS is merged
OFF:= no maps will be merged.

Query

Implements disjunction operation based on the value of the columns related to the selected cells in the current active

XOR Implements exclusion operation based on the value of the columns related to the selected cells in the current active s

Implemens conjunction operation based on the value of the columns related to the selected cells in the current active s

Implements negation operation based on the value of the columns related to the selected cells in the current active s

By Color Implements 'AND', 'XOR', 'OR', 'NOT' operation based on predefined query.
By default, one of the NOT, AND, OR, XOR is selected when Query Form shows up
The operation will perform relevant process on the columns related to the selected cells if *By Color* is not selected.

By Color operation only works when one or multiple predefined queries were selected.
Under predefined queries, the AND operation is marked as Red color, XOR operation is marked as yellow color,
OR operation is marked as Green color while NOT operation is marked as Blue color.

Figure A-7: Excel Version of Help Sheet

A.2.5 Query

1. After clicking Query from sub-menu in CONTEXT+, an interactive GUI Form will be popped up, Figure A-8.
2. In this form, there are four groups of buttons which can implement different functionalities.
3. The **Query** Section is the main and most important part. The **Mode** Section is used as input criteria for Query Section. The **Output** Section is the output format for the result of the Query Section. And **Run** Section has several independent functions which is as same as sub-menu in CONTEXT+.

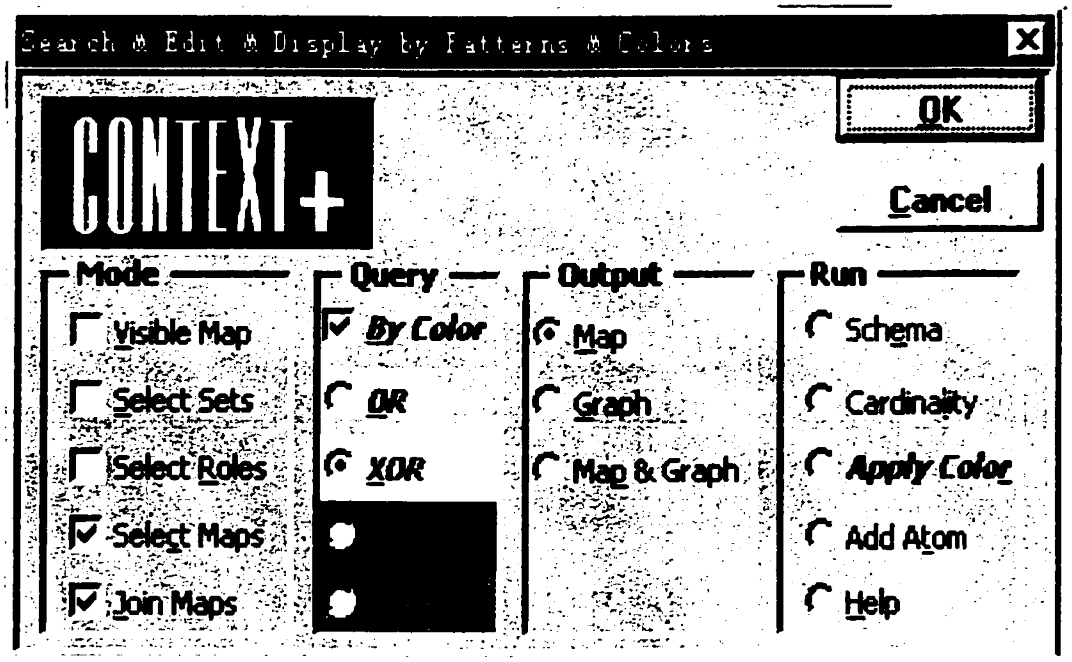


Figure A-8: Query Form for CONTEXT+ Menu

A.2.5.1 Query Section Functionality

1. AND, XOR, OR and NOT operations:

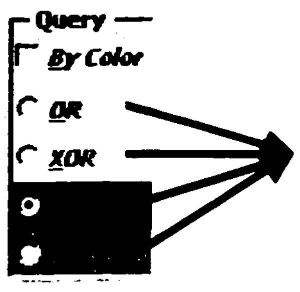
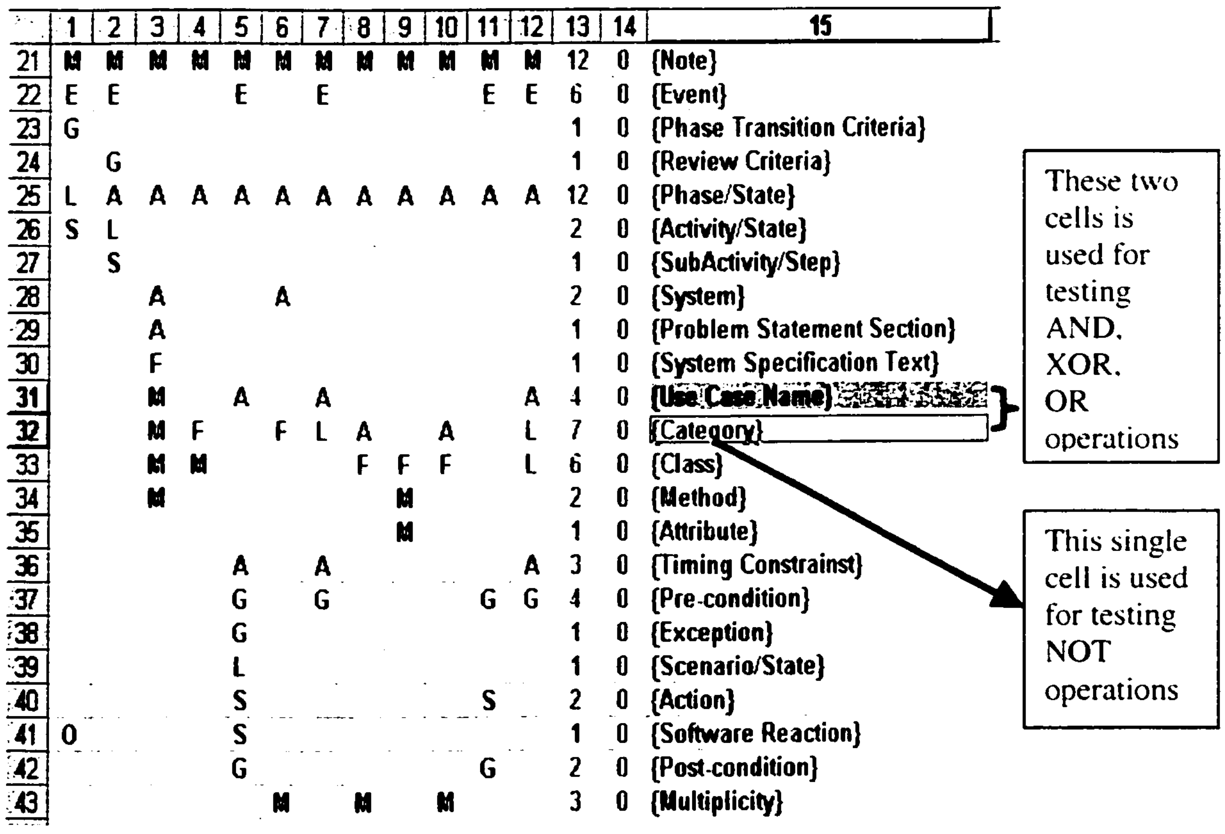


Figure A-9: Query Section

- These four buttons perform operations according to different colors. Figure 5-10 is a Test Input Sheet.
- AND and XOR operations allow 2 - n rows selected by holding CTRL key. OR operation allows selected 1 - n rows. NOT operation allows only single row. Otherwise it will display error message.
- The restriction of these functions is that the selected cells must have value.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	M	M	M	M	M	M	M	M	M	M	M	M	12	0	{Note}
22	E	E			E		E				E	E	6	0	{Event}
23	G												1	0	{Phase Transition Criteria}
24		G											1	0	{Review Criteria}
25	L	A	A	A	A	A	A	A	A	A	A	A	12	0	{Phase/State}
26	S	L											2	0	{Activity/State}
27		S											1	0	{SubActivity/Step}
28			A			A							2	0	{System}
29			A										1	0	{Problem Statement Section}
30			F										1	0	{System Specification Text}
31			M		A		A					A	4	0	{Use Case Name}
32			M	F		F	L	A		A		L	7	0	{Category}
33			M	M				F	F	F		L	6	0	{Class}
34			M						M				2	0	{Method}
35									M				1	0	{Attribute}
36					A		A					A	3	0	{Timing Constrainst}
37					G		G				G	G	4	0	{Pre-condition}
38					G								1	0	{Exception}
39					L								1	0	{Scenario/State}
40					S						S		2	0	{Action}
41	0				S								1	0	{Software Reaction}
42					G						G		2	0	{Post-condition}
43						M		M		M			3	0	{Multiplicity}

These two cells is used for testing AND, XOR, OR operations

This single cell is used for testing NOT operations

Figure A-10: Test Input Sheet for AND, XOR, OR, NOT Operations

AND------ Implementing conjunction operation based on the values of the columns related to the selected cells in the current active sheet.

1. Tag the 2-n selected cells by holding CTRL key.

2. Check the "AND" buttom in Figure A-9.

3. Click "OK" buttom.

The result map, Figure A-11after implementing **AND** operation on Figure A-10, will be displayed.

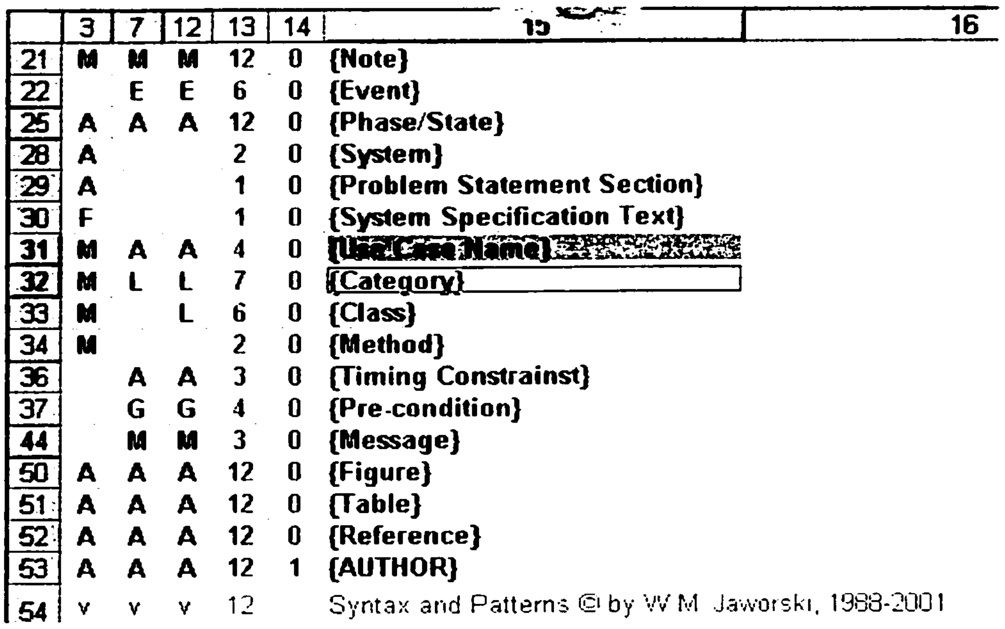
XOR ------ Implementing exclusion operation based on the value of the columns related to the selected cells in the current active sheet.

1. Tag the 2-n selected cells by holding CTRL key.

2. Check the "XOR" buttom in Figure A-9.

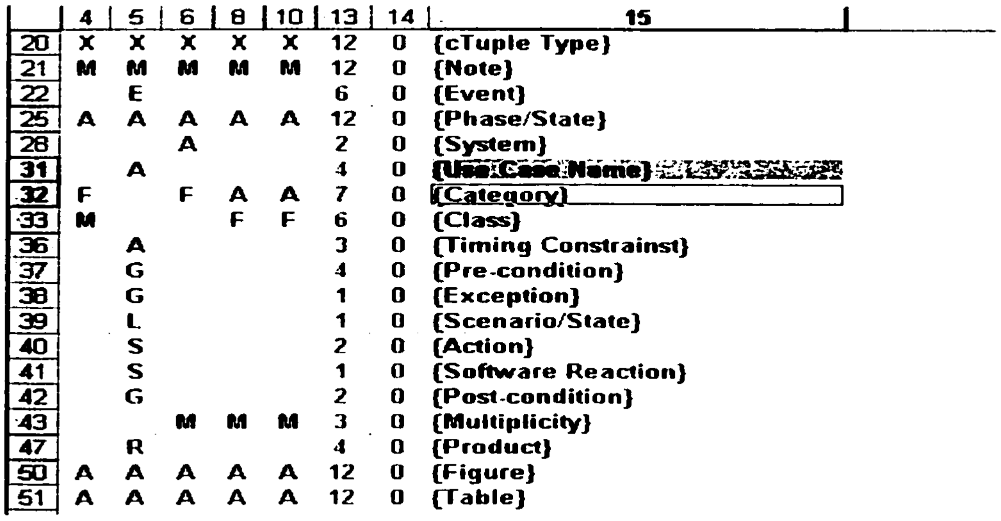
3. Click "OK" buttom.

The result map Figure A-12 after implementing **XOR** operation on Figure A-10 will be displayed.



	3	7	12	13	14	15	16
21	M	M	M	12	0	{Note}	
22		E	E	6	0	{Event}	
25	A	A	A	12	0	{Phase/State}	
28	A			2	0	{System}	
29	A			1	0	{Problem Statement Section}	
30	F			1	0	{System Specification Text}	
31	M	A	A	4	0	{UseCaseName}	
32	M	L	L	7	0	{Category}	
33	M		L	6	0	{Class}	
34	M			2	0	{Method}	
36		A	A	3	0	{Timing Constrainst}	
37		G	G	4	0	{Pre-condition}	
44		M	M	3	0	{Message}	
50	A	A	A	12	0	{Figure}	
51	A	A	A	12	0	{Table}	
52	A	A	A	12	0	{Reference}	
53	A	A	A	12	1	{AUTHOR}	
54	v	v	v	12		Syntax and Patterns © by W M Jaworski, 1988-2001	

Figure A-11: Result for AND Operation



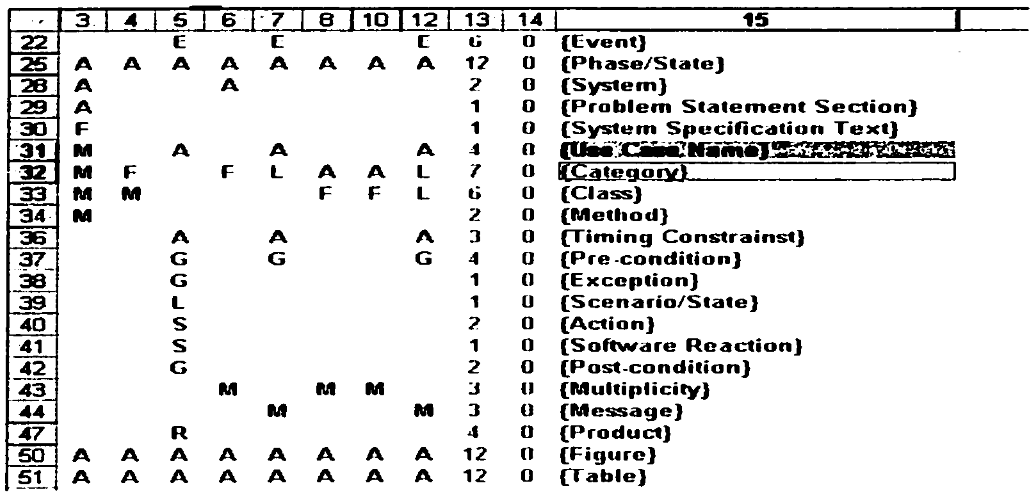
	4	5	6	8	10	13	14	15
20	X	X	X	X	X	12	0	{cTuple Type}
21	M	M	M	M	M	12	0	{Note}
22		E				6	0	{Event}
25	A	A	A	A	A	12	0	{Phase/State}
28			A			2	0	{System}
31		A				4	0	{Use Case Name}
32	F		F	A	A	7	0	{Category}
33	M			F	F	6	0	{Class}
36		A				3	0	{Timing Constrainst}
37		G				4	0	{Pre-condition}
38		G				1	0	{Exception}
39		L				1	0	{Scenario/State}
40		S				2	0	{Action}
41		S				1	0	{Software Reaction}
42		G				2	0	{Post-condition}
43			M	M	M	3	0	{Multiplicity}
47		R				4	0	{Product}
50	A	A	A	A	A	12	0	{Figure}
51	A	A	A	A	A	12	0	{Table}

Figure A-12: Result For XOR Operation

OR ------ Implementing disjunction operation based on the value of the columns related to the selected cells in the current active sheet.

1. Tag the 1-n selected cells by holding CTRL key.
2. Check the "OR" buttom in Figure A-9.
3. Click "OK" buttom.

The result map, Figure A-13 after implementing **OR** operation on Figure A-10, will be displayed.



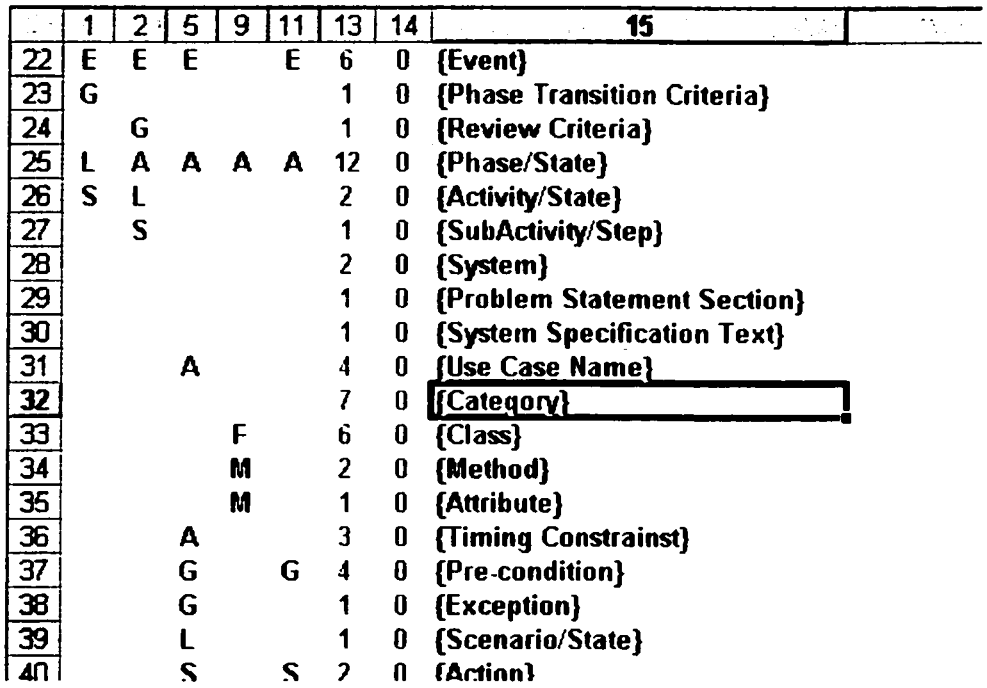
	3	4	5	6	7	8	10	12	13	14	15
22			E		E			E	6	0	{Event}
25	A	A	A	A	A	A	A	A	12	0	{Phase/State}
28	A			A					2	0	{System}
29	A								1	0	{Problem Statement Section}
30	F								1	0	{System Specification Text}
31	M		A		A			A	4	0	{Use Case Name}
32	M	F		F	L	A	A	L	7	0	{Category}
33	M	M				F	F	L	6	0	{Class}
34	M								2	0	{Method}
36			A		A			A	3	0	{Timing Constrainst}
37			G		G			G	4	0	{Pre-condition}
38			G						1	0	{Exception}
39			L						1	0	{Scenario/State}
40			S						2	0	{Action}
41			S						1	0	{Software Reaction}
42			G						2	0	{Post-condition}
43				M		M	M		3	0	{Multiplicity}
44					M			M	3	0	{Message}
47			R						4	0	{Product}
50	A	A	A	A	A	A	A	A	12	0	{Figure}
51	A	A	A	A	A	A	A	A	12	0	{Table}

Figure A-13: Result For OR Operation

NOT------ Implementing negation operation based on the value of the columns related to the selected cell in the current active sheet.

1. Tag the only single cell.
2. Check the "NOT" button in Figure A-9.
3. Click "OK" button.

The result map Figure A-14 after implementing **NOT** operation on Figure A-10 will be displayed.



	1	2	5	9	11	13	14	15
22	E	E	E		E	6	0	{Event}
23	G					1	0	{Phase Transition Criteria}
24		G				1	0	{Review Criteria}
25	L	A	A	A	A	12	0	{Phase/State}
26	S	L				2	0	{Activity/State}
27		S				1	0	{SubActivity/Step}
28						2	0	{System}
29						1	0	{Problem Statement Section}
30						1	0	{System Specification Text}
31			A			4	0	{Use Case Name}
32						7	0	{Category}
33				F		6	0	{Class}
34				M		2	0	{Method}
35				M		1	0	{Attribute}
36			A			3	0	{Timing Constrainst}
37			G		G	4	0	{Pre-condition}
38			G			1	0	{Exception}
39			L			1	0	{Scenario/State}
40			S		S	2	0	{Action}

Figure A-14: Result for NOT Operation

Note: AND, XOR and OR will hide the whole empty rows, however NOT won't. The NOT operation will hide the columns with values. For the example shows above, the whole row 32 is empty.

2. BY COLOR operation:

- This button will be implemented with one of 'AND', 'XOR', 'OR', 'NOT' operations based on selected predefined - query.
- The predefined - query should be marked in three places in the Context Maps.
    1) Concept Value cells under Concept Set column, as showing Predefined Query (1) in the Figure A-15.
    2) Cells with "q" value relevant to the Concept Value columns with predefined queries, as showing Predefined Query (2) in the Figure A-15.
    3) Tagged cells with color background, [illegible] stands for **AND** operation, Yellow stands for **XOR** operation, Bright Green stands for **OR** operation and [illegible] stands for **NOT** operation, as showing Predefined Query (3) in the Figure A-15. And all these color are optional.
- Select at least one predefined query as described above. If only one predefine query is selected, this function will ignore the selected 'AND' or 'XOR' or 'OR' or 'NOT' operations from the QUERY Form, and only implement the predefined query. If multiple predefined queries are selected, the function will implement each predefined query first, then it will merge the result based on the selected 'AND' or 'XOR' or 'OR' or 'NOT' operations. Detailed implementation as following:
    1) Predefined queries in the Context Map
    2) Tagged the cells for the queries to implement.
    3) Click one of the 'AND', 'XOR', 'OR', 'NOT' operations.
    4) Click 'By Color' button.

5) Click 'OK'.

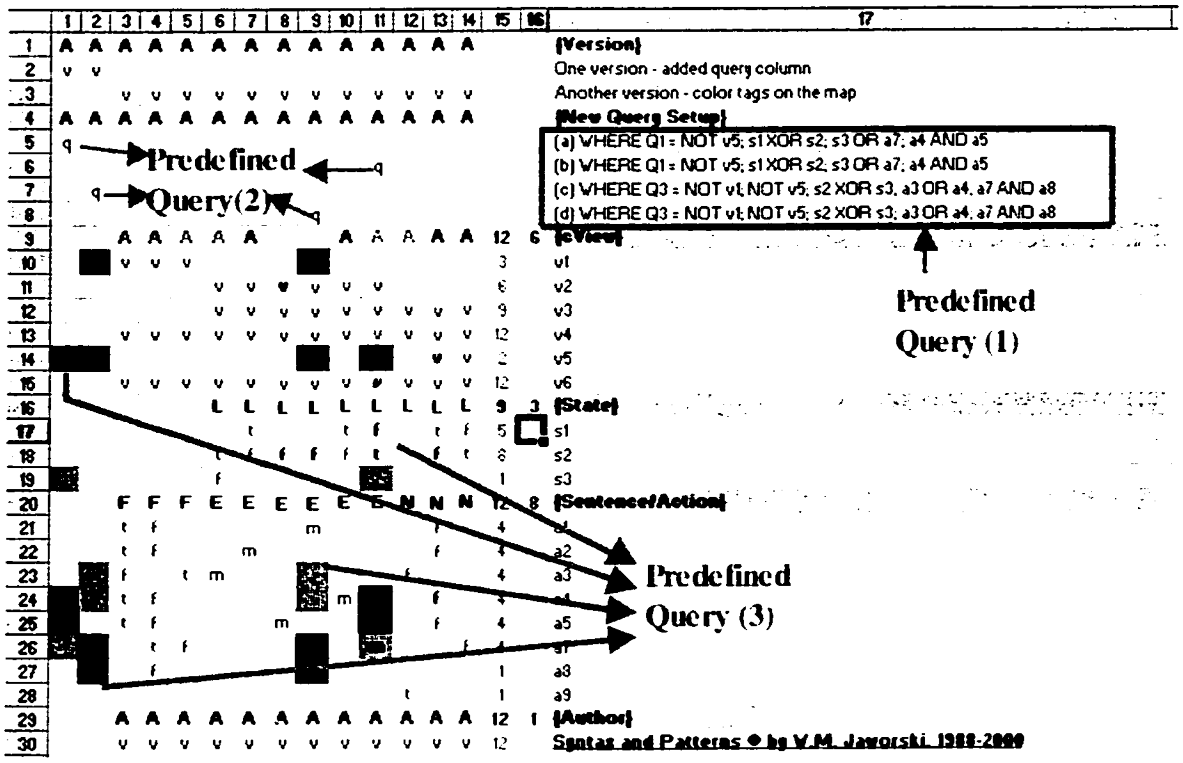
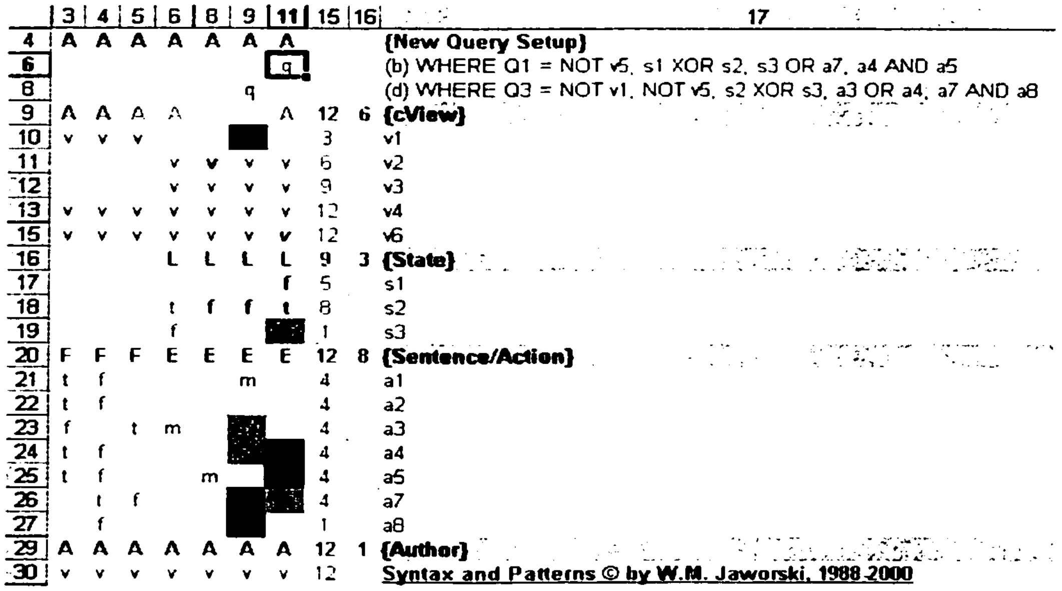

{Version}
One version - added query column
Another version - color tags on the map
{New Query Setup}
(a) WHERE Q1 = NOT v5; s1 XOR s2; s3 OR a7; a4 AND a5
(b) WHERE Q1 = NOT v5; s1 XOR s2; s3 OR a7; a4 AND a5
(c) WHERE Q3 = NOT v1; NOT v5; s2 XOR s3, a3 OR a4, a7 AND a8
(d) WHERE Q3 = NOT v1; NOT v5; s2 XOR s3; a3 OR a4, a7 AND a8
{eView}
{State}
{Sentence/Action}
{Author}
Syntax and Patterns © by V.M. Javorski 1988-2000

Figure A-15: Input Test Sheet for Query By Color

- After predefined, there is a restriction when performing the BY COLOR operation. In each predefined-query column, only one colored cell can be allowed to tag. Otherwise, a selection error message will be displayed.
- For example, in Figure A-15, the column 1, 2, 9, 11 are four predefined queries. If two predefined queries R6:C11 and R8:C9 are tagged, first the function will process predefined query R6:C11: (b) WHERE Q1 =NOT v5; s1 XOR s2; s3 OR a7; a4 AND a5, the result will be saved in the memory and showing as following Figure A-16.

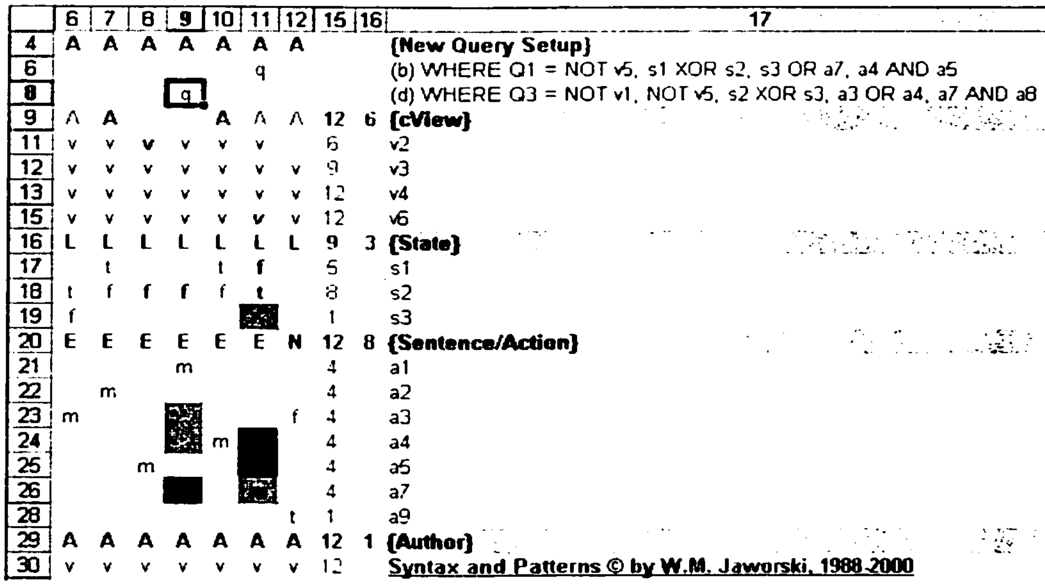


	3	4	5	6	8	9	11	15	16	17
4	A	A	A	A	A	A	A			**{New Query Setup}**
6							q			(b) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
8						q				(d) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3, a3 OR a4, a7 AND a8
9	A	A	A	A			A	12	6	**{cView}**
10	v	v	v			■		3		v1
11				v	v	v	v	6		v2
12				v	v	v	v	9		v3
13	v	v	v	v	v	v	v	12		v4
15	v	v	v	v	v	v	v	12		v6
16				L	L	L	L	9	3	**{State}**
17							f	5		s1
18				t	f	f	t	8		s2
19				f			▨	1		s3
20	F	F	F	E	E	E	E	12	8	**{Sentence/Action}**
21	t	f				m		4		a1
22	t	f						4		a2
23	f		t	m		▨		4		a3
24	t	f				▨	■	4		a4
25	t	f			m		■	4		a5
26		t	f			■	▨	4		a7
27		f				■		1		a8
29	A	A	A	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	v	v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-16: Temporary Result Sheet For Implementing Predefined

Query Q1 of "By Color" Function

Second the function will process the second predefined query R8:C9: (d) WHERE Q3 = NOT v1; NOT v5; s2 XOR s3; a3 OR a4; a7 AND a8. The result will be saved in the memory also and shown as in Figure A-17.

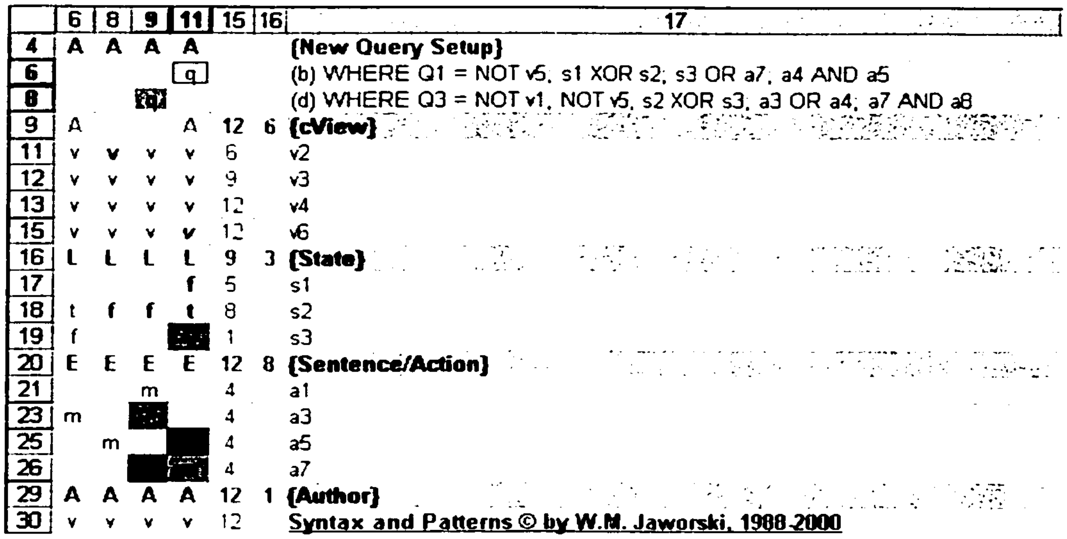


	6	7	8	9	10	11	12	15	16	17
4	A	A	A	A	A	A	A			**{New Query Setup}**
6						q				(b) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
8				q						(d) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3, a3 OR a4, a7 AND a8
9	A	A			A	A	A	12	6	**{cView}**
11	v	v	v	v	v	v		6		v2
12	v	v	v	v	v	v	v	9		v3
13	v	v	v	v	v	v	v	12		v4
15	v	v	v	v	v	v	v	12		v6
16	L	L	L	L	L	L	L	9	3	**{State}**
17		t			t	f		5		s1
18	t	f	f	f	f	t		8		s2
19	f					▨		1		s3
20	E	E	E	E	E	E	N	12	8	**{Sentence/Action}**
21				m				4		a1
22		m						4		a2
23	m			▨			f	4		a3
24				▨	m	■		4		a4
25			m			■		4		a5
26				■		▨		4		a7
28							t	1		a9
29	A	A	A	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	v	v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-17: Temporary Result Sheet For Implementing Predefined

Query Q3 of "By Color" Function

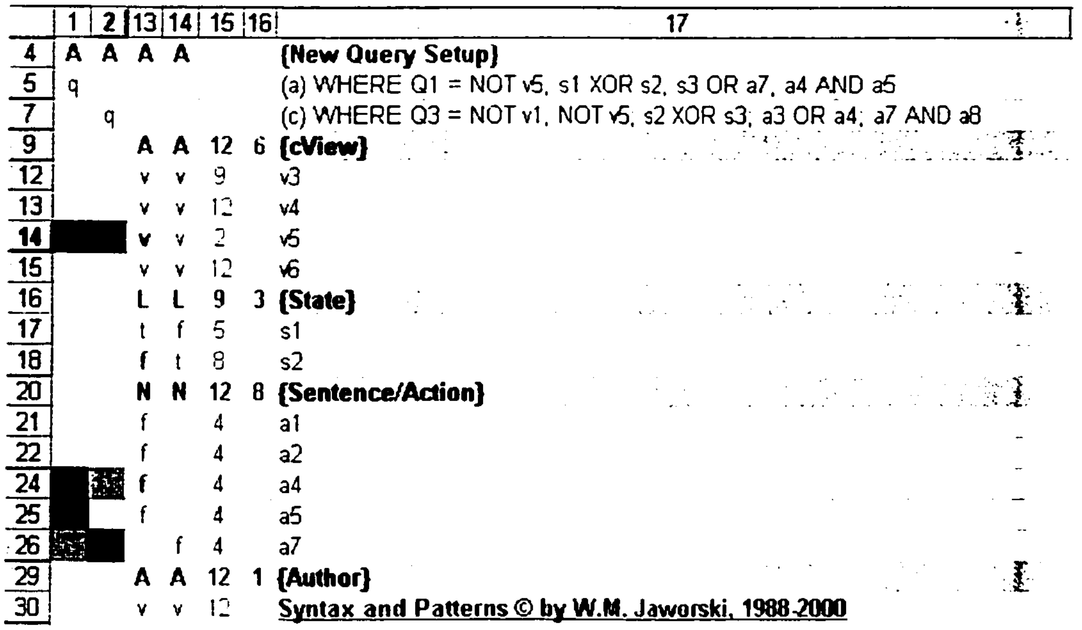
Finally, if AND operation is selected, it will process Q1 AND Q3, the result will be as following Figure A-18.



	6	8	9	11	15	16	17
4	A	A	A	A			**{New Query Setup}**
6				q			(b) WHERE Q1 = NOT v5, s1 XOR s2; s3 OR a7, a4 AND a5
8			q				(d) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3, a3 OR a4; a7 AND a8
9	A			A	12	6	**{cView}**
11	v	v	v	v	6		v2
12	v	v	v	v	9		v3
13	v	v	v	v	12		v4
15	v	v	v	v	12		v6
16	L	L	L	L	9	3	**{State}**
17				f	5		s1
18	t	f	f	t	8		s2
19	f			■	1		s3
20	E	E	E	E	12	8	**{Sentence/Action}**
21			m		4		a1
23	m		■		4		a3
25		m		■	4		a5
26			■	■	4		a7
29	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-18: Output Result Sheet for Q1 AND Q3.

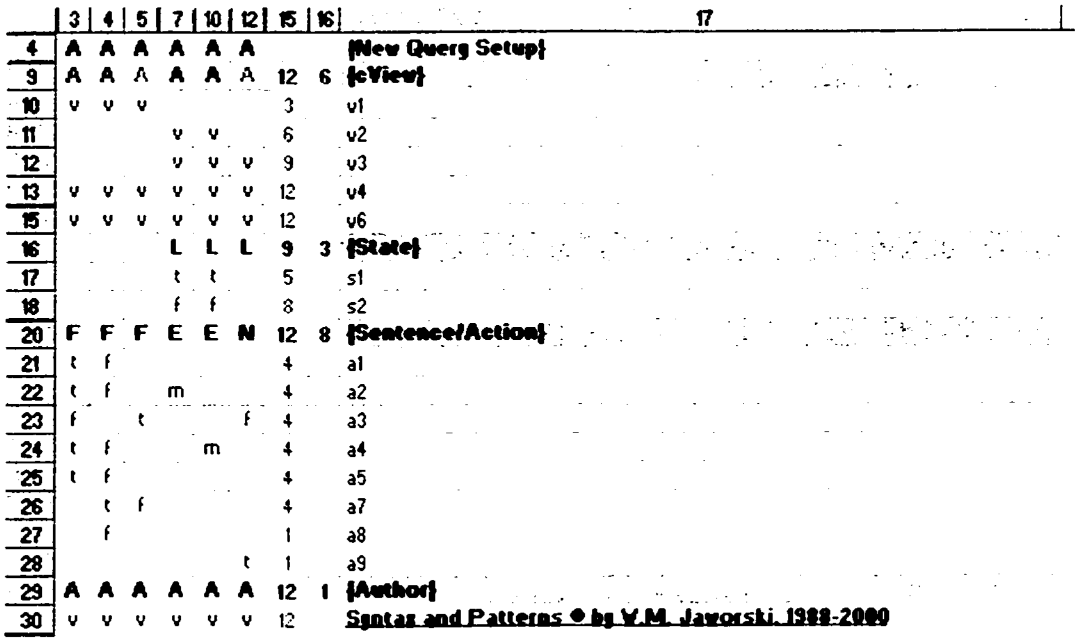
If the NOT operation is selected, it will process the opposite way to AND operation. It will display all the columns not showing in Q1 and Q3. The result is shown in Figure A-19.



	1	2	13	14	15	16	17
4	A	A	A	A			**{New Query Setup}**
5	q						(a) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
7		q					(c) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3; a3 OR a4; a7 AND a8
9			A	A	12	6	**{cView}**
12			v	v	9		v3
13			v	v	12		v4
14	■	■	v	v	2		v5
15			v	v	12		v6
16			L	L	9	3	**{State}**
17			t	f	5		s1
18			f	t	8		s2
20			N	N	12	8	**{Sentence/Action}**
21			f		4		a1
22			f		4		a2
24	■	■	f		4		a4
25	■		f		4		a5
26	■	■		f	4		a7
29			A	A	12	1	**{Author}**
30			v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-19: Output Result Sheet for Q1 NOT Q3

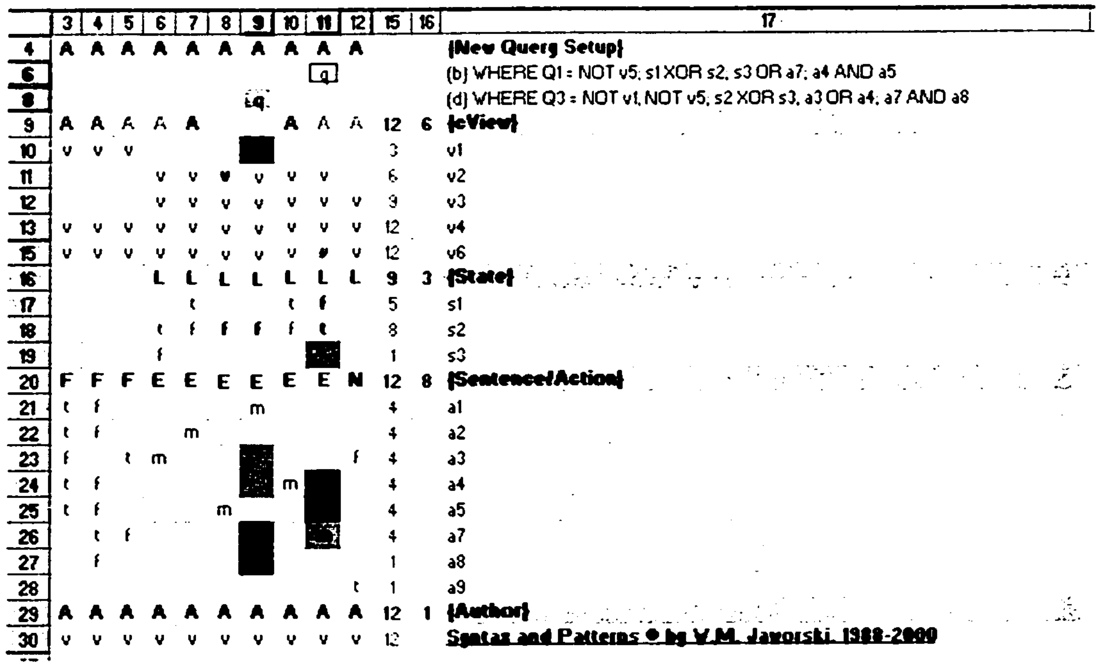
If XOR operation is selected, the result will be shown in Figure A-20.



	3	4	5	7	10	12	15	16	17
4	A	A	A	A	A	A			**{New Query Setup}**
9	A	A	A	A	A	A	12	6	**{cView}**
10	v	v	v				3		v1
11				v	v		6		v2
12				v	v	v	9		v3
13	v	v	v	v	v	v	12		v4
15	v	v	v	v	v	v	12		v6
16				L	L	L	9	3	**{State}**
17				t	t		5		s1
18				f	f		8		s2
20	F	F	F	E	E	N	12	8	**{Sentence/Action}**
21	t	f					4		a1
22	t	f		m			4		a2
23	f		t			f	4		a3
24	t	f			m		4		a4
25	t	f					4		a5
26		t	f				4		a7
27		f					1		a8
28						t	1		a9
29	A	A	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	v	v	12		**Syntax and Patterns © by V.M. Javorski. 1988-2000**

Figure A-20: Output Result Sheet for Q1 XOR Q3

If OR operation is selected, the result will be shown in Figure A-21.



	3	4	5	6	7	8	9	10	11	12	15	16	17
4	A	A	A	A	A	A	A	A	A	A			**{New Query Setup}**
6									q				(b) WHERE Q1 = NOT v5; s1 XOR s2, s3 OR a7; a4 AND a5
8							Lq						(d) WHERE Q3 = NOT v1, NOT v5; s2 XOR s3, a3 OR a4; a7 AND a8
9	A	A	A	A	A			A	A	A	12	6	**{cView}**
10	v	v	v				■				3		v1
11				v	v	v	v	v	v		6		v2
12				v	v	v	v	v	v	v	9		v3
13	v	v	v	v	v	v	v	v	v	v	12		v4
15	v	v	v	v	v	v	v	v	v	v	12		v6
16				L	L	L	L	L	L	L	9	3	**{State}**
17					t			t	f		5		s1
18				t	f	f	f	f	t		8		s2
19				f					■		1		s3
20	F	F	F	E	E	E	E	E	E	N	12	8	**{Sentence/Action}**
21	t	f					m				4		a1
22	t	f			m						4		a2
23	f		t	m			■			f	4		a3
24	t	f					■	m	■		4		a4
25	t	f				m			■		4		a5
26		t	f				■		■		4		a7
27		f					■				1		a8
28										t	1		a9
29	A	A	A	A	A	A	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	v	v	v	v	v	v	12		**Syntax and Patterns © by V.M. Javorski. 1988-2000**

Figure A-21: Output Result Sheet for Q1 OR Q3

A.2.5.2 Mode Section Functionality

Visible Maps------This button has to work with Query Section and display the map without hidden part.

1. Check the Query Section.
2. If Visible box is selected, the result map will only display visible part of current active sheet.
3. Otherwise, the result map will display all the hidden rows and columns.

Select Sets ------- This function will automatically list the all Set values the active sheet has, users can check all or some of these Set values they would like. Figure A-22 is a sample of Select Sets Form.

1. This button should be combined with Query Section.
   (a) Check the Query Section, click the Select Sets button, click the "OK" button from Query Form
   (b) The user will see the prompt form as Figure A-22, and then the user can select all or some sets.
   (c) After clicking "OK" button, the result map will only remain the selected sets part of the map after implementation.
2. There is an exception when it works with 'By Color' function. While the 'By Color' button is selected and the 'Select Sets' is not selected, it will automatically perform the Select Sets function without displaying the Select Sets Form. Under this condition, it checks the sets with light gray

background colors in concept column and only remains these sets in the result map.

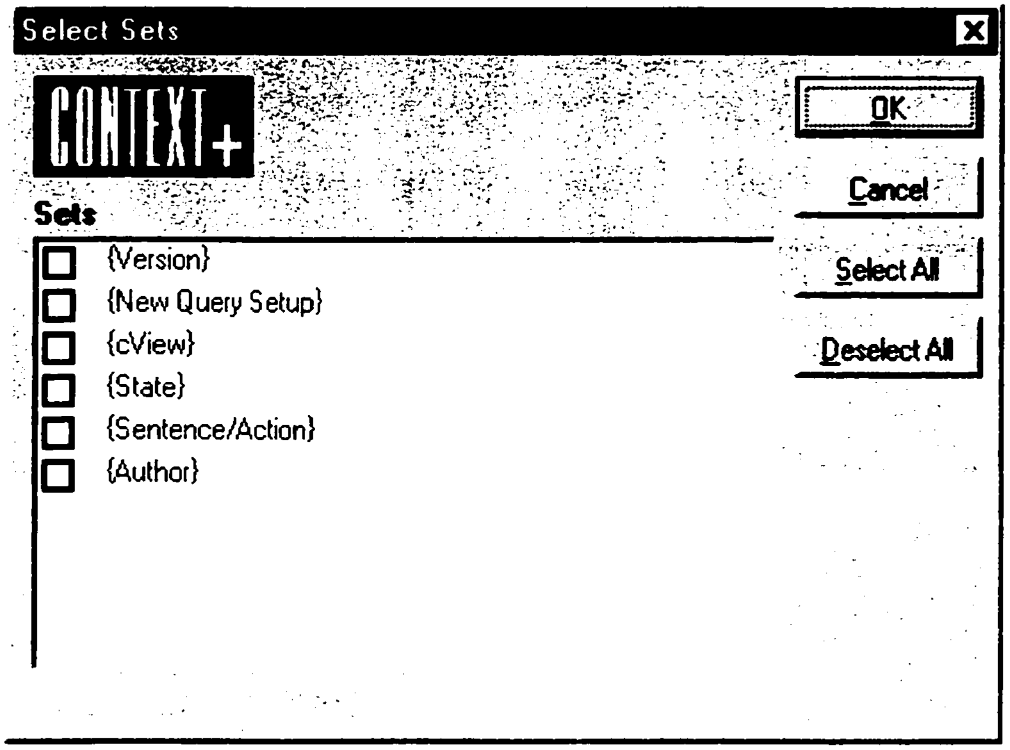
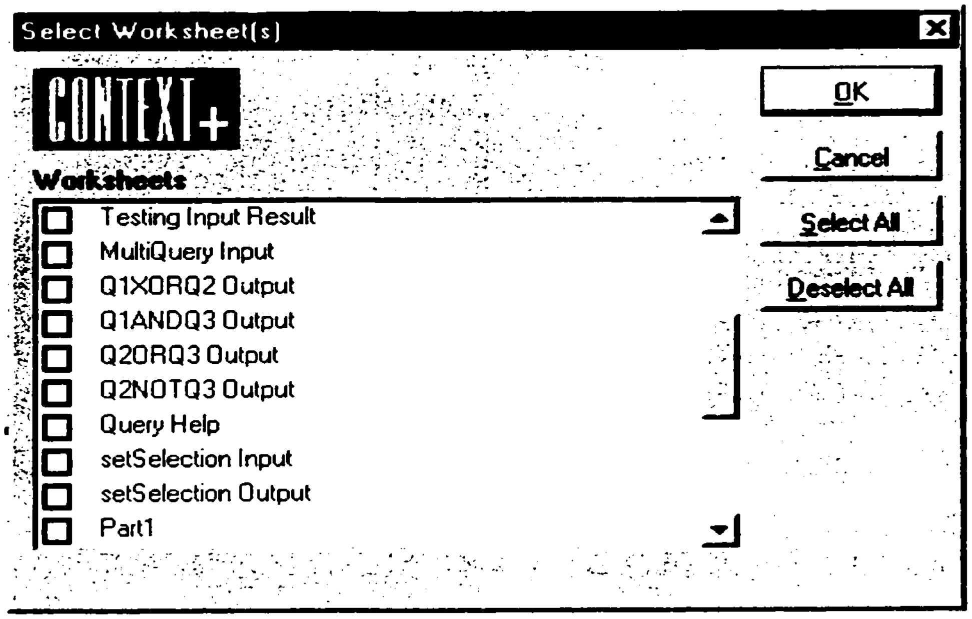


Figure A-22: Select Sets Form

Select Maps -------- This function will list all the available worksheets for the current workbook and user can check all or some of these worksheets to perform "By Color" operation simultaneously. Figure A-23 is a sample of WorkSheets List Form.

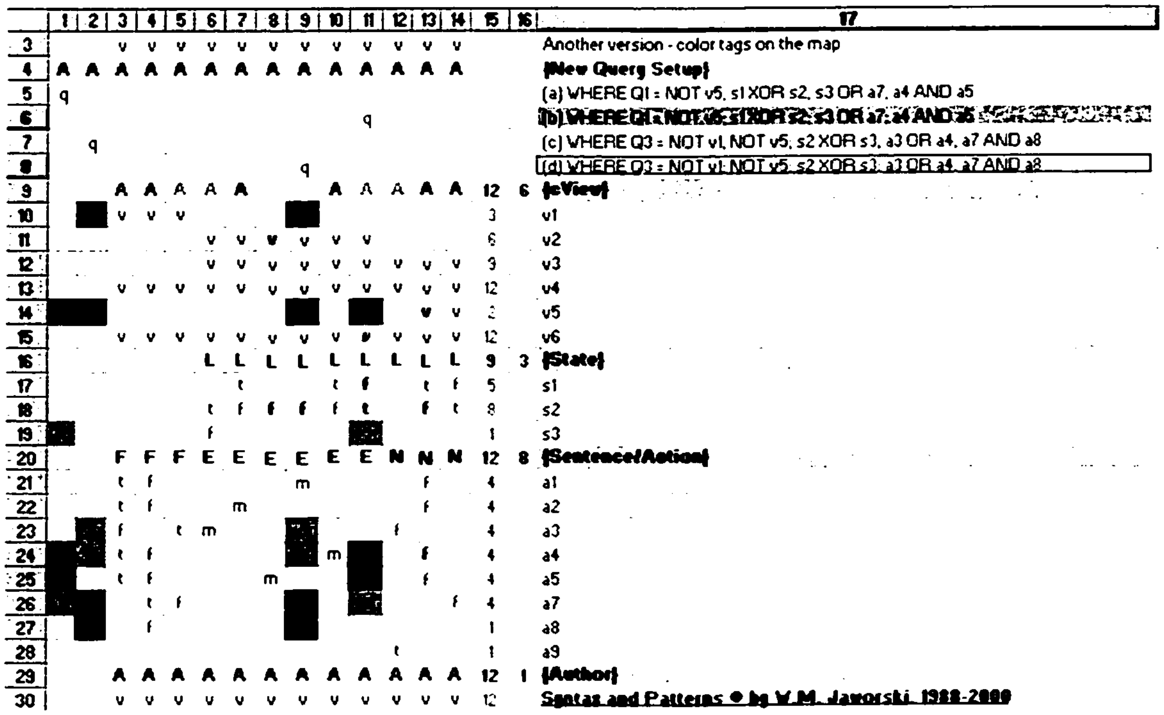
1. This button should work with "By Color" function.
2. The criterion for this button is that predefined query should be selected before processing this function.
3. When "Select Maps" button is selected, the following form will pop up.



Testing Input Result
MultiQuery Input
Q1XORQ2 Output
Q1ANDQ3 Output
Q2ORQ3 Output
Q2NOTQ3 Output
Query Help
setSelection Input
setSelection Output
Part1

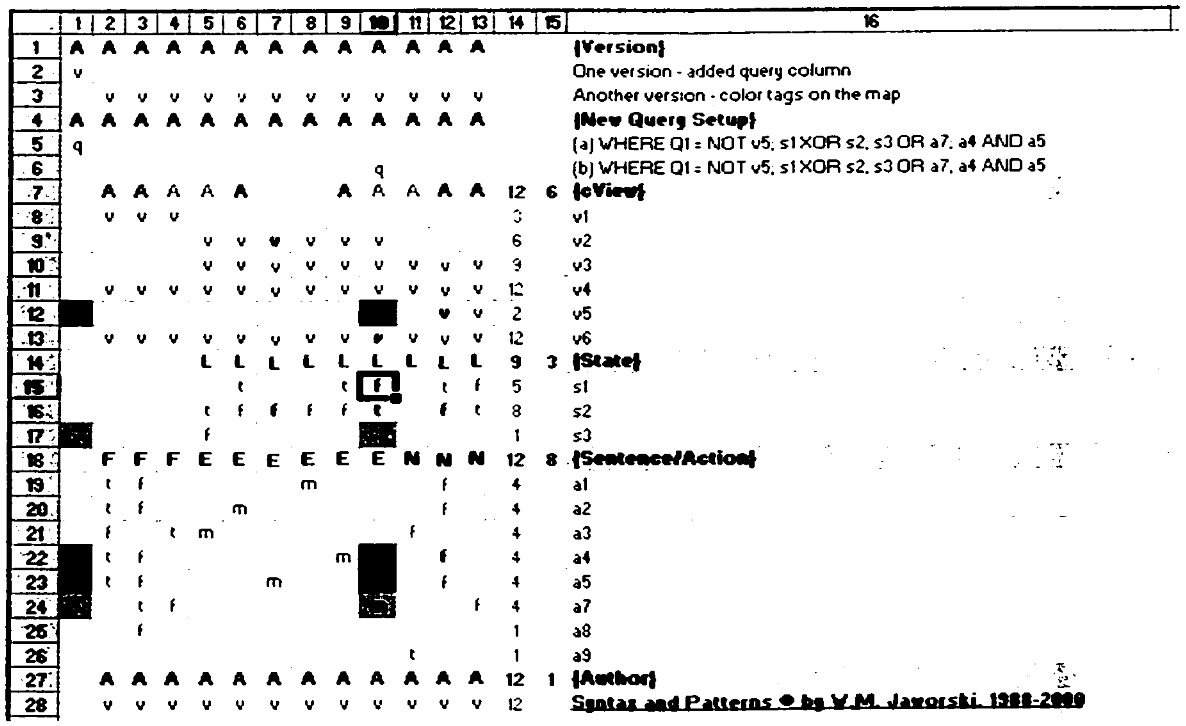
Figure A-23: WorkSheets List Form for Current Active WorkBook

4. After selecting worksheets, Excel will automatically open several new sheets corresponding to the number of sheets user selected under the same workbook and rename the new sheets as original sheets name plus "Result" suffix respectively. If there are already the same name sheets in the current active workbook, it will prompt a message to ask whether user would like to delete the old sheets. It should always click **YES** because Excel won't allow two same name sheets existing in one workbook.
5. The following figures will demonstrate 'Select Maps' function. From Query Form, click "Select Maps", "By Color" and "AND" buttons, then click "OK" button. It pops up a Worksheets List Form. Figure A-24 and Figure A-25 are two input worksheets selected from Worksheets List Form. After implementation, two new worksheets Figure A-26 and Figure A-27 are automatically added to save the corresponding result.



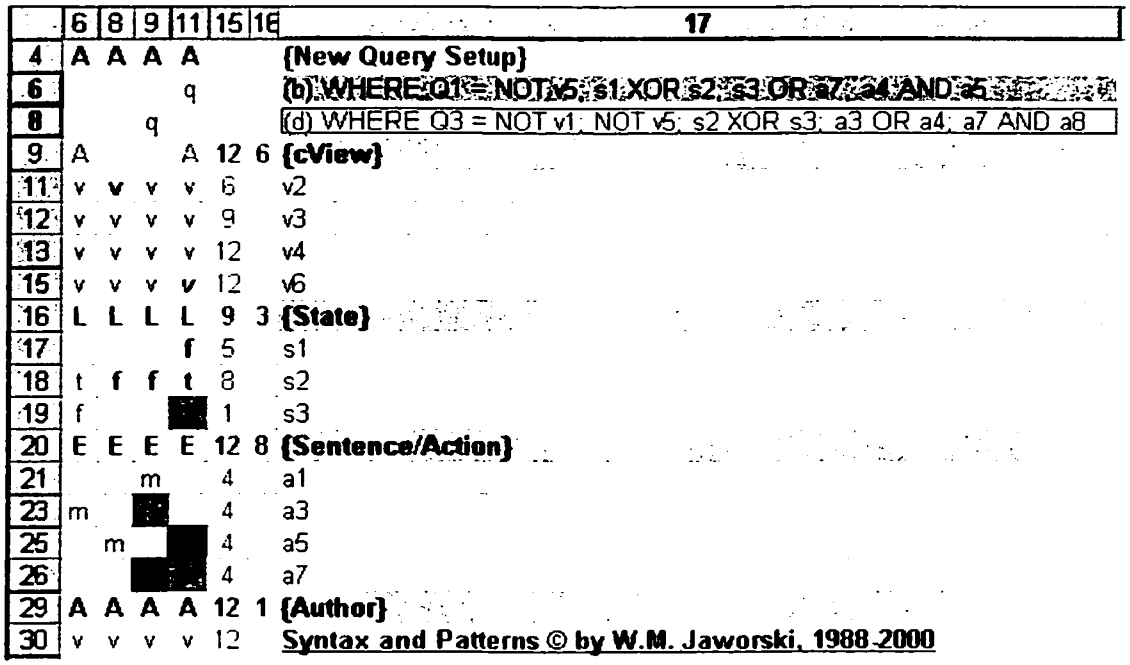
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
3			v	v	v	v	v	v	v	v	v	v	v	v			Another version - color tags on the map
4	A	A	A	A	A	A	A	A	A	A	A	A	A	A			**{New Query Setup}**
5	q																(a) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
6											q						**(b) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5**
7		q															(c) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3, a3 OR a4, a7 AND a8
8									q								(d) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3, a3 OR a4, a7 AND a8
9			A	A	A	A	A			A	A	A	A	A	12	6	**{cView}**
10		■	v	v	v				■						3		v1
11						v	v	v	v	v	v				6		v2
12						v	v	v	v	v	v	v	v	v	9		v3
13			v	v	v	v	v	v	v	v	v	v	v	v	12		v4
14	■	■							■		■		v	v	2		v5
15			v	v	v	v	v	v	v	v	v	v	v	v	12		v6
16						L	L	L	L	L	L	L	L	L	9	3	**{State}**
17							t			t	f		t	f	5		s1
18						t	f	f	f	f	t		f	t	8		s2
19	■					f					■				1		s3
20			F	F	F	E	E	E	E	E	E	N	N	N	12	8	**{Sentence/Action}**
21			t	f					m				f		4		a1
22			t	f			m						f		4		a2
23		■	f		t	m			■			f			4		a3
24	■	■	t	f					■	m	■		f		4		a4
25	■		t	f				m			■		f		4		a5
26	■	■		t	f				■		■			f	4		a7
27		■		f					■						1		a8
28												t			1		a9
29			A	A	A	A	A	A	A	A	A	A	A	A	12	1	**{Author}**
30			v	v	v	v	v	v	v	v	v	v	v	v	12		**Syntax and Patterns © by V.M. Javorski 1988-2000**

Figure A-24: Input Sheet1 For 'Select Maps' Function



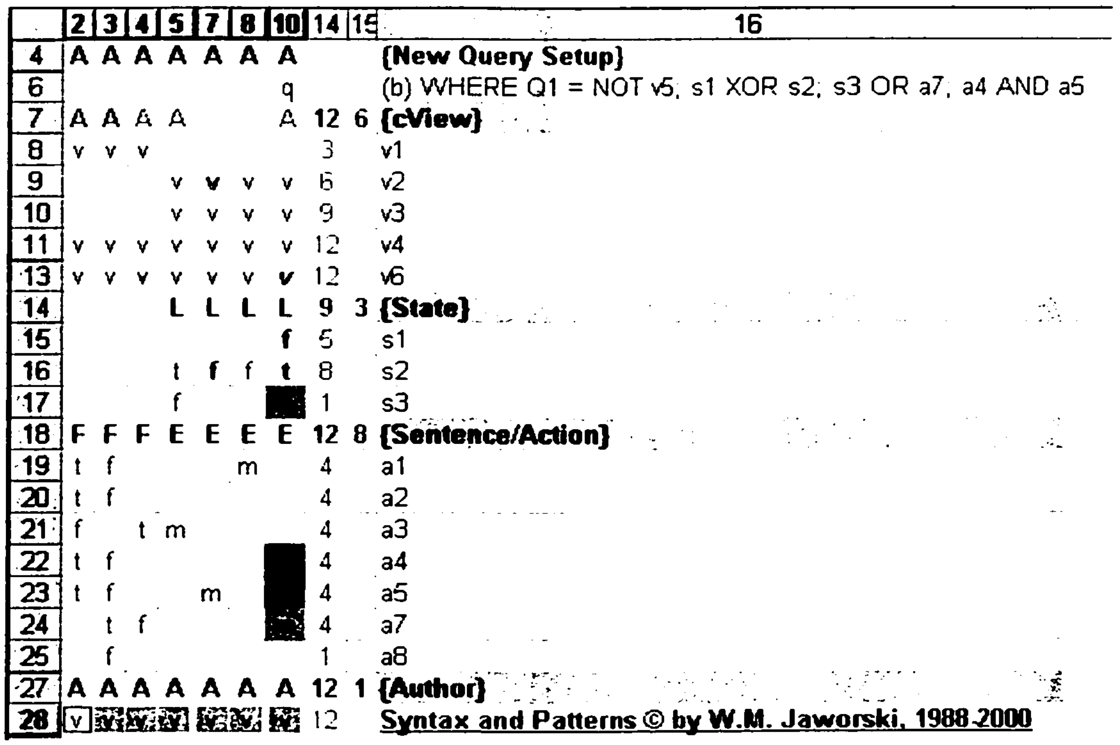
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	A	A	A	A	A	A	A	A	A	A	A	A	A			**{Version}**
2	v															One version - added query column
3		v	v	v	v	v	v	v	v	v	v	v	v			Another version - color tags on the map
4	A	A	A	A	A	A	A	A	A	A	A	A	A			**{New Query Setup}**
5	q															(a) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
6										q						(b) WHERE Q1 = NOT v5, s1 XOR s2, s3 OR a7, a4 AND a5
7		A	A	A	A	A			A	A	A	A	A	12	6	**{cView}**
8		v	v	v										3		v1
9					v	v	v	v	v	v				6		v2
10					v	v	v	v	v	v	v	v	v	9		v3
11		v	v	v	v	v	v	v	v	v	v	v	v	12		v4
12	■									■		v	v	2		v5
13		v	v	v	v	v	v	v	v	v	v	v	v	12		v6
14					L	L	L	L	L	L	L	L	L	9	3	**{State}**
15						t			t	f		t	f	5		s1
16					t	f	f	f	f	t		f	t	8		s2
17	■				f					■				1		s3
18		F	F	F	E	E	E	E	E	E	N	N	N	12	8	**{Sentence/Action}**
19		t	f					m				f		4		a1
20		t	f			m						f		4		a2
21		f		t	m						f			4		a3
22	■	t	f						m	■		f		4		a4
23	■	t	f				m			■		f		4		a5
24	■		t	f						■			f	4		a7
25			f											1		a8
26											t			1		a9
27		A	A	A	A	A	A	A	A	A	A	A	A	12	1	**{Author}**
28		v	v	v	v	v	v	v	v	v	v	v	v	12		**Syntax and Patterns © by V.M. Javorski 1988-2000**

Figure A-25: Input Sheet2 For 'Select Maps' Function



	6	8	9	11	15	16	17
4	A	A	A	A			**{New Query Setup}**
6				q			**(b) WHERE Q1 = NOT v5; s1 XOR s2; s3 OR a7; a4 AND a5**
8			q				(d) WHERE Q3 = NOT v1; NOT v5; s2 XOR s3; a3 OR a4; a7 AND a8
9	A			A	12	6	**{cView}**
11	v	v	v	v	6		v2
12	v	v	v	v	9		v3
13	v	v	v	v	12		v4
15	v	v	v	v	12		v6
16	L	L	L	L	9	3	**{State}**
17				f	5		s1
18	t	f	f	t	8		s2
19	f			■	1		s3
20	E	E	E	E	12	8	**{Sentence/Action}**
21			m		4		a1
23	m		■		4		a3
25		m		■	4		a5
26			■	■	4		a7
29	A	A	A	A	12	1	**{Author}**
30	v	v	v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-26: Result For Input Sheet1 after Implementing 'Select Maps' Function



	2	3	4	5	7	8	10	14	15	16
4	A	A	A	A	A	A	A			**{New Query Setup}**
6							q			(b) WHERE Q1 = NOT v5, s1 XOR s2; s3 OR a7, a4 AND a5
7	A	A	A	A			A	12	6	**{cView}**
8	v	v	v					3		v1
9				v	v	v	v	6		v2
10				v	v	v	v	9		v3
11	v	v	v	v	v	v	v	12		v4
13	v	v	v	v	v	v	v	12		v6
14				L	L	L	L	9	3	**{State}**
15							f	5		s1
16				t	f	f	t	8		s2
17				f			■	1		s3
18	F	F	F	E	E	E	E	12	8	**{Sentence/Action}**
19	t	f				m		4		a1
20	t	f						4		a2
21	f		t	m				4		a3
22	t	f					■	4		a4
23	t	f			m		■	4		a5
24		t	f				■	4		a7
25		f						1		a8
27	A	A	A	A	A	A	A	12	1	**{Author}**
28	v	v	v	v	v	v	v	12		**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

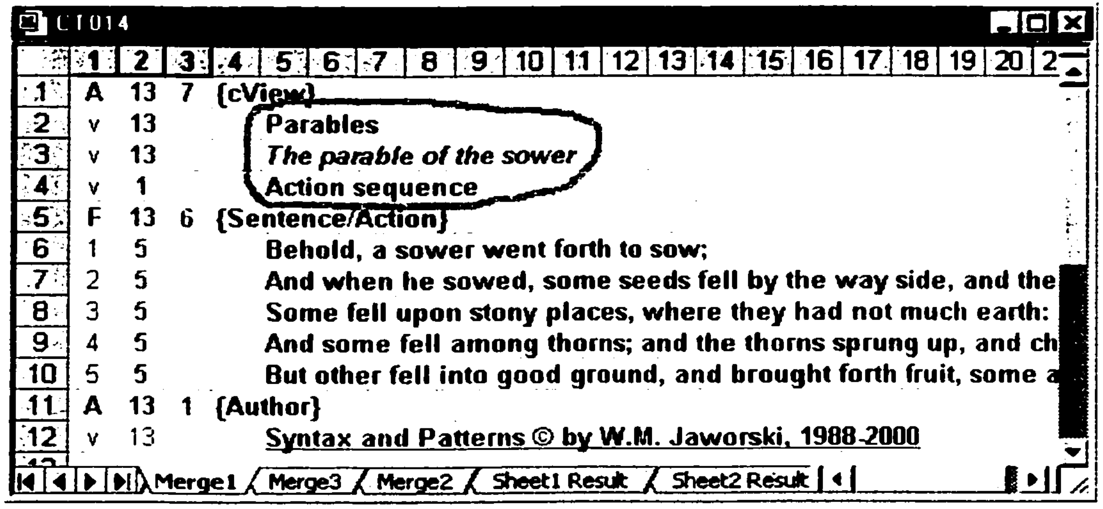
Figure A-27: Result For Input Sheet2 after Implementing 'Select Maps' Function

Join Maps------- List all the available worksheets for the current workbook and choose some worksheet to merge. The criterion for this function is: comparing the values of concept column within each selected worksheet, join the columns with the same values of concept. Or add addition rows with different values of concept. Then save the result into separate sheet named "Merge Result". This function can implement independently.

1. In Query Form, click the Jion Maps box.
2. Click "OK" button. It pops up a Worksheets List Form as Figure A-23.
3. Select worksheets, which should be merged from Worksheets List Form.
4. Click "OK" button in Worksheets List Form.
5. Produce a new worksheet named "Merge Result" to save the result.

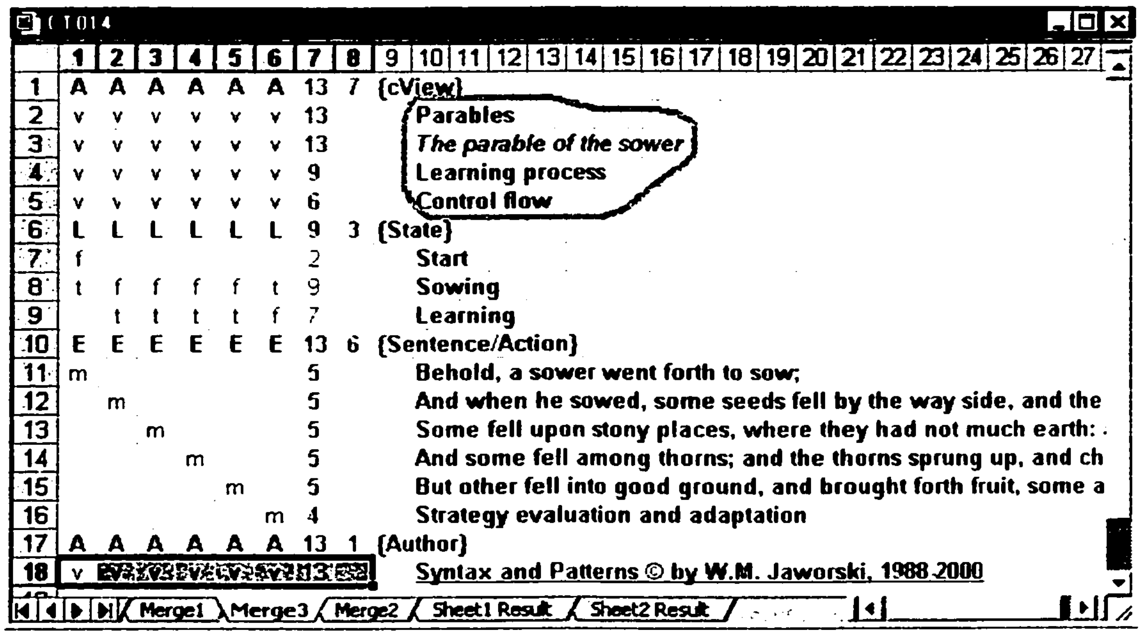
For example, Figure A-28, Figure A-29 and Figure A-30 are three input worksheets for Joining Maps. After implementation, an output worksheet, Figure A-31, will be automatically produced to save the corresponding result.

Notes: Look at all the sub-concept values under Concept {cView} in the figures.



	1	2	3	4	5
1	A	13	7	{cView}	
2	v	13			Parables
3	v	13			*The parable of the sower*
4	v	1			Action sequence
5	F	13	6	{Sentence/Action}	
6	1	5			Behold, a sower went forth to sow;
7	2	5			And when he sowed, some seeds fell by the way side, and the
8	3	5			Some fell upon stony places, where they had not much earth:
9	4	5			And some fell among thorns; and the thorns sprung up, and ch
10	5	5			But other fell into good ground, and brought forth fruit, some a
11	A	13	1	{Author}	
12	v	13			Syntax and Patterns © by W.M. Jaworski, 1988-2000

Figure A-28: Input Sheet Named "Merge1" for Joining Maps

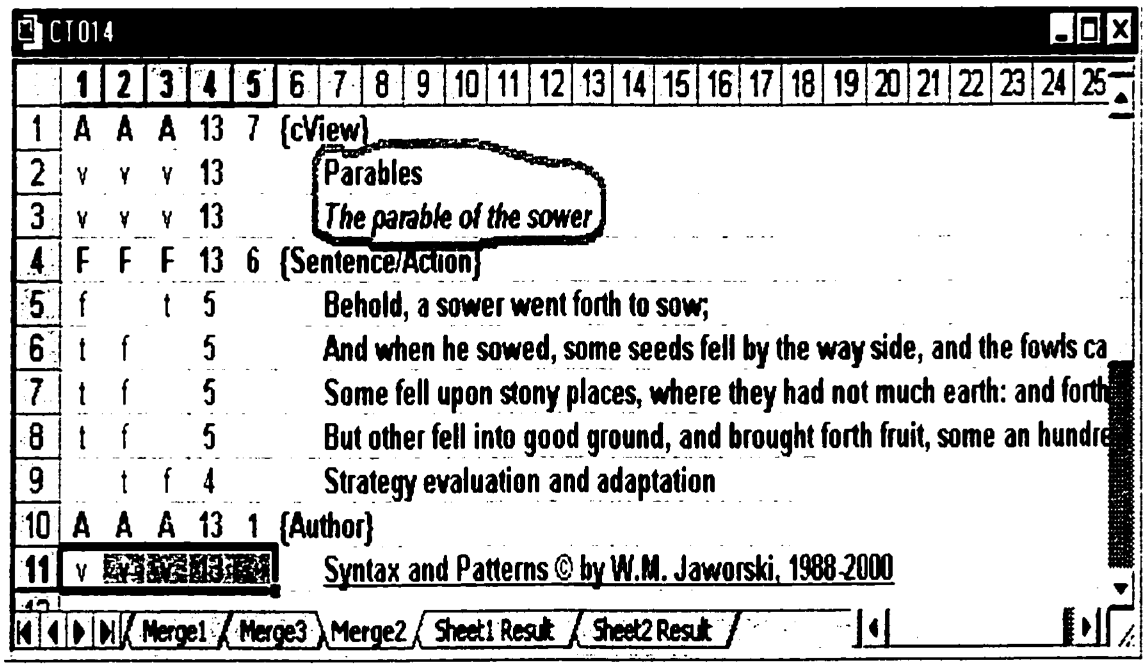


CT014

	1	2	3	4	5	6	7	8	9	10
1	A	A	A	A	A	A	13	7	{cView}	
2	v	v	v	v	v	v	13			Parables
3	v	v	v	v	v	v	13			*The parable of the sower*
4	v	v	v	v	v	v	9			Learning process
5	v	v	v	v	v	v	6			Control flow
6	L	L	L	L	L	L	9	3	{State}	
7	f						2			Start
8	t	f	f	f	f	t	9			Sowing
9		t	t	t	t	f	7			Learning
10	E	E	E	E	E	E	13	6	{Sentence/Action}	
11	m						5			Behold, a sower went forth to sow;
12		m					5			And when he sowed, some seeds fell by the way side, and the
13			m				5			Some fell upon stony places, where they had not much earth: .
14				m			5			And some fell among thorns; and the thorns sprung up, and ch
15					m		5			But other fell into good ground, and brought forth fruit, some a
16						m	4			Strategy evaluation and adaptation
17	A	A	A	A	A	A	13	1	{Author}	
18	v	[illegible]								Syntax and Patterns © by W.M. Jaworski, 1988-2000

Merge1 / Merge3 / Merge2 / Sheet1 Result / Sheet2 Result

Figure A-29: Input Sheet Named "Merge3" for Joining Maps

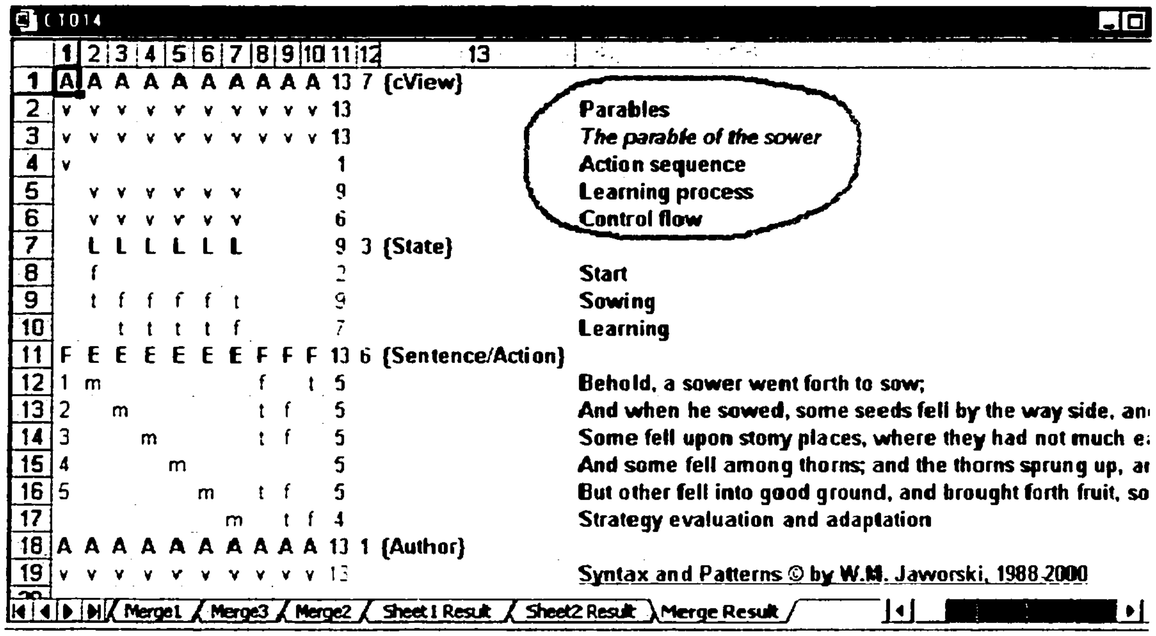


CT014

	1	2	3	4	5	6	7
1	A	A	A	13	7	{cView}	
2	v	v	v	13			Parables
3	v	v	v	13			*The parable of the sower*
4	F	F	F	13	6	{Sentence/Action}	
5	f		t	5			Behold, a sower went forth to sow;
6	t	f		5			And when he sowed, some seeds fell by the way side, and the fowls ca
7	t	f		5			Some fell upon stony places, where they had not much earth: and forth
8	t	f		5			But other fell into good ground, and brought forth fruit, some an hundre
9		t	f	4			Strategy evaluation and adaptation
10	A	A	A	13	1	{Author}	
11	v	[illegible]					Syntax and Patterns © by W.M. Jaworski, 1988-2000

Merge1 / Merge3 / Merge2 / Sheet1 Result / Sheet2 Result

Figure A-30: Input Sheet Named "Merge2" for Joining Maps



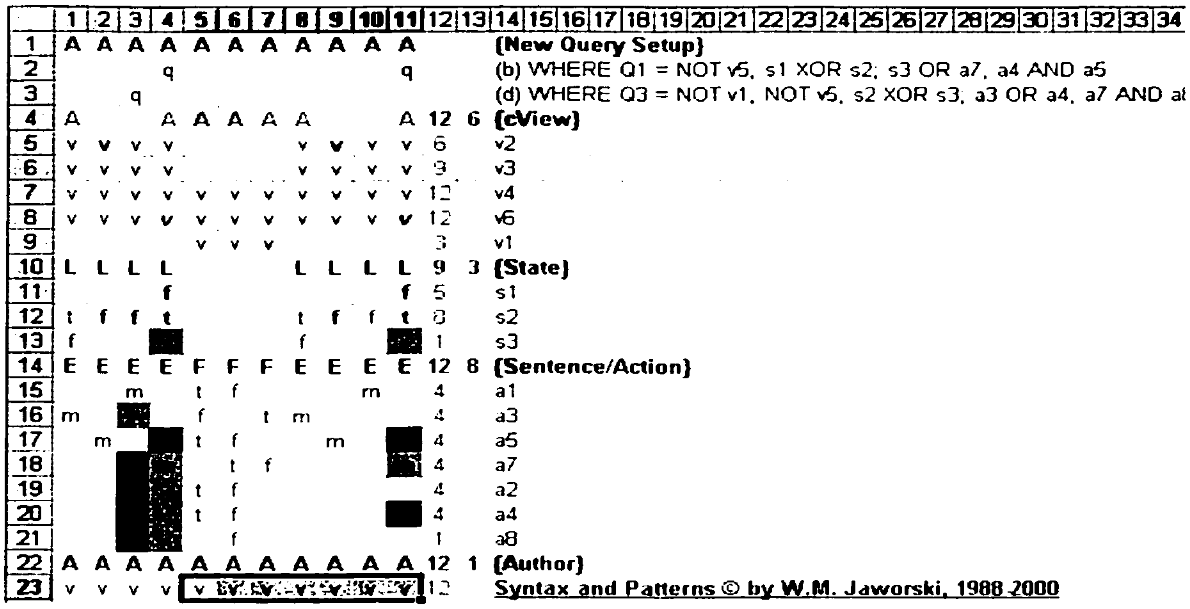
	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	A	A	A	A	A	A	A	A	A	A	13	7	{cView}	
2	v	v	v	v	v	v	v	v	v	v	13			**Parables**
3	v	v	v	v	v	v	v	v	v	v	13			***The parable of the sower***
4	v										1			**Action sequence**
5		v	v	v	v	v	v				9			**Learning process**
6		v	v	v	v	v	v				6			**Control flow**
7		L	L	L	L	L	L				9	3	{State}	
8		f									2			**Start**
9		t	f	f	f	f	t				9			**Sowing**
10			t	t	t	t	f				7			**Learning**
11	F	E	E	E	E	E	E	F	F	F	13	6	{Sentence/Action}	
12	1	m						f		t	5			**Behold, a sower went forth to sow;**
13	2		m					t	f		5			**And when he sowed, some seeds fell by the way side, an**
14	3			m				t	f		5			**Some fell upon stony places, where they had not much e**
15	4				m						5			**And some fell among thorns; and the thorns sprung up, a**
16	5					m		t	f		5			**But other fell into good ground, and brought forth fruit, so**
17							m		t	f	4			**Strategy evaluation and adaptation**
18	A	A	A	A	A	A	A	A	A	A	13	1	{Author}	
19	v	v	v	v	v	v	v	v	v	v	13			**Syntax and Patterns © by W.M. Jaworski, 1988-2000**

Figure A-31: Output Sheet Named "Merge Result" for Joining Maps

This function "Joining Maps" can also work with 'By Color' function while 'Select Maps' is selected. Under this circumstance, 'Select Maps' function will implement first, then 'By Color' function, finally it will join the results and save into a new sheet Named 'Merge Result'. There is an example.

1. Predefine query on Figure A-24 Input Sheet1 and Figure A-25 Input Sheet2.
2. From Query Form, buttons "Select Maps", "Join Maps", "By Color" and "AND" are selected.
3. Click "OK" button, pops up WorkSheets List Form.
4. Select "Sheet1" and "Sheet2", then click "OK" button in WorkSheets List Form.

5. Automatically add the two new worksheets "Sheet1 Result" and "Sheet2 Result" like Figure A-26 and Figure A-27.
6. Finally produce a new worksheet named "Merge Result", Figure A-32, after joining "Sheet1 Result" and "Sheet2 Result".



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	A	A	A	A	A	A	A	A	A	A	A			**{New Query Setup}**
2				q							q			(b) WHERE Q1 = NOT v5, s1 XOR s2; s3 OR a7, a4 AND a5
3			q											(d) WHERE Q3 = NOT v1, NOT v5, s2 XOR s3; a3 OR a4, a7 AND a
4	A			A	A	A	A	A			A	12	6	**{cView}**
5	v	v	v	v				v	v	v	v	6		v2
6	v	v	v	v				v	v	v	v	9		v3
7	v	v	v	v	v	v	v	v	v	v	v	12		v4
8	v	v	v	v	v	v	v	v	v	v	v	12		v6
9					v	v	v					3		v1
10	L	L	L	L				L	L	L	L	9	3	**{State}**
11				f							f	5		s1
12	t	f	f	t				t	f	f	t	8		s2
13	f			■				f			■	1		s3
14	E	E	E	E	F	F	F	E	E	E	E	12	8	**{Sentence/Action}**
15			m		t	f				m		4		a1
16	m		■		f		t	m				4		a3
17		m		■	t	f			m		■	4		a5
18			■	■		t	f				■	4		a7
19			■	■	t	f						4		a2
20			■	■	t	f					■	4		a4
21			■	■		f						1		a8
22	A	A	A	A	A	A	A	A	A	A	A	12	1	**{Author}**
23	v	v	v	v	v	v	v	v	v	v	v	12		<u>Syntax and Patterns © by W.M. Jaworski, 1988-2000</u>

Figure A-32: Result Sheet named "Merge Result" for 'Select Maps', 'By Color', 'Join Maps"

A.2.5.3 Output Section Functionality

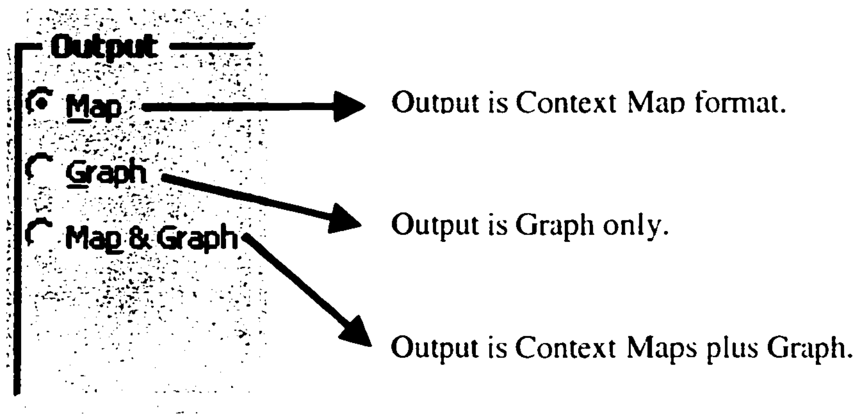

Output
[✓ Map]
Graph
Map & Graph
Output is Context Map format.
Output is Graph only.
Output is Context Maps plus Graph.

Figure A-33: Output from Query Form

The output format can be selected. However, in this report, all outputs are in Context Map forrmat. The Graph format is recommended for future work.

A.2.5.4 Run Section Functionality

Schema ------ Output Map is produced as Zoom-in Map, which is a work frame of Context Maps. This function is as same as 5.2.1 Show Schema.

Cardinality ------Compute the number of non-empty roles of each row and the number of sub-concept value of each Concept set. Detail to see 5.2.2 Compute Cardinality.

Apply Color ------This function is applying different colors for the map in spreadsheet based on the value of each cell. This function is as same as 5.2.3 Apply Color.

Help ------ This function is supplying a short version "Help", which has brief description for each button on the Query Form. This is also as same as 5.2.4 Help.

Appendix B---- Program Source Code

The program was coded by using VB language, the project consists of three parts: user forms, modules and class modules

A user form contains user interface controls, such as command buttons and text boxes

A module is a set of declarations followed by procedures—a list of functions that a program implements.

A class module defines an object, its properties, and its methods. A class module acts as a template from which an instance of an object is created at run time.

A. User Form Source Code

The source code for User Form includes as following created forms:

- **frmProjection**
- **frmSelectConcepts**
- **frmSelectWorksheets**

A.1 frmProjection

```
Option Explicit

Private m_Canceled As Boolean
Private m_SaveView As Boolean
Private m_NewSheet As Boolean
Private m_Type As Long
Private m_PromptForConcepts As Boolean
Private m_CheckVisible As Boolean
Private m_QueryColor As Boolean
Private m_AutoByColor As Boolean
Private m_SelectMaps As Boolean
Private m_JoinMaps As Boolean
Private m_Cardinality As Boolean
Private m_ApplyColor As Boolean

Public Property Get Canceled() As Boolean
 On Error Resume Next
 Canceled = m_Canceled
End Property

Public Property Let Canceled(flgCancel As Boolean)
 On Error Resume Next
 m_Canceled = flgCancel
End Property
```

```
Public Property Get SaveView() As Boolean
 On Error Resume Next
 SaveView = m_SaveView
End Property
Public Property Get CheckVisible() As Boolean
 On Error Resume Next
 CheckVisible = m_CheckVisible
End Property

Public Property Get SelectMap() As Boolean
 On Error Resume Next
 SelectMap = m_SelectMaps
End Property

Public Property Get QueryColor() As Boolean
 On Error Resume Next
 QueryColor = m_QueryColor
End Property

Public Property Get AutoByColor() As Boolean
 On Error Resume Next
 AutoByColor = m_AutoByColor
End Property

Public Property Get JoinMap() As Boolean
 On Error Resume Next
 JoinMap = m_JoinMaps
End Property

Public Property Get Cardinality() As Boolean
 On Error Resume Next
 Cardinality = m_Cardinality
End Property

Public Property Get ApplyColor() As Boolean
 On Error Resume Next
 ApplyColor = m_ApplyColor
End Property

Public Property Get NewSheet() As Boolean
 On Error Resume Next
 NewSheet = m_NewSheet
End Property


Public Property Get PromptForConcept() As Boolean
 On Error Resume Next
 PromptForConcept = m_PromptForConcepts
End Property

Public Property Get ProjectionType() As Long
 On Error Resume Next
 ProjectionType = m_Type
End Property

Private Sub chkAutoByColor_Click()

 If Not chkSelectMaps.Value Then
  If chkAutoByColor.Value Then
     If chkJoinMaps.Value Then
        MsgBox "You selected By Color. Join Maps can't be selected at this time"
        chkJoinMaps.Value = False
```

```
      End If
    End If
  End If
End Sub

Private Sub chkJoinMaps_Click()
  If Not chkSelectMaps.Value Then
   If chkJoinMaps.Value Then
      If chkAutoByColor.Value Then
        MsgBox "You selected Join Maps. By Color can't be selected at this time"
        chkAutoByColor.Value = False
      End If
   End If
  End If
End Sub

Private Sub chkSelectMaps_Click()

  If chkSelectMaps.Value = False Then
    If chkAutoByColor.Value And chkJoinMaps.Value Then
      MsgBox "You can't select Join Maps and By Color at the same time"
      chkAutoByColor.Value = False
    End If
  End If
End Sub

Private Sub cmdCancel_Click()

  On Error Resume Next
  m_Canceled = True
  Call Me.Hide

End Sub

Private Sub cmdOK_Click()

  On Error Resume Next

  m_PromptForConcepts = chkPromptForConcepts.Value
  m_CheckVisible = chkVisible.Value
  m_AutoByColor = chkAutoByColor.Value
  m_SelectMaps = chkSelectMaps.Value
  m_JoinMaps = chkJoinMaps.Value
  m_Cardinality = optCardinality.Value
  m_ApplyColor = optApplyColor.Value

  If optTypeAnd.Value Then
          m_Type = pjtAND

  ElseIf optTypeOr.Value Then
          m_Type = pjtOR

  ElseIf optTypeXor.Value Then
          m_Type = pjtXOR

  Else
          m_Type = pjtNORMAL

  End If

   'Save the frmProjection Setting
SaveAlphaSetting REG_KEY_INTERACTIVE_PROJECTION_TYPE, CStr(m_Type)
```

```
  SaveAlphaSetting REG_KEY_INTERACTIVE_PROJECTION_PROMPT_FOR_CONCEPTS.
           CLng(m_PromptForConcepts)
  SaveAlphaSetting REG_KEY_Check_Visible_OUTPUT, CLng(m_CheckVisible)
  SaveAlphaSetting REG_KEY_Query_Color_OUTPUT, CLng(m_QueryColor)
  SaveAlphaSetting REG_KEY_Auto_ByColor_OUTPUT, CLng(m_AutoByColor)
  SaveAlphaSetting REG_KEY_SelectMap_OUTPUT, CLng(m_SelectMaps)
  SaveAlphaSetting REG_KEY_JoinMap_OUTPUT, CLng(m_JoinMaps)
  SaveAlphaSetting REG_KEY_Cardinality_OUTPUT, CLng(m_Cardinality)
  SaveAlphaSetting REG_KEY_ApplyColor_OUTPUT, CLng(m_ApplyColor)
  SaveAlphaSetting REG_KEY_INTERACTIVE_PROJECTION_OUTPUT, ipoNONE

  m_Canceled = False
  Call Me.Hide

End Sub

Private Sub OptHelp_Click()

'Option Button Help click, showing the help sheet
   Sheet10.Activate
   m_Canceled = True
      Call Me.Hide

End Sub

Private Sub UserForm_initialize()

  On Error Resume Next

  Select Case GetAlphaSetting(REG_KEY_INTERACTIVE_PROJECTION_TYPE.
REG_DEF_INTERACTIVE_PROJECTION_TYPE)
     Case pjtAND
        optTypeAnd.Value = True
     Case pjtOR
        optTypeOr.Value = True
     Case pjtXOR
        optTypeXor.Value = True
     Case Else
        optTypeNormal.Value = True
  End Select

   chkPromptForConcepts.Value =
          CBool(GetAlphaSetting(REG_KEY_INTERACTIVE_PROJECTION_PROMPT_FOR_CONCEPTS,
          REG_DEF_INTERACTIVE_PROJECTION_PROMPT_FOR_CONCEPTS))

  chkVisible.Value = CBool(GetAlphaSetting(REG_KEY_Check_Visible_OUTPUT,
          REG_DEF_Check_Visible_OUTPUT))

  chkAutoByColor.Value = CBool(GetAlphaSetting(REG_KEY_Auto_ByColor_OUTPUT,
          REG_DEF_Auto_ByColor_OUTPUT))

  chkSelectMaps.Value = CBool(GetAlphaSetting(REG_KEY_SelectMap_OUTPUT,
          REG_DEF_SelectMap_OUTPUT))

  chkJoinMaps.Value = CBool(GetAlphaSetting(REG_KEY_JoinMap_OUTPUT,
          REG_DEF_JoinMap_OUTPUT))

  optCardinality.Value = CBool(GetAlphaSetting(REG_KEY_Cardinality_OUTPUT,
          REG_DEF_Cardinality_OUTPUT))

   optApplyColor.Value = CBool(GetAlphaSetting(REG_KEY_ApplyColor_OUTPUT,
          REG_DEF_ApplyColor_OUTPUT))
```

```
End Sub
```

A.2 frmSelectConcepts

* Elide the detail source code for this FORM.

A.3 frmSelectWorksheets

* Elide the detail source code for this FORM.

B. Modules Source Code

The source code for Modules includes as following created modules:

- **mApplyColor**
- **mAutoByColor**
- **mMerge**
- **mProjection**

B.1 mApplyColor

```
'Purpose of this model is applying different background color according to the cells' value

Option Private Module

Public Sub ApplyColor()


Dim mySheet As Excel.Range

'Check if it's concept set value
 CheckBold

'Defined the color index

 AutoRAqua = RGB(60, 195, 238)
 AutoN = RGB(230, 160, 162)
 AutoELime = RGB(153, 192, 46)
 AutotGreen = RGB(0, 251, 0)
 AutoFRed = RGB(255, 0, 0)
 AutomLightOrg = RGB(222, 150, 51)
 AutoPink = RGB(255, 0, 255)
 AutoYBarkblue = RGB(107, 7, 184)
 Autopaleblue = RGB(153, 204, 255)
 AutoLightPink = RGB(255, 166, 205)
 AutovYellow = RGB(238, 207, 0)
 AutoVDarkYellow = RGB(226, 178, 0)
 AutoSBrightYellow = RGB(255, 255, 0)
 AutoAGray = RGB(128, 128, 128)
 AutoXLightPurple = RGB(204, 137, 255)
 AutoLPurple = RGB(255, 0, 255)
 AutoGDarkGreen = RGB(0, 95, 0)
 AutobDarkPurple = RGB(128, 0, 128)
 AutoBrightBlue = RGB(0, 204, 255)
 AutoUPale = RGB(192, 192, 192)
 AutoDRed = RGB(194, 61, 87)
 AutoCGreen = RGB(62, 193, 104)
 AutoOgray = RGB(123, 123, 123)


'wmj
```

```
   AutoAqua = RGB(60, 186, 196)
   AutoELime = RGB(153, 178, 51)
   AutoGreen = RGB(0, 251, 0)
   AutoRed = RGB(255, 0, 0)
   AutoLightOrg = RGB(222, 144, 51)
   AutoPink = RGB(255, 0, 255)
   Autopaleblue = RGB(153, 204, 255)
   AutoLightPink = RGB(255, 166, 205)
   AutoYellow = RGB(238, 192, 65)
   AutoBrightYellow = RGB(255, 255, 0)
   Autogray = RGB(128, 128, 128)
   AutoLightPurple = RGB(204, 137, 255)
   AutoPurple = RGB(255, 0, 255)
   AutoDarkGreen = RGB(0, 95, 0)
   AutoBrightBlue = RGB(0, 204, 255)
   'wmj

 Set mySheet = ActiveSheet.UsedRange

 With mySheet
    Set c = .Find("A", lookat:=xlWhole, MatchCase:=True)
    If Not c Is Nothing Then
       firstAddress = c.Address
       Do
          c.Interior.Color = AutoAGray
          Set c = .FindNext(c)
       Loop While Not c Is Nothing And c.Address <> firstAddress
       End If

  'wmj
   Set c = .Find("{", LookIn:=xlValue, lookat:=xlPart)
     If Not c Is Nothing Then
      firstAddress = c.Address
      Do
         c.Interior.colorIndex = 15
         Set c = .FindNext(c)
      Loop While Not c Is Nothing And c.Address <> firstAddress
       End If
  'wmj

  ** Elide the detail code for applying different background color

  End With
  For Each c In mySheet.Range(Cells(1, ConceptCol - 2), Cells(mySheet.rows.count,
           ConceptCol - 1))

              If IsNumeric(c.Value) Then
          c.Font.colorIndex = 3
        End If
  Next

End Sub


Private Sub CheckBold()

 Dim wks As Worksheet
 Dim rng As Range
 Dim RowCount As Long
 Dim ColCount As Long
 Dim rowIndex As Long
 Dim colIndex As Long
```

```
  Set wks = ActiveSheet

  With wks.Cells.SpecialCells(xlLastCell)
    RowCount = .Row
    ColCount = .Column
  End With

  'Search for the Concept Column within whole sheet

  For rowIndex = 1 To RowCount
         For colIndex = ColCount To 1 Step -1

      If IsConceptLabel(wks.Cells(rowIndex, colIndex).Value) Then
        wks.rows(rowIndex).Font.Bold = True
        Exit For
      End If

    Next colIndex
  Next rowIndex

End Sub
```

B.2 mAutoByColor

'Purpose of this model is using for implementing ByColor function 'based on predefined query.

```
        Const BLOCK_ALLOCATION = 100

        Dim flgArray() As Boolean
        Dim addSheet() As Integer
        Dim addSheetCount As Integer
        Dim currColumn As Long
        Dim currRow As Long
        Dim flgFirst As Long
        Dim flgPredefined As Boolean

        Private Sub AutoQueryByColor(wks As Worksheet, ByVal pnRows_IN As Long, ByVal pnColMax_IN As
        Long, panColumns_OUT() As Long)

        Dim I, J, RowCount, ColCount As Integer
        Dim totalCount As Integer
        Dim lngColumns() As Long
        Dim lngRows() As Long
        Dim alngColumns() As Long
        Dim nRowLast, nColLast, notCol As Long
        Dim flgFunction As Boolean
        Dim flgAnd As Boolean
        Dim flgXor As Boolean
        Dim flgOr As Boolean

        Set wks = ActiveSheet
        totalCount = 0

        ReDim lngRows(1 To pnRows_IN)

        'Check the Red color which means there needs AND operation
        For I = 1 To pnRows_IN

              If wks.Cells(I, currColumn).Interior.colorIndex = 3 Then
               totalCount = totalCount + 1
```

```
            lngRows(totalCount) = I
            flgFunction = True
           End If

    Next I

    'Save the result of And operation into array
    If flgFunction = True Then

       ReDim Preserve lngRows(1 To totalCount)
       SPAndHideCols wks, lngRows, pnColMax_IN, panColumns_OUT()
          flgAnd = True

    End If

    totalCount = 0

    flgFunction = False
    ReDim lngRows(1 To pnRows_IN)

    'Check the Yellow color which means there needs XOR operation
    For I = 1 To pnRows_IN

      If wks.Cells(I, currColumn).Interior.colorIndex = 6 Then
          totalCount = totalCount + 1
          lngRows(totalCount) = I
          'pnColMax_IN = j
          flgFunction = True
      End If

    Next I

    'Save the result of XOR operation into array
    If flgFunction = True Then

       ReDim Preserve lngRows(1 To totalCount)
       SPXorHideCols wks, lngRows, pnColMax_IN, panColumns_OUT()
          flgXor = True

    End If

    flgFunction = False
    totalCount = 0

    ReDim lngRows(1 To pnRows_IN)

    'Check the Green color which means there needs OR operation
    For I = 1 To pnRows_IN
      If wks.Cells(I, currColumn).Interior.colorIndex = 4 Then
          totalCount = totalCount + 1
          lngRows(totalCount) = I
          flgFunction = True
      End If
    Next I

    'Save the result of OR operation into array
    If flgFunction = True Then
      ReDim Preserve lngRows(1 To totalCount)
      SPOrHideCols wks, lngRows, pnColMax_IN, panColumns_OUT()
          flgOr = True
    End If
```

```
totalCount = 0
flgFunction = False

ReDim lngRows(1 To pnRows_IN)

'Check the Blue color which means there needs NOT operation
For I = 1 To pnRows_IN
   If wks.Cells(I, currColumn).Interior.colorIndex = 41 Then
       totalCount = totalCount + 1
       lngRows(totalCount) = I
       notCol = I
       flgFunction = True
   End If
Next I

'Save the result of NOT operation into array
If flgFunction = True Then
  ReDim Preserve lngRows(1 To totalCount)
   SPNotHideCols wks, lngRows, pnColMax_IN, panColumns_OUT()
End If

End Sub

Private Sub SPAndHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As
Long, panColumns_OUT() As Long)

 Dim nColCurr As Long
 Dim nColumnsIndex As Long
 Dim nColumnsUpperBound As Long
 Dim nRowsIndex As Long
 Dim nRowsUpperBound As Long
 Dim nRowsLowerBound As Long
 Dim bFoundBlank As Boolean

 nColumnsUpperBound = BLOCK_ALLOCATION
 ReDim panColumns_OUT(1 To nColumnsUpperBound)
 nColumnsIndex = 0
 nRowsUpperBound = UBound(panRows_IN)
 nRowsLowerBound = LBound(panRows_IN)

  With wks
        ' find next stating Column which is not related to the concept
        For nColCurr = addSheetCount + 1 To pnColMax_IN

      bFoundBlank = False
      For nRowsIndex = nRowsLowerBound To nRowsUpperBound
         If Len(.Cells(panRows_IN(nRowsIndex), nColCurr)) = 0 Then
            bFoundBlank = True
            Exit For
                   End If
        Next nRowsIndex
      If bFoundBlank Then
         ' Hide the column
                   flgArray(flgFirst, nColCurr) = False

      ' All cells full
      Else
                   'store the Column number into an array

                   nColumnsIndex = nColumnsIndex + 1

         If nColumnsIndex > nColumnsUpperBound Then
```

```
            nColumnsUpperBound = nColumnsUpperBound + BLOCK_ALLOCATION
            ReDim Preserve panColumns_OUT(1 To nColumnsUpperBound)
                  End If
          panColumns_OUT(nColumnsIndex) = nColCurr
          flgArray(flgFirst, nColCurr) = True
        End If
    Next nColCurr
    'Resize the array Column

    If nColumnsIndex > 0 Then
        ReDim Preserve panColumns_OUT(1 To nColumnsIndex)
    Else
        Erase panColumns_OUT
          End If

  End With

End Sub

Private Sub SPXorHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As
Long, panColumns_OUT() As Long)

Private Sub SPNotHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As
Long, panColumns_OUT() As Long)

Private Sub SPOrHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As
Long, panColumns_OUT() As Long)

*Elide the detail source code for these three function

Public Sub startMultiQuery(wks As Worksheet, objSelection As Object, ByVal flgSelect As Boolean, ByVal
sheetNo As Long, ByVal flgConcept As Boolean, ByVal colConcept As Long, ByVal rows As Long, ByVal
cols As Long, ByVal operationType As Long, cols_OUT() As Long)

  Dim I, J, K, colNoStart, colNoEnd As Long
  Dim objArea As Range
  Dim currSelection As Range
  Dim selectedCol As Long
  Dim flgBlank As Boolean
  Dim Temp As Integer
  Dim queryColCount As Long
  Dim colCollection() As Long
  Dim BlankProjection As Boolean

  Set wks = ActiveSheet
  addSheetCount = 0
  flgFirst = 0

  Set objArea = objSelection
  selectedCol = objArea.Areas.count
   ReDim flgArray(1 To selectedCol, 1 To cols)
   ReDim colCollection(1 To selectedCol)
'check the selection is valid for the criteria which means the 'selected column has to have Red or Yellow or
Blue or Green 'Color

   If Not valBycolorParameter(wks, objArea, colConcept) Then
       Exit Sub
    End If

       queryColCount = 1

   For Each currSelection In objArea.Areas
```

```
        If (objArea.Cells.Columns.count <> 1) Or (objArea.Cells.rows.count <> 1)
                                        Then
                MsgBox "Each selection must be a single cell.", vbInformation +
                                vbOKOnly, PROC_DISPLAY_NAME
          frmProjection.Canceled = True
          Exit Sub
        End If
        currColumn = currSelection.Column
        currRow = currSelection.Row
        If flgPredefined = True Then
          For I = 1 To cols
            If LCase(wks.Cells(currRow, I)) = "q" Then
              currColumn = I
                        Exit For
            End If
          Next I
        End If
        flgFirst = flgFirst + 1
        colCollection(flgFirst) = currColumn
        Call AutoQueryByColor(wks, rows, cols, cols_OUT())
        flgArray(queryColCount, currColumn) = True
      Next

    'Hide columns for 4 operations
      For J = 1 To cols
        flgBlank = False
        Temp = 0
        If selectedCol > 1 Then
          For I = 1 To selectedCol
           If J = colCollection(I) Then
             flgBlank = True
             Exit For
           Else
            Select Case operationType
              Case pjtOR
                If flgArray(I, J) Then
                  flgBlank = True
                  Exit For
                Else
                  flgBlank = False
                  'wks.Columns(J).Hidden = True
                  'Exit For
                End If
              Case pjtAND
                If Not flgArray(I, J) Then
                  flgBlank = False
                  Exit For
                Else
                  flgBlank = True
                End If
              Case pjtXOR
                flgBlank = True
                If flgArray(I, J) Then Temp = Temp + 1
                If Temp = selectedCol Or Temp = 0 Then flgBlank = False
              Case pjtNORMAL
                flgBlank = True
                If flgArray(I, J) Then
                  flgBlank = False
                  Exit For
                Else
                  flgBlank = True
                End If
```

```
                  If wks.Cells(I, J).Value = "Q" Then
                     flgBlank = False
                     Exit For
                  End If

              End Select
            End If
           Next I
           If flgBlank = False Then
              wks.Columns(J).Hidden = True
            End If
        Else
          If J = currColumn Then
                        wks.Columns(J).Hidden = False
          ElseIf Not flgArray(I, J) Then
            wks.Columns(J).Hidden = True
                  End If
        End If
     Next J

'check the gray color for the concept value
    If Not flgConcept Then
       flgBlank = False
       For I = 1 To rows
        If IsConceptLabel(wks.Cells(I, colConcept)) Then
                   For J = 1 To colConcept
           If wks.Cells(I, J).Interior.colorIndex = 15 Or InStr(wks.Cells(I, colConcept).Value, "Query") > 0
Then
             flgBlank = False
                        Exit For

           Else
             flgBlank = True
           End If
                Next J
         If flgBlank = True Then wks.rows(I).Hidden = True
        Else
         If flgBlank = True Then wks.rows(I).Hidden = True
        End If
       Next I
       End If

   ReDim cols_OUT(0)
       J = 0
       For I = 1 To colConcept - 3
        If Not wks.Columns(I).Hidden Then
         J = J + 1
         ReDim Preserve cols_OUT(0 To J)
         cols_OUT(J) = I
            End If
       Next I
    ReDim Preserve cols_OUT(0 To J)
End Sub

Public Function PromptWorkSheets(ByRef colSelectedSheets As Collection) As Boolean

    Dim nSheetIndex As Long
    Dim colAllSheets As Collection
    On Error Resume Next
    Set colAllSheets = New Collection
    With ActiveWorkbook
```

```
        For nSheetIndex = 1 To .sheets.count
          colAllSheets.Add (.sheets.item(nSheetIndex).Name)
        Next nSheetIndex
    End With

    If Not SelectWorkSheet(colAllSheets, colSelectedSheets) Then
        Exit Function
    End If
        PromptWorkSheets = True
End Function

Private Function valBycolorParameter(ByRef wks As Worksheet, ByRef objSelected As Object, ByVal
ConceptColumn As Long) As Boolean

  Dim nAreaCount As Long
  Dim rng As Range
  Dim rngCellCurr As Range
  Dim nColInstance As Long
  Dim colorIndex As Long
  Dim strQuery As String
  On Error Resume Next
  flgPredefined = False
  Set rng = objSelected
  nAreaCount = rng.Areas.count
  'Check that the selection consists of non empty cells in the same column

  For Each rngCellCurr In rng.Areas
     If (rng.Cells.Columns.count <> 1) Or (rng.Cells.rows.count <> 1) Then
        MsgBox "Each selection must be a single cell.", vbInformation + vbOKOnly,
PROC_DISPLAY_NAME
        Exit Function
     End If
      If nColInstance = 0 Then
        nColInstance = rngCellCurr.Column

        If rngCellCurr.Column = ConceptColumn Then
          nColInstance = 0
        End If

     ElseIf nColInstance = rngCellCurr.Column Then
        MsgBox "The selected cells must all be in the same column.", vbInformation + vbOKOnly,
PROC_DISPLAY_NAME
        Exit Function
     End If

        'check interior color
        colorIndex = wks.Cells(rngCellCurr.Row, rngCellCurr.Column).
                     Interior.colorIndex
        strQuery = UCase(wks.Cells(rngCellCurr.Row, ConceptColumn))
        If rngCellCurr.Column = ConceptColumn Then
      If InStr(strQuery, "AND") <> 0 Or InStr(strQuery, "XOR") <> 0 Or InStr(strQuery, "OR") <> 0 Or
InStr(strQuery, "NOT") = 0 Then
        flgPredefined = True
      End If
        ElseIf colorIndex <> 41 And colorIndex <> 3 And colorIndex <> 4 And colorIndex <> 6 Then

      If InStr(strQuery, "AND") = 0 And InStr(strQuery, "XOR") = 0 And InStr(strQuery, "OR") = 0 And
InStr(strQuery, "NOT") = 0 Then

           MsgBox "Modified later The selected cells must cannot be empty.", vbInformation + vbOKOnly,
PROC_DISPLAY_NAME
        Exit Function
```

```
                End If
              End If
          Next rngCellCurr
          Set rngCellCurr = Nothing
          Set rng = Nothing
          valBycolorParameter = True
        End Function
```

B.3 mMerge

```
'Purpose of this module is implementing merging selected sheets 'into
 one sheet called Merge Result. It can either run by 'itself or work
 with Select Maps Funciton


   Public Sub Merge(joinSheets As Collection)

     Dim WS_Count As Integer
     Dim I As Integer
     Dim flgNewSheet As Boolean
     flgNewSheet = flase
     For I = 1 To ActiveWorkbook.Worksheets.count
       If ActiveWorkbook.Worksheets(I).Name = "Merge Result" Then
          sheets("Merge Result").Select
          Application.CutCopyMode = False
          Application.DisplayAlerts = False
          ActiveWindow.SelectedSheets.Delete
          Exit For
       End If
     Next I

    Dim wks As Worksheet
    Dim newWks As New Worksheet
    sheets(joinSheets(1)).Select
    Set wks = ActiveSheet
    wks.UsedRange.SpecialCells(xlCellTypeVisible).Copy
     Set newWks = ActiveWorkbook.Worksheets.Add(after:=Worksheets(Worksheets.count))
    newWks.Name = "Merge Result"
   With newWks
      .Paste
    End With
    Call JoinMaps(joinSheets)
  End Sub

  Private Sub JoinMaps(mergeSheets As Collection)

  Dim resultNO, wksNO, I, J, K As Long
  Dim sheetNo As Integer
  Dim resultRows As Long
  Dim resultCols As Long
  Dim resultColConcept As Long
  Dim rstRowIndex As Long
  Dim rstRowIndexDes As Long
  Dim wksRowIndex As Long
  Dim wksRowIndexDes As Long
  Dim wksRows As Long
  Dim wksCols As Long
  Dim wksColConcept As Long
  Dim Temp As Long
  Dim insertRow As Long
  Dim wks As Worksheet
Dim resultWks As Worksheet
Dim resultSet() As String
Dim wksSet() As String
Dim flgSetEqual As Boolean
Dim flgAtomEqual As Boolean

 Set resultWks = Worksheets("Merge Result")
For sheetNo = 2 To mergeSheets.count
'get the rows and columns number, the concept value column no.
    If Not FindDataExtent(resultWks, resultRows, resultCols, resultColConcept) Then
      Call MsgBox(ERM_UNKNOWN_DATA_EXTENT, vbInformation + vbOKOnly, "jMap Projection")
    End If
    Set wks = Worksheets(mergeSheets(sheetNo))
    If Not FindDataExtent(wks, wksRows, wksCols, wksColConcept) Then
      Call MsgBox(ERM_UNKNOWN_DATA_EXTENT, vbInformation + vbOKOnly, "jMap Projection")
    End If

'Saved the concept set value and its row number into array
    ReDim wksSet(2, 0)
    Temp = findConceptSet(wks, wksSet(), wksRows, wksColConcept)
    'Get the visible row number, col number and concept column no from the Select Maps result sheet
    wksRows = 0
    wksCols = 0
    For I = 1 To wks.UsedRange.rows.count
      If Not wks.rows(I).Hidden Then
        wksRows = wksRows + 1
      End If
    Next I

    For I = 1 To wks.UsedRange.Columns.count
      If Not wks.Columns(I).Hidden Then
        wksCols = wksCols + 1
        If IsConceptLabel(wks.Cells(1, I)) Then
          wksColConcept = I
          wksCols = wksCols - 3
          Exit For
        End If
      End If
     Next I

'Move the columns in merge result sheets
    resultWks.Range(Cells(1, resultCols + 1), Cells(resultRows, resultCols + 4)).Select
    Selection.Cut Destination:=Range(Cells(1, resultCols + wksCols + 1), Cells(resultRows, resultCols + wksCols
      + 4))

    'Find Set value and row number

    ReDim resultSet(2, 0)
    Temp = findConceptSet(resultWks, resultSet(), resultRows, resultColConcept + wksCols)
    insertRow = 0

    'Compare Concept Set Value
    For wksNO = 1 To UBound(wksSet, 2)
      flgSetEqual = False
       For resultNO = 1 To UBound(resultSet, 2)
         If wksSet(1, wksNO) = resultSet(1, resultNO) Then
           flgSetEqual = True
           Exit For
         End If
      Next resultNO
    wksRowIndex = wksSet(2, wksNO)
```

```
        If wksNO = UBound(wksSet, 2) Then
           wksRowIndexDes = wks.UsedRange.rows.count + 1
        Else
           wksRowIndexDes = wksSet(2, wksNO + 1)
        End If

'Equal
        If flgSetEqual Then
           rstRowIndex = resultSet(2, resultNO) + insertRow
           If resultNO = UBound(resultSet, 2) Then
              rstRowIndexDes = resultWks.UsedRange.rows.count + 1
           Else
              rstRowIndexDes = resultSet(2, resultNO + 1) + insertRow
           End If

           For J = wksRowIndex To wksRowIndexDes - 1
            flgAtomEqual = False

             If Not wks.rows(J).Hidden Then
                K = rstRowIndex
                Do While K < rstRowIndexDes
                   Debug.Print resultWks.Cells(K, resultColConcept + wksCols)
Debug.Print wks.Cells(J, wksColConcept)
                   If (resultWks.Cells(K, resultColConcept + wksCols) = wks.Cells(J, wksColConcept)) _
                     And resultWks.Cells(K, resultColConcept + wksCols + 1) = wks.Cells(J, wksColConcept + 1)
        Then
                     flgAtomEqual = True
                   End If

                   If flgAtomEqual Then
                      wks.Activate
                      wks.Range(Cells(J, 1), Cells(J, wksColConcept - 3)).SpecialCells(xlCellTypeVisible).Select
                      Selection.Copy
                      resultWks.Activate
                      resultWks.Cells(K, resultCols + 1).Select
                      K = K + 1
                      ActiveSheet.Paste
                      Exit Do
                   End If
                   K = K + 1
                Loop

                If Not flgAtomEqual Then
                  resultWks.Activate
                  resultWks.rows(rstRowIndexDes).Select
                  Selection.Insert shift:=xlDown
                  wks.Activate
                  wks.Range(Cells(J, 1), Cells(J, wksColConcept + 1)).SpecialCells(xlCellTypeVisible).Select
                  Selection.Copy
                  resultWks.Activate
                  resultWks.Cells(K, resultCols + 1).Select
                  ActiveSheet.Paste
                  insertRow = insertRow + 1
                  rstRowIndexDes = rstRowIndexDes + 1
                End If
              End If
           Next J
        Else
           rstRowIndex = resultSet(2, resultNO - 2) + insertRow
           resultWks.Activate
           For I = wksRowIndex To wksRowIndexDes - 1
             resultWks.rows(rstRowIndex).Select
```

```
                Selection.Insert shift:=xlDown
              Next I
              wks.Activate
              wks.Range(Cells(wksRowIndex, 1), Cells(wksRowIndexDes - 1, wksColConcept +
            1)).SpecialCells(xlCellTypeVisible).Select
              Selection.Copy
              resultWks.Activate
              resultWks.Cells(rstRowIndex, resultCols + 1).Select
              ActiveSheet.Paste
              insertRow = insertRow + wksRowIndexDes - wksRowIndex
              rstRowIndexDes = rstRowIndexDes + wksRowIndexDes - wksRowIndex
         End If
         Next wksNO
      Next sheetNo
      resultWks.Columns.ColumnWidth = 2
   End Sub

   Private Function findConceptSet(ByRef wks As Worksheet, ByRef setArray() As String, ByVal rowNo As Long,
            ByVal cCol As Long) As Long

   Dim index As Long
   Dim count As Long
   count = 0
   For index = 1 To rowNo
         If IsConceptLabel(wks.Cells(index, cCol)) Then
          If Not wks.rows(index).Hidden Then
            count = count + 1
            ReDim Preserve setArray(2, UBound(setArray, 2) + 1)
            setArray(1, count) = wks.Cells(index, cCol)
            setArray(2, count) = index
          End If
         End If
      Next index
    findConceptSet = count
End Function
```

B.4 mProjection

'Purpose of this module is retrieve the control value from the frmProjection form, and running And, 'Xor, Or, Not operation

```
Public UserMap As Worksheet
Public Const pjtNORMAL As Long = 0
Public Const pjtAND As Long = 1
Public Const pjtOR As Long = 2
Public Const pjtXOR As Long = 3
Public Const pjtANDNOT As Long = 4
Public Const pjtORNOT As Long = 5
Public Const pjtAutoColor As Long = 10
Public Const relopEQUAL = 0            ' =
Public Const relopGREATEREQUAL = 1     ' >=
Public Const relopLESSEQUAL = 2        ' <=
Public Const PROC_DISPLAY_NAME As String = "jMap Projection"

Public Sub InteractiveProjection(ByRef wks As Worksheet, ByRef objSelected As Object)
 Dim frmProjectionOptions As frmProjection
 On Error Resume Next
 Set frmProjectionOptions = New frmProjection
 Load frmProjectionOptions
 With frmProjectionOptions
        .Show
          If Not .Canceled Then
        ProjectjMap ActiveSheet.
Selection, .SaveView, .NewSheet, .PromptForConcept, .CheckVisible, .QueryColor, .AutoByColor, .SelectMap, .Join
Map, .Cardinality, .ApplyColor, .ProjectionType
          End If
  End With frmProjectionOptions
  Unload frmProjectionOptions
  Set frmProjectionOptions = Nothing
End Sub

Private Function ValidateProjectionParameters(ByRef wks As Worksheet, ByRef objSelected As Object, ByRef
ProjectionType_INOUT As Long) As Boolean

  Dim nAreaCount As Long
  Dim rng As Range
  Dim rngCellCurr As Range
  Dim nColInstance As Long
  On Error Resume Next
  'Type
  Select Case ProjectionType_INOUT
    Case pjtXOR, pjtAND, pjtOR, pjtNORMAL
    Case Else
      ProjectionType_INOUT = pjtNORMAL
  End Select

  'check if user select more than one area
  Set rng = objSelected
  nAreaCount = rng.Areas.count
  Select Case nAreaCount
'Check the selection validation
    Case Is < 1
      MsgBox "This macro does not support the specified selection.", vbInformation + vbOKOnly,
PROC_DISPLAY_NAME
      Exit Function
'Single Selection
    Case 1
      If ProjectionType_INOUT <> pjtOR Then
       ProjectionType_INOUT = pjtNORMAL
      End If
     'Multiple Selection
    Case Is > 1
      If ProjectionType_INOUT = pjtNORMAL Then

        MsgBox "This macro does not support multi selection. Please try either the 'And' or the 'Or' projection.",
vbInformation + vbOKOnly, PROC_DISPLAY_NAME
                      Exit Function
      End If
    End Select

  'Check that the selection consists of non empty cells in the same column
  For Each rngCellCurr In rng.Areas
          If (rng.Cells.Columns.count <> 1) Or (rng.Cells.rows.count <> 1) Then
      MsgBox "Each selection must be a single cell.", vbInformation + vbOKOnly, PROC_DISPLAY_NAME
      Exit Function
    End If
'Same Column
    If nColInstance = 0 Then
      nColInstance = rngCellCurr.Column
    End If
    If nColInstance <> rngCellCurr.Column Then
      MsgBox "The selected cells must all be in the same column.", vbInformation + vbOKOnly,
PROC_DISPLAY_NAME
      Exit Function
    End If
'Non Empty
    If Len(wks.Cells(rngCellCurr.Row, rngCellCurr.Column).Value) <= 0 Then
      MsgBox "The selected cells must cannot be empty.", vbInformation + vbOKOnly, PROC_DISPLAY_NAME
      Exit Function
    End If
  Next rngCellCurr
  ValidateProjectionParameters = True
End Function

Public Function ProjectjMap(ByRef wks As Worksheet, ByRef obj As Object, Optional ByVal SaveView As Boolean
= False, Optional ByVal PasteToNewSheet As Boolean = False, Optional ByVal PromptForConcepts As Boolean =
False, _
                Optional ByVal CheckVisibles As Boolean = False, Optional ByVal QueryColors As Boolean = False,
Optional ByVal AutoByColors As Boolean = False, Optional ByVal SelectMaps As Boolean = False, _
                Optional ByVal JoinMaps As Boolean = False, Optional ByVal getCardinality As Boolean = False,
Optional ByVal ApplyingColor As Boolean = False, _
                Optional ByVal ProjectionType As Long = pjtNORMAL, Optional ByVal SuppressUserInteraction
As Boolean = False) As Boolean

  Dim nRowCurr As Long
  Dim nColCurr As Long              'Row & Col Pointers
  Dim nColConcept As Long            'Concept col
  Dim strViewName As String          'Input provided by the user
  Dim nColMax As Long
  Dim nRowMax As Long                'Last col and row in the map
  Dim alngColumns() As Long           'Working cols
  Dim alngRows() As Long             'Working Rows
  Dim rngSelection As Range
  Dim rngCurr As Range
  Dim nRangeCurr As Long             'Multiple selection place holders
  Dim svi As CSuspendVisualInterface
  Dim selectedWorkSheets As Collection
  Dim resultSheets As Collection
  Dim nIndex As Long
  Dim nParentConceptRow As Long        'Parent Concept Row for Concept Selection
  Dim asParentConcepts() As String
  Dim bBlankProjection As Boolean
  Dim c As Range

 'Instance a class
 Set svi = New CSuspendVisualInterface
 svi.StatusBar = "Projecting..."
 If Not AutoByColors And Not JoinMaps Then
   If Not ValidateProjectionParameters(wks, obj, ProjectionType) Then
     Exit Function
       End If
 End If
 Set rngSelection = obj

 'get the rows and columns count number, and concept column number
  If Not FindDataExtent(ActiveSheet, nRowMax, nColMax, nColConcept) Then
   Call MsgBox(ERM_UNKNOWN_DATA_EXTENT, vbInformation + vbOKOnly, "jMap Projection")
   GoTo PROC_EXIT
  End If

'check the visible button from the frmProjection
If CheckVisibles <> True Then
  ShowAll wks
End If
With wks
  nRowCurr = rngSelection.Row
```

```
    'Check whether the ByColor button from the frmProjection is true
        If AutoByColors Then
         If SelectMaps Then
              'doing Select Maps
         Set selectedWorkSheets = New Collection
         Set resultSheets = New Collection
         Call PromptWorkSheets(selectedWorkSheets)
         If selectedWorkSheets.count > 0 Then
           For nIndex = 1 To selectedWorkSheets.count
              'Set wks = selectedWorkSheets(nIndex)
              sheets(selectedWorkSheets(nIndex)).Select
              Set wks = ActiveSheet
              ShowAll wks
              If Not FindDataExtent(wks, nRowMax, nColMax, nColConcept) Then
                 Call MsgBox(ERM_UNKNOWN_DATA_EXTENT, vbInformation + vbOKOnly, "jMap Projection")
                 GoTo PROC_EXIT
              End If
              svi.StatusBar = "MulitSheets ByColor ...."
              Call startMultiQuery(wks, Selection, SelectMaps, nIndex, False, nColConcept, nRowMax, nColMax,
ProjectionType, alngColumns())
              Call resultSheets.Add(wks.Name)
           Next nIndex
         End If
       'Select Maps and Join Maps work togetehr
         If JoinMaps Then
              svi.StatusBar = "Merging Selected Sheets ...."
              Call Merge(resultSheets)
         End If
       Else
'Select Maps only with ByColor
           svi.StatusBar = "MulitSheets ByColor ...."
           Call startMultiQuery(wks, obj, SelectMaps, nIndex, PromptForConcepts, nColConcept, nRowMax, nColMax,
ProjectionType, alngColumns())
       End If
         ElseIf SelectMaps = False And JoinMaps = True Then
         'Join Maps only without select Maps and ByColor
       Set selectedWorkSheets = New Collection
       Call PromptWorkSheets(selectedWorkSheets)
       svi.StatusBar = "Merging Selected ByColor ...."
       Call Merge(selectedWorkSheets)
Else
'Implementing AND, XOR, OR, NOT only with ByColor
    Select Case ProjectionType
        Case pjtAND, pjtOR
         ' Add all the rows to the array
          ReDim alngRows(1 To rngSelection.Areas.count)
           nRangeCurr = 1
          For Each rngCurr In rngSelection.Areas
             alngRows(nRangeCurr) = rngCurr.Row
             nRangeCurr = nRangeCurr + 1
          Next rngCurr
          ' Project
          If QueryColors <> True Then
            If ProjectionType = pjtAND Then
               SPAndHideCols wks, alngRows, nColMax, alngColumns()
            ElseIf ProjectionType = pjtXOR Then
               SPXorHideCols wks, alngRows, nColMax, alngColumns()
                 Else
               SPOrHideCols wks, alngRows, nColMax, alngColumns()
            End If
          Else
```

```
            If ProjectionType = pjtAND Then
              QuerySPAndHideCols wks, alngRows, nColMax, alngColumns()
            Else
              QuerySPOrHideCols wks, alngRows, nColMax, alngColumns()
            End If

          End If
        Case pjtXOR
          ReDim alngRows(1 To rngSelection.Areas.count)
          nRangeCurr = 1
          For Each rngCurr In rngSelection.Areas
             alngRows(nRangeCurr) = rngCurr.Row
             nRangeCurr = nRangeCurr + 1
           Next rngCurr
        If QueryColors <> True Then
          Call TestSPXorHideCols(wks, alngRows, nColMax, alngColumns())
        Else
          Call QuerySPXorHideCols(wks, alngRows, nColMax, alngColumns())
        End If

        Case Else
         If QueryColors <> True Then
           Call TestSPNotHideCols(wks, nRowCurr, nColMax, alngColumns())
         Else
           Call QuerySPNotHideCols(wks, nRowCurr, nColMax, alngColumns())
         End If
      End Select
    End If

    svi.StatusBar = "Projecting ...."

    'check the Select Sets value from frmProjection form
     If PromptForConcepts Then
         ReDim asParentConcepts(1 To rngSelection.Areas.count)
       nRangeCurr = 1
       For Each rngCurr In rngSelection.Areas
         If Not ConceptFromInstance(rngCurr.Row, wks, nColConcept, nParentConceptRow) Then
            MsgBox "Unable to locate the parent concept for the chosen instance(s). Please verify jMap integrity before
using this macro.", vbInformation + vbOKOnly, "jMap Projection"
            Exit Function
         End If
         asParentConcepts(nRangeCurr) = wks.Cells.item(nParentConceptRow, nColConcept).Value
         nRangeCurr = nRangeCurr + 1
       Next rngCurr

       ' Get the rows
     If Not SelectMaps Then
       If Not SPPromptForRows(wks, nColConcept, nRowMax, alngColumns(), asParentConcepts()) Then
         Exit Function
       End If
     End If
    Else
     'Hiding empty rows
       If ProjectionType <> pjtNORMAL Or AutoByColors Then
        If Not JoinMaps Then
          Call SPHideRows(wks, nColConcept, nRowMax, alngColumns(), bBlankProjection)
        End If
       End If
  End If
  End With

  'Calculate Cardinality
```

```
    If getCardinality Then
      Call ComputeCardinality(ActiveSheet)
    End If

'Applying Color
    If ApplyingColor Then
       svi.StatusBar = "Applying Color...."
      ApplyColor
    End If


  If bBlankProjection Then
     If Not SuppressUserInteraction Then
        If vbYes = NotifyMacroFinished("The selected instance(s) have produced a blank Projection. Do you want to
undo the projection?", PROC_DISPLAY_NAME, svi.StartTime, vbYesNo) Then
          ShowAll wks
        End If
     End If
  Else
     With wks
     Call .Activate
     ActiveWindow.ScrollColumn = 1
     'changing to use Map output to save the result
      If SaveView Then
          strViewName = Trim$(InputBox("Please enter a name for the jMap Projection: ", _
                        "Add jMap Projection to View Manager", _
                        .Cells(nRowCurr, nColMax + 4).Text + " View"))
          If Len(strViewName) > 0 Then
             Call ActiveWorkbook.CustomViews.Add(strViewName, True, True)
          End If
      End If
      If PasteToNewSheet Then
          'Copy the current jMap projection
          Selection.CurrentRegion.Select
          Selection.SpecialCells(xlCellTypeVisible).Select
          Selection.Copy
          'Deselect the selected cells
          .Cells(1, 1).Select
          'Create a new Workbook
          Dim wkbNew As Workbook
          Set wkbNew = Workbooks.Add
          'Using the new workbook
          With wkbNew.Worksheets(1)
             'Paste the projection into it
             .Paste
             'Make it look nice by adjusting column sizes
             .Columns.AutoFit
             'Fix the concept column so instances look good
             Call FindConceptColumn(wkbNew.Worksheets(1), nColConcept)
             .Columns(nColConcept).ColumnWidth = 1
             'Fix text and cell formatting
             Selection.WrapText = False
             Selection.Orientation = 0
             Selection.ShrinkToFit = False
             Selection.MergeCells = False
             'Deselect the sheet
             .Cells(1, 1).Select
          End With
          'Deselect the sheet and turn off copy mode
          Application.CutCopyMode = False
          'Fill in the groupings
          Call GroupjMap(ActiveSheet)
      End If
    End With

    If Not SuppressUserInteraction Then
      Call NotifyMacroFinished("Finished the Projection.", PROC_DISPLAY_NAME, svi.StartTime)
     End If
  End If

  ProjectjMap = True

PROC_EXIT:
   On Error Resume Next
   Erase alngColumns
   Erase alngRows
End Function

Public Sub SPHideRows(wks As Worksheet, ByVal colConcept As Long, ByVal nRowMax As Long, alColumns() As
Long, ByRef BlankProjection_OUT As Boolean)
'Purpose: hideBlankRow used by both projection and prijectionSave

  Dim nRowCurr As Long
  Dim nColCurr As Long
  Dim blnFoundNonBlank As Boolean
  Dim nColumnsStillVisible As Long
  Dim svi As CSuspendVisualInterface
  On Error Resume Next
  Set svi = New CSuspendVisualInterface
  svi.StatusBar = "Projection: Hiding Rows"
  nColumnsStillVisible = UBound(alColumns)

  If Err.Number <> 0 Then
    ActiveSheet.rows("1:" & CStr(nRowMax)).Hidden = True
    BlankProjection_OUT = True
    Exit Sub
  Else
    BlankProjection_OUT = False
  End If
  On Error GoTo 0
  With wks.Cells
' find blank row then hide the entire row
    For nRowCurr = 1 To nRowMax
      blnFoundNonBlank = False
      If Left(wks.Cells(nRowCurr, colConcept).Value, 1) = "Q" Then
        blnFoundNonBlank = True
      Else
       For nColCurr = 1 To nColumnsStillVisible
        If Len(.item(nRowCurr, alColumns(nColCurr)).Value) > 0 Then
          blnFoundNonBlank = True
          Exit For
        End If
       Next nColCurr
       If Not blnFoundNonBlank Then
        wks.rows(nRowCurr).Hidden = True
       End If
      End If
    Next nRowCurr
  End With
End Sub

Private Function SPPromptForRows(wks As Worksheet, ByVal colConcept As Long, ByVal nRowMax As Long,
alColumns() As Long, ParentConcepts() As String) As Boolean
```

```
'Purpose: hideBlankRow used by both projection and prijectionSave
 Dim nIndex As Long
 Dim nRowCurr As Long
 Dim nColCurr As Long
 Dim blnFoundNonBlank As Boolean
 Dim nColumnsStillVisible As Long
 Dim colConceptsRelated As Collection
 Dim colConceptsSelected As Collection
 Dim sConceptCurr As String
 Dim nConceptStart As Long
 Dim svi As CSuspendVisualInterface
 On Error Resume Next
  Set svi = New CSuspendVisualInterface
 svi.StatusBar = "Projection: Hiding Rows"
  nColumnsStillVisible = UBound(alColumns)
 If Err.Number <> 0 Then
   ActiveSheet.rows("1:" & CStr(nRowMax)).Hidden = True
   Exit Function
 End If
 Set colConceptsRelated = New Collection
 With wks.Cells
  For nRowCurr = 1 To nRowMax
      If IsConceptLabel(.item(nRowCurr, colConcept)) Then
         blnFoundNonBlank = False
         For nColCurr = 1 To nColumnsStillVisible
            If Len(.item(nRowCurr, alColumns(nColCurr)).Value) > 0 Then
               blnFoundNonBlank = True
               Exit For
            End If
          Next nColCurr
         If blnFoundNonBlank Then
            Call colConceptsRelated.Add(.item(nRowCurr, colConcept), UCase$(.item(nRowCurr, colConcept)))
          End If
      End If
   Next nRowCurr
 End With 'wks.Cells

 svi.ReEnableInterface
 If Not SelectConcepts(colConceptsRelated, colConceptsSelected) Then
   Exit Function
 End If
 svi.ReDisableInterface
  With wks.Cells
  ' find the first concept
   nRowCurr = 1
   Do Until IsConceptLabel(.item(nRowCurr, colConcept))
      nRowCurr = nRowCurr + 1
   Loop

   Do Until nRowCurr > nRowMax
      ' nRowCurr points to a concept row
      Err.Clear
      sConceptCurr = colConceptsSelected.item(UCase$(.item(nRowCurr, colConcept)))
       If Err.Number = 0 Then
        ' Skip the current concept
        nRowCurr = nRowCurr + 1
        ' find blank row then hide the entire row

        Do Until nRowCurr > nRowMax _Or IsConceptLabel(.item(nRowCurr, colConcept))
              blnFoundNonBlank = False
           For nColCurr = 1 To nColumnsStillVisible
              If Len(.item(nRowCurr, alColumns(nColCurr)).Value) > 0 Then
```

```
                blnFoundNonBlank = True
                Exit For
              End If
            Next nColCurr
            If Not blnFoundNonBlank Then
              wks.rows(nRowCurr).Hidden = True
            End If
            nRowCurr = nRowCurr + 1
          Loop

        ' Hiding the whole set for that concept
        Else
         ' Find the start of the next concept
          nConceptStart = nRowCurr
          nRowCurr = nRowCurr + 1
          Do Until nRowCurr > nRowMax _
            Or IsConceptLabel(.item(nRowCurr, colConcept))
            nRowCurr = nRowCurr + 1
          Loop
          ' Hide the current concept
          If nConceptStart = nRowCurr - 1 Then
            wks.rows(nConceptStart).Hidden = True
          Else
            wks.rows(CStr(nConceptStart) & ":" & CStr(nRowCurr - 1)).Hidden = True
          End If
        End If
      Loop
    End With
    SPPromptForRows = True
  End Function


  Public Sub ShowAll(ByRef wks As Worksheet)
    On Error Resume Next
    UnHideAll wks
  End Sub

  Private Sub SPAndHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

  Private Sub SPOrHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)


  Private Sub SPXorHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

  Private Sub QuerySPXorHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

  Private Sub TestSPNotHideCols(wks As Worksheet, ByRef panRows_IN As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

  Private Sub QuerySPAndHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

   Private Sub QuerySPOrHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)

  Private Sub QuerySPNotHideCols(wks As Worksheet, ByRef panRows_IN As Long, ByVal pnColMax_IN As Long,
  panColumns_OUT() As Long)
```

```
Private Sub TestSPXorHideCols(wks As Worksheet, ByRef panRows_IN() As Long, ByVal pnColMax_IN As Long,
panColumns_OUT() As Long)
```

- Elide the detail source code for these functions

C. Classes Source Code

The source code for Class includes as following:

- **CSuspendVisualInterface**

```
Option Explicit

Private mblnScreenUpdating As Boolean
Private mlngMousePointer As Long
Private mvarInitialStatusBar As Variant
Private msngStartTime As Single

Private Sub Class_Initialize()
  On Error Resume Next
    With Application
    ' Save the current state
    mlngMousePointer = .Cursor
    mblnScreenUpdating = .ScreenUpdating
    mvarInitialStatusBar = .StatusBar

    ' Set the wait state
    .Cursor = xlWait
    If CBool(GetAlphaSetting(REG_KEY_SCREEN_UPDATING_OFF, REG_DEF_SCREEN_UPDATING_OFF))
Then
      .ScreenUpdating = False
    End If
    .StatusBar = "Processing..."
    msngStartTime = timer
  End With
End Sub

Private Sub Class_Terminate()
  On Error Resume Next
  With Application
    .Cursor = mlngMousePointer
    .ScreenUpdating = mblnScreenUpdating
    If 0 = StrComp(CStr(mvarInitialStatusBar & ""), "FALSE", vbTextCompare) Then
      .StatusBar = False
    Else
      .StatusBar = mvarInitialStatusBar
    End If
  End With
End Sub

Public Sub ReEnableInterface()
  On Error Resume Next
  With Application
    .Cursor = xlDefault
    .ScreenUpdating = True
  End With
End Sub

Public Sub ReDisableInterface()
  On Error Resume Next
  DoEvents
```

```
    With Application
        .Cursor = xlWait
        .ScreenUpdating = False
    End With
End Sub

Public Property Get StatusBar() As String
    On Error Resume Next
         StatusBar = Application.StatusBar
 End Property

Public Property Let StatusBar(ByRef NewValue As String)
    On Error Resume Next
    Application.StatusBar = NewValue
End Property

Public Property Get StartTime() As Single
    On Error Resume Next
    StartTime = msngStartTime
End Property
```