

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

KNOWLEDGE MINING WITH CONTEXT MAPS

**CONVERTING OPEN PROCESS METHODOLOGY TO A
3P-ABLE FORMAT**

JING BAI

**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

© JING BAI, 2002



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-72926-5

Canada

Abstract

Knowledge Mining with Context Maps

Converting OPEN Process Methodology to a 3P-able Format

Jing Bai

Knowledge mining is a process to represent, analyze and extract information from knowledge base. There are many high-level representation technologies for knowledge mining. An advanced, powerful, and easy to implement technology named *Context Maps* is introduced as a common modeling language. This report presents the procedures of converting *OPEN Process* methodology to a 3P-able format by applying *Context Maps* technology. The functionalities and attributes of *Context Maps* are demonstrated by modeling *OPEN Process Framework* in terms of the concepts and relationships from “www.donald-firesmith.com” website. The implementation of this technology is illustrated by using MS Excel spreadsheet. The purpose of this report is to demonstrate that *Context Maps* is not only a formalized notation for representing knowledge, but also a powerful tool that can be used to manipulate the structured information.

Acknowledgement

I would like to express my special gratitude to all the people who gave me great help during this major report.

- I am greatly indebted to my supervisor, Professor Wojciech M. Jaworski. He gave me a lot of useful guidelines and encouraged my interest in knowledge mining field. With his patient advice and continuous help, I benefited from his theory and completed this major report successfully.
- I am pleased to thank Professor Peter Grogono for acceptance to be my co-supervisor. He gave me many valuable comments on the report.
- My sincere thanks are extended to Professor Olga Ormandjieva, who is an examiner for the report, for her helpful suggestions and support.
- My special thanks are also due to Ms. Monkiewicz Halina, the graduate program secretary, for her collaboration and help.

Finally, I would like to appreciate the faculties and staffs in Computer Science Department at Concordia University, who provided large support during my master program study.

Table of Contents

List of Figures	vii
List of Tables.....	viii
Permission Statement.....	ix
1. Introduction	1
1.1 Background	1
1.2 Outline of Report	2
1.3 Objective of Study	3
1.4 Procedure of Research	4
2. Context Maps.....	5
2.1 3P-able Notation	5
2.2 Context Maps Inception	6
2.3 Context Paradigm.....	7
2.4 Context Maps Syntax and Process	7
2.5 Context Maps Notation	11
2.6 Context Maps Tool	13
3. OPEN Process Framework.....	15
3.1 OPEN Process Framework Introduction.....	15
3.2 OPEN Process Framework Reusable Process Components.....	16
3.2.1 Definition.....	16
3.2.2 Repository Description	16
3.2.3 Component Metamodel	16
3.2.4 Component Hierarchy.....	17
3.2.5 Guidelines.....	19
3.3 OPEN Process Framework Principles	19
3.3.1 Flexibility	19
3.3.2 Industry Best Practices	20
3.3.3 Completeness.....	20
3.3.4 Usability	23
3.3.5 Reuse	23
3.4 OPEN Process Framework Underlying Theory.....	23
3.5 Process Construction Guidelines.....	27

3.5.1	Process Framework.....	27
3.5.2	Process Framework Repository	27
3.5.3	Metamodel, Framework, and Process.....	28
3.5.4	Construction Process	29
4.	Converting Open Process Framework to Context Maps.....	32
4.1	Procedure of Converting OPF to Context Maps.....	32
4.1.1	Identification of Sets.....	33
4.1.2	Enumeration of Set Members	34
4.1.3	Definition of Schema	42
4.1.4	Creation of Knowledge Tuples	42
4.1.5	Aggregation of Knowledge Tuples into Knowledge Views	42
4.2	Stages: Cycles and Phases View	45
4.3	Milestones: Business Strategy Phase View	50
4.4	Work Products View	58
4.5	Work Units View	62
4.6	Producers View	66
4.7	Advantages of Converting OPF to a 3P-able Format by Context Maps.....	70
5.	Comparing Open Process Framework with Rational Unified Process	73
5.1	Rational Unified Process Definition	73
5.2	Major Differences between OPF and RUP.....	73
5.2.1	Phase.....	74
5.2.2	Work Products	74
5.2.3	Terminology	78
5.3	OPEN Process Framework Strengths.....	79
5.4	RUP Weaknesses Addressed by OPF.....	79
6.	Conclusion and Recommendation	81
6.1	General Conclusions.....	81
6.2	Recommendations for Future Works	82
	Bibliography	84
	Appendix	86
A-1	Stages: Cycles and Phases View	86
A-2	Milestones: Business Strategy Phase View	89
A-3	Work Products View	92
A-4	Work Units View	95
A-5	Producers View	99

List of Figures

Figure 1	The <i>Context Maps</i> of Workflows Diagram in Unified Process.....	9
Figure 2	The <i>Context Maps</i> Notation	12
Figure 3	The Interface Diagram of “Query” Function.....	14
Figure 4	The <i>Context Maps</i> of OPF Component Metamodel	17
Figure 5	The <i>Context Maps</i> of OPF Layered Model	25
Figure 6	The <i>Context Maps</i> of Merging Figure 4 into Figure 5.....	26
Figure 7	The Metamodel, Framework, and Process of OPF.....	28
Figure 8	The <i>Context Maps</i> Schema of OPEN Process Framework	43
Figure 9	The Relationship Diagram of Cycles and Phases.....	48
Figure 10	The State Transition Diagram of Cycles and Phases.....	48
Figure 11	The <i>Context Maps</i> of Cycles and Phases	49
Figure 12	The Milestones Diagram of {Business Strategy Phase}.....	51
Figure 13	The State Machine of Milestones in {Business Strategy Phase}	51
Figure 14	The <i>Context Maps</i> of {Business Strategy Phase}	54
Figure 15	The <i>Context Maps</i> of {Work Products}.....	61
Figure 16	The <i>Context Maps</i> of {Work Units}	65
Figure 17	The <i>Context Maps</i> of {Producers}.....	69
Figure 18	The <i>Context Maps</i> of “Floating”	83

List of Tables

Table 1 Major Differences of {Phases} between <i>OPF</i> and <i>RUP</i>	74
Table 2 Major Differences of {Work Products} between <i>OPF</i> and <i>RUP</i>.....	75
Table 3 Major Differences of {Work Products} Sets between <i>OPF</i> and <i>RUP</i>.....	77
Table 4 Major Differences of Terminology between <i>OPF</i> and <i>RUP</i>	78

Permission Statement


The Context Maps Syntax and Patterns are copyrighted by Dr. Wojciech M. Jaworski
1988-2002.

All of the Context Maps Syntax and Patterns shown in this major report were used with
the permission from Dr. Wojciech M. Jaworski, the supervisor of the report.

Permission from: **Dr. Wojciech M. Jaworski**

Permission to: **Jing Bai**

Signature:

A handwritten signature in black ink, appearing to be 'Wojciech M. Jaworski', written over the 'Signature:' label.

Date:

Chapter 1

1. Introduction

1.1 Background

Knowledge mining is the technology that administers the representation, manipulation of information in a knowledge base. The most important issue for knowledge mining is how to represent and analyze the large amount of information. This report introduces the knowledge representation by applying *Context Maps* technology.

There are many representations for knowledge mining, such as *Rational Unified Process (RUP)*, which is a software engineering process delivered through a web-enabled, searchable knowledge base [1]. However, it is a challenge to develop a technology that needs to be powerful, easy to implement, and low cost. The *Context Maps* in this report is one way to satisfy the above requirements.

Context Maps is a notation and method for representing system architecture, structure, processes, and reusable templates. This technology was first introduced by Dr. Wojciech M. Jaworski. *Context Maps* was initially developed as a tool for recovering and refining knowledge from the legacy sources [2]. It can be applied in many domains, such as modeling, mining, and evaluation of contexts, enterprises, methods, processes, projects,

artifacts, databases, websites, information systems, and knowledge models with generic templates, domain experts, and proprietary notational technology.

Context Maps is the technology with a 3P-able notation, namely, Plug-able, Process-able, and Pattern search-able. *Context Maps* can easily integrate different knowledge into one consistent map. By using different syntax in the spreadsheet, it is feasible to describe and process conceptual knowledge. By using the logical query of context tools, it is convenient to retrieve and extract the specific information that users expect to search from the map.

OPEN Process methodology can be converted to a 3P-able format by applying *Context Maps* technology. The *OPEN* (Object-oriented Process, Environment, and Notation) *Process Framework (OPF)* is a practical, public-domain, industry-standard, general-purpose management, and engineering process framework that is primarily intended for the object-oriented, component-based development of software-intensive systems [3]. By considering the structure of *OPF*, it is possible for us to represent this methodology to *Context Maps*.

1.2 Outline of Report

Chapter I introduces the objective of study and the procedure of research.

Context Maps takes advantage of the 3P-able notation to represent system architecture, processes, and reusable components. Chapter 2 introduces 3P-able concept, inception, paradigm, syntax, notation, and tools of *Context Maps* technology.

OPEN Process Framework is a standard framework for creating a project-specific delivery process. Chapter 3 describes the reusable process components, principle, underlying theory, and process construction guidelines of *OPEN Process Framework*.

OPEN Process methodology can be represented by *Context Maps* technology based on concepts and relationships. Chapter 4 illustrates how to convert *OPF* to a 3P-able format, and the advantages of *Contexts Maps* technology.

Rational Unified Process (RUP) is another methodology of software engineering process. Chapter 5 compares the major differences between *OPF* and *RUP*, and analyzes the *OPF* strengths as well as *RUP* weaknesses addressed by *OPF*.

Chapter 6 provides the evaluative conclusions and recommendations for future works.

1.3 Objective of Study

The main purpose of this major report is to introduce a new method, named *Context Maps*, to represent *OPEN Process Framework* in a 3P-able notation. Based on this technology, *OPF* can be represented in a spreadsheet in terms of the concepts and

relationships. After converting *OPF* and *RUP* to *Context Maps*, it is easy to compare the similarities and differences between these two processes.

1.4 Procedure of Research

The research work was supervised by Professor Wojciech M. Jaworski. It was started from January 2002. The development of this report has progressed in the following steps:

- 1) Analyze the requirements for this major report.
- 2) Do research on *Context Maps* syntax, and study to represent the diagrams by using *Context Maps* notation.
- 3) Get familiar with *OPEN Process Framework*, especially in understanding the organization guidelines of this methodology for knowledge representation.
- 4) Convert *OPEN Process* methodology to a 3P-able format by applying *Context Maps* technology.
- 5) Compare *OPEN Process Framework* with *Rational Unified Process*, and illustrate the similarities and differences.
- 6) Make a conclusion for this research, and provide recommendations for future works.

Chapter 2

2. Context Maps

2.1 3P-able Notation

The 3P-able notation is a technology for representing enterprise architectures, static and dynamic structures, templates, and schema for system processes and artifacts. The library of this notation acts as a knowledge repository. The most important character of *Context Maps* is the 3P-able notation, which can be illustrated as the following formula:

3P-able ::= Plug-able + Process-able + Pattern search-able [4]

- **Plug-able:** Merge-able

Context Maps is a collection of different pieces of knowledge connected in a logical way. Many scattered elements and relationships among the concepts in software engineering process can be integrated to one *Context Map*, so that a simple and clear view is presented for users. Based on *Context Maps* technology, separate knowledge can be merged into one view.

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

By using different syntax in spreadsheet, it is feasible to describe and process conceptual knowledge. The letters “c”, “r”, “u”, and “d” stand for “create”, “read”, “update”, and “delete” in software engineering process.

- **Pattern search-able:** Search-able, Navigate-able

Context Maps uses spreadsheet to control, access, and navigate knowledge, and creates virtual maps for information system. By using the logical query of context tools (see section 2.6), it is convenient to retrieve and extract the specific information that users expect to search from the maps.

The 3P-able knowledge is represented by vertical knowledge tuples (see section 2.4).

- Knowledge tuple consists of "Set", "Set Member", and "Role" tuples.
- Knowledge tuple includes its own schema.
- Knowledge tuple contains identifier, type, and descriptor.

The order of knowledge tuples is meaningless.

- Knowledge tuple is a member of one or more knowledge views.
- Knowledge view is a member of one or more knowledge hubs.
- Knowledge hub is a member of one or more knowledge domains.

2.2 Context Maps Inception

Context Maps was first introduced by Dr. Wojciech M. Jaworski [5]. The technology was initially developed as a tool for recovering and refining knowledge from the legacy systems. This technology has a history of names. During the late 1970s and early 1980s, based on conceptual graphs introduced by J. F. Sowa, it was named as Array Based Language (ABL) [6]. In the late 1980s, it was renamed as ABL/W4 [7]. W4 is the

meaning of what, when, where, and which. In the early 1990s, by considering existing notations and methodologies, Dr. Jaworski named this technology as *jMaps* [8]. In the late 1990s until now, *jMaps* is represented as *Context Maps* [9].

2.3 Context Paradigm

Context Maps consists of unlimited number of context tuples that is a generic association of set members cast in roles [10]. In the extended spreadsheet, a column of roles and the related set members define context tuples. Context tuples can represent system behaviours, processes, tasks, procedures, and programs. From the graphical view, context tuples are represented by a compound edge and the connected compound nodes. A directed edge object consists of a tail object, middle object, and head object. Data tuples, data flows, rules, state transitions, process tuples, and multitude of other constructs are special cases of context tuples. The aggregated context tuples develop the associative model of data.

2.4 Context Maps Syntax and Process

In a technical sense, *Context maps* describes what a knowledge set is about, by formally declaring topics, and by linking the relevant parts of the knowledge set to the appropriate topics. The syntax of *Context Maps* is based on the relationship-oriented paradigm, and support multilingual modeling [10]. *Context Maps* represents the relationships between different knowledge nodes in a spreadsheet by columns in vertical level. The knowledge tuple is the fundamental structure defined by the concepts and instances related to roles.

The relevant mechanism is implemented by allocating roles to sets in schema and their instances to set members in the maps.

Compared to diagrams, *Context Maps* is very compact with offering a rich context within limited space of computer screen. *Context Maps* is created or edited within an organized electronic spreadsheet, which assures efficient manipulation of relationships (columns) and heavy reuse of components (rows). *Context Maps* represents the relationships between different knowledge nodes by vertical columns, and provides functionality of spreadsheets, arrays, graphs, relational tables, etc.

Here is an example to explain how to represent a series of workflows in Unified Process by applying *Context Maps* technology. The *RUP* is a software engineering process, which provides a disciplined approach to assigning tasks and responsibilities within a development organization [1]. The workflows group activities logically by nature. According to the diagram on the right side of Figure 1, each stage can be converted to its subsequent stage after achieving all the tasks involved in this stage.

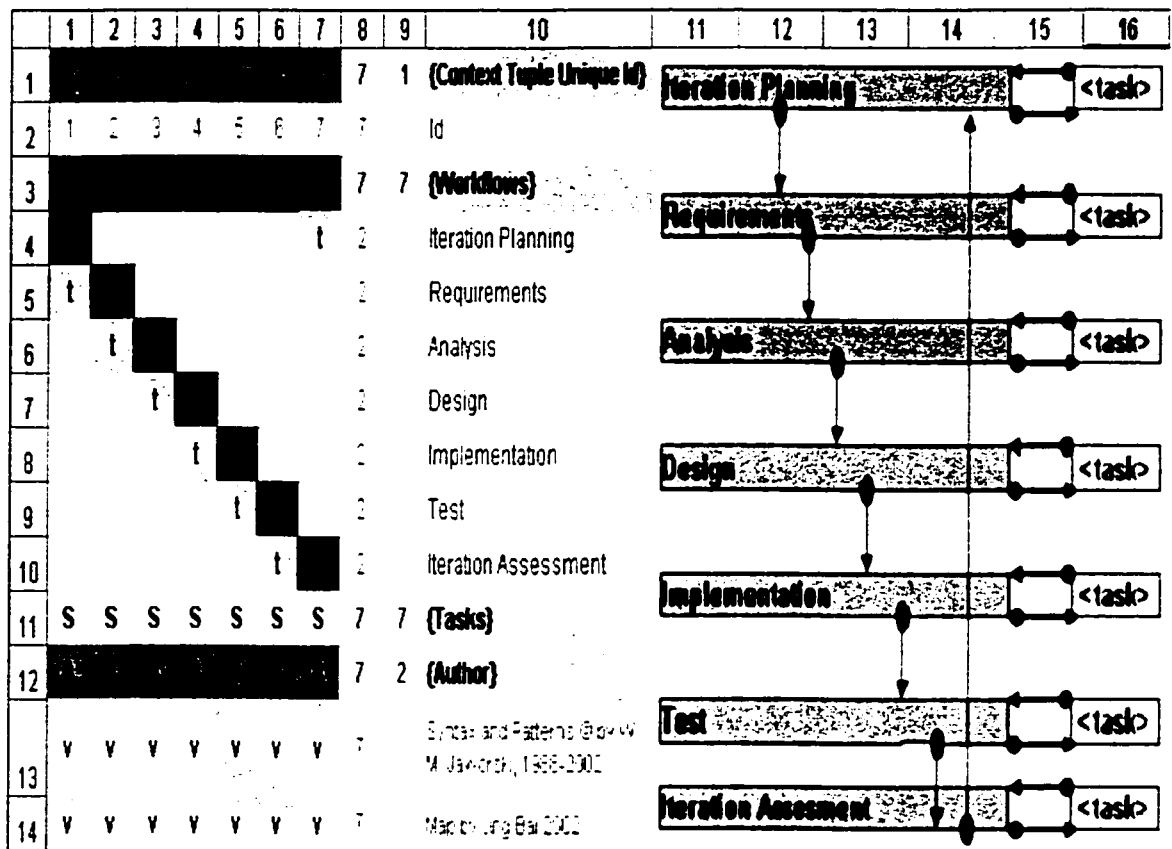


Figure 1 The *Context Maps* of Workflows Diagram in Unified Process

In Figure 1, nodes are represented as “Sets” or “Set members”, and arrows are represented as “Set Roles” or “Set Member Roles”. The terminology and syntax of *Context Maps* are introduced as follows:

- 1) **Sets**: the bold { } in column10, such as {**Workflows**} and {**Tasks**}.
- 2) **Set Members**: the elements below the bold { } in column10, such as “Iteration Planning”, “Requirements”, and “Analysis”.
- 3) **Context Tuples**: the contents from column1 to column7.
- 4) **Set Roles**: the upper case letters between column1 and column7:

- “I”: Identifier Tuple Unique ID
- “L”: Flow graph with cycles
- “S”: Sequence
- “A”: Aggregation of columns

5) **Set Member Roles**: the lower case letters or digitals between column1 and column7:

- “l”: loop node
- “t”: to node
- “v”: marker

6) **Number of Set Members**: the column9 counts the amount of “Set Members”.

7) **Number of Set Member Roles**: the column8 counts the amount of “Set Member Roles”.

In general, abstract concepts appear on the right side of the map in bold and between curly brackets. They can be thought as a heading of a table column or row. The instances are the actual contents listed in the table. Each column is to be read vertically using the syntax that was described above. While reading down a column, each variable should be read towards the right side of the map to see which concept or instance the variable is referring to. Beside each instance and under the total number of instances, counts the number of times the instance is referred to. If large *Context Maps* needs to be developed, we can hide irrelevant columns and rows, edit visible cells, and insert new columns and rows.

2.5 Context Maps Notation

Context Maps notation allows efficient recovery and modeling of generic schema for processes, objects, and views of information systems. *Context Maps* notation can be used in many fields such as:

- Information system architecture
- Evolving information systems
- Automation of system design
- Recovery and reuse of system patterns
- Context management system
- Software evaluation and renewal
- Modeling of websites and knowledge hubs

Figure 2 illustrates the *Context Maps* notation:

														14	1	{Context Tuple Unique Id}													
1	2	3	4	5	6	7	8	9	10	11	12	13	14	14	14	1	Id												
														14	1	{Context View}													
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14	14		Syntax												
														14	2	{Association}													
														2		Is a													
														12		Is valid member role for													
														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 = ft													
														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)elele													
														2		x - component of Cartesian Product													
														14	1	{Author}													
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14	14		Syntax and Patterns © by W M Jaworski, 1988-2002												

Figure 2 The Context Maps Notation

2.6 Context Maps Tool

The environment of creating and developing *Context Maps* is MS Excel. Excel is a spreadsheet that is used to analyze data, complete calculations, draw charts, develop professional reports, and make decisions.

The tool of *Context Maps*, named "CONTEXT+", is developed by Dr. Jaworski to retrieve the useful information and generate the corresponding maps. This tool includes the following main functions:

- Show Schema
- Query
- Compute Cardinality
- Apply Color

The prominent context tool "Query" is the most useful function in "CONTEXT+" tool. It gives users a convenient and flexible way to manipulate and control the entries of *Context Maps*. In order to use this function, the combination of set members should be selected in *Context Maps*. After choosing the "OR", "XOR", "AND", or "NOT" option, and clicking the "OK" button on the interface, the specific set members and relevant relationships are picked out to generate the result *Context Maps*. Figure 3 shows the interface of the "Query" function in "CONTEXT+" tool:

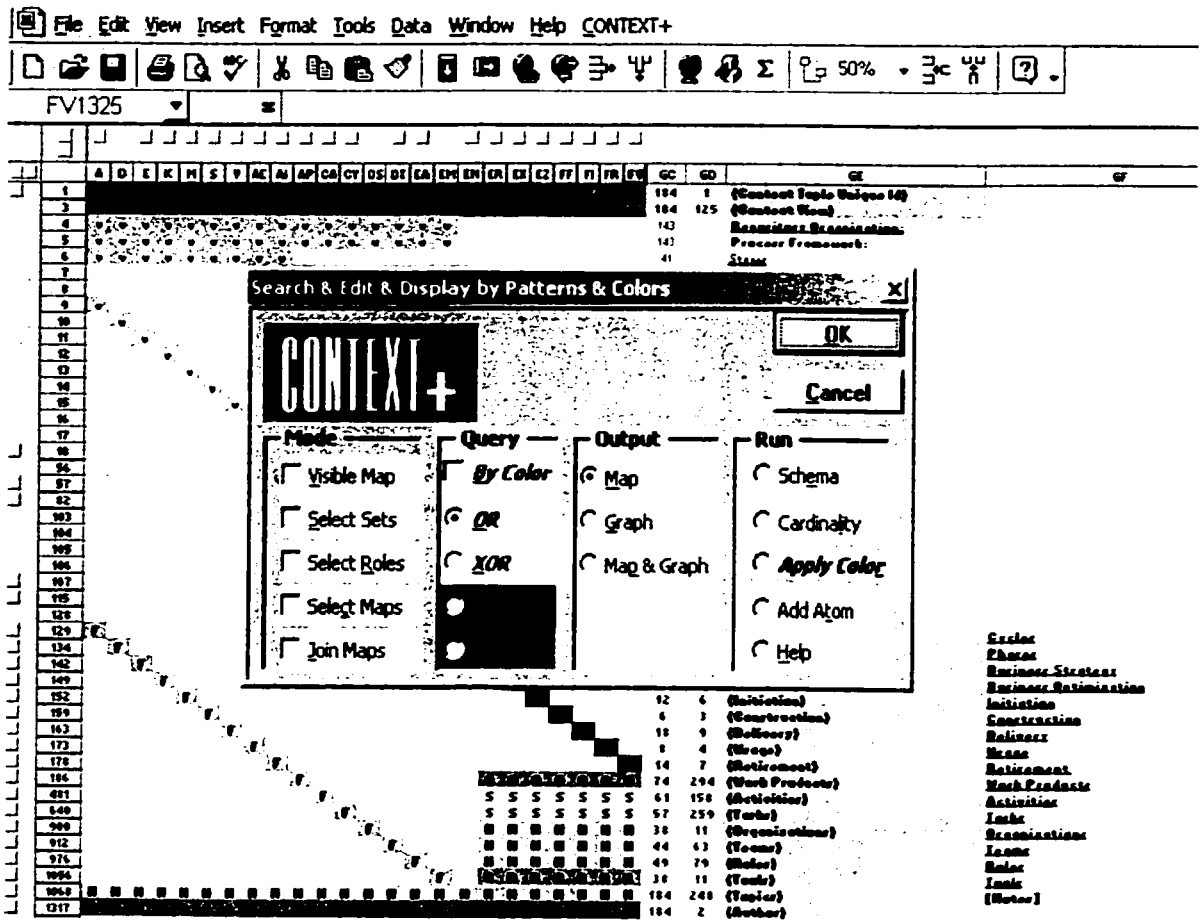


Figure 3 The Interface Diagram of "Query" Function

The syntax and patterns of *Context Maps* are protected by copyright and trade secret law, and may not be copied or produced for commercial purpose, without written permission from Dr. Wojciech M. Jaworski.

Chapter 3

3. OPEN Process Framework

3.1 OPEN Process Framework Introduction

The *OPEN Process Framework (OPF)* is a practical, public-domain, industry-standard, general-purpose management, and engineering process framework that is primarily intended for the object-oriented, component-based development of software-intensive systems [3].

The *OPF* consists of a:

- **Repository** of predefined reusable process components.
- **Metamodel** that organizes and provides a theoretical foundation for these components.
- **Guidelines** for using these process components to construct endeavor-specific processes.

The *OPF* provides managers, developers, strategists, user experience personnel, process engineers, methodologists, consultants, trainers, and academics with the best current industry practices for processes to perform:

- **Business (Re)Engineering** including business requirements engineering, business architecting, digital branding, management, etc.

- **Applications Development** including application requirements engineering, architecting, design, implementation, integration, testing, etc.
- **Applications Usage** including operations, maintenance, content management, eventual retirement, etc.
- **Reusable Component Development** including requirements engineering, architecting, etc.

3.2 OPEN Process Framework Reusable Process Components

3.2.1 Definition

An *OPF* process component is a reusable component that can be integrated with other such components to construct an organizational-specific or endeavor-specific process.

3.2.2 Repository Description

As a framework, the *OPF* comes with a repository containing:

- A cohesive class library of reusable interrelated process components (e.g. Milestone).
- Selected instances of these classes (e.g. Requirements Frozen Milestone, Architecture Frozen Milestone, Construction Complete Milestone).

3.2.3 Component Metamodel

As a relatively complete process framework, the *OPF* comes with the following predefined classes of process components as documented on the right side of Figure 4.

This component metamodel is represented by applying *Context Maps* technology on the left side of Figure 4:

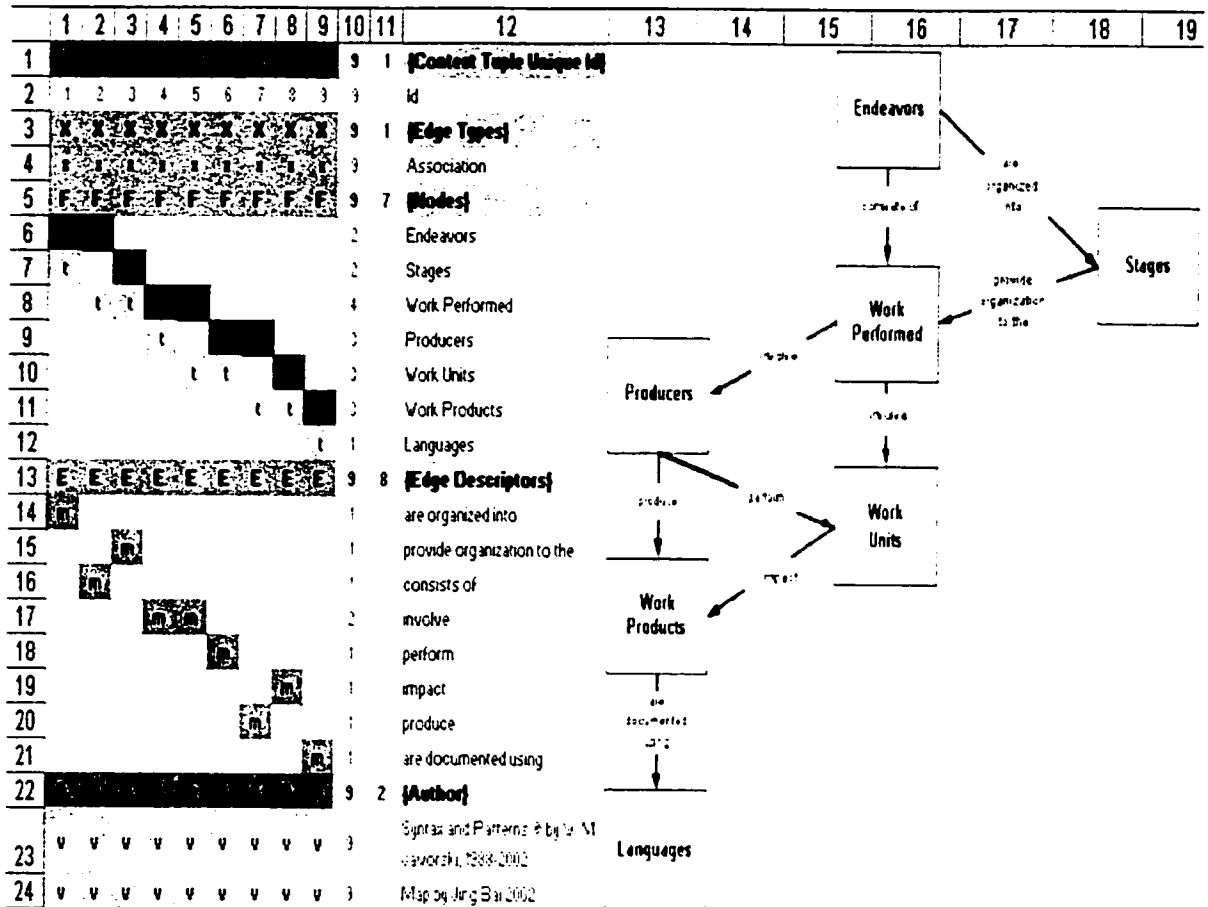


Figure 4 The *Context Maps* of *OPF* Component Metamodel

3.2.4 Component Hierarchy

As illustrated in the preceding component metamodel, the *OPF* repository of process components contains the following predefined classes of reusable process components for producing state-of-the-art development processes:

- **Endeavors:** are process components that model large-scale ventures undertaken by collaborating producers during multiple stages to develop and maintain one or more related applications.
 e.g. Enterprises, Programs, Projects.
- **Stages:** are process components that model formally identified time periods or points in time that provide organization to the work units of the delivery process.
 e.g. Cycles, Phases, Builds, Milestones.
- **Work Performances:** are process components that model work units as performed by producers.
 e.g. Work Flows, Task Performances.
- **Producers:** are process components that model anything that produces, either directly or indirectly, versions of one or more work products.
 e.g. Organizations, Teams, Roles, Tools, Persons.
- **Work Units:** are process components that model functionally cohesive operations that are performed by producers during the delivery process.
 e.g. Activities, Tasks, Techniques.
- **Work Products:** are process components that model anything of value (e.g. documents, diagrams, applications, classes) that are produced by the collaboration of one or more producers during the performance of one or more tasks.
 e.g. Management, Business Engineering, Application Development, Models, Diagrams, Versions.

- **Languages:** are process components that model the languages used to document work products.
e.g. Natural Languages, Modeling Languages, Implementation Languages.

3.2.5 Guidelines

- Different endeavors will require different processes. In turn, these will require different process components.
- Select only those process components that are appropriate for the associated endeavor.
- The most important kinds of process components are work products, producers, work units, stages, and endeavors. Work performances and languages are less critical and included primarily for theoretical completeness reasons.
- Process components should be relevant, useful, usable, and cost-effective.

3.3 OPEN Process Framework Principles

The *OPEN Process Framework* is based on the following principles:

3.3.1 Flexibility

The *OPEN Process Framework* was designed for maximum flexibility.

- As a repository-based process framework, it contains a large class library of process components from which to choose only those that are appropriate and cost effective.
- The chosen process components are intended to be tailor-able.

3.3.2 Industry Best Practices

The *OPEN Process Framework* is based on the industry's best practices:

- Many internationally-recognized and authorized sources.
- Requirements, architecting.
- Web-development guidelines.

3.3.3 Completeness

A complete process framework has numerous advantages:

- A complete process framework enables many more endeavor-specific processes to be constructed from its components without the need for extending it with new components.
- A complete process framework is more likely to be usable for any endeavor.
- A process framework, the components of which are completely documented, is less likely to be missing significant documentation.

The *OPEN Process Framework* provides a repository of process components that is very complete:

1) The *OPF* supports the major management, business engineering, application development, and operation activities:

a) Business Engineering:

- Business Strategy
- Technology Strategy
- Business Architecting
- Digital Branding

b) Application development:

- Requirements Engineering
- Architecting
- Design
- Implementation
- Integration
- Testing
- Delivery

c) Operations:

- Content Management
- Maintenance
- Operations

2) The *OPF* includes all major classes of process components:

- Endeavors
- Stages

- Producers
- Work Units
- Work Products
- Languages

3) The *OPF* provides a balanced inheritance hierarchy containing numerous subclasses for each major class of process component:

- Endeavors: 15 subclasses of enterprises, programs, and projects.
- Stages: 40 subclasses of cycles, phases, builds, and milestones.
- Producers: over 120 subclasses of organizations, teams, roles, persons, and tools.
- Work Units: over 300 subclasses of activities, tasks, and techniques.
- Work Products: over 350 subclasses of management, business engineering, application development, models, diagrams, and versions.
- Languages: 10 subclasses.

4) The *OPF* covers all major organizations, teams, and roles.

5) The *OPF* covers all major work products:

- Documents
- Application components (data, hardware, personnel, and software)
- Business components (applications facilities, business units)
- Metrics

- Models

6) Descriptions of the *OPF* process components are very complete.

3.3.4 Usability

The *OPEN Process Framework* is highly usable:

- The *OPF*-compliant endeavor-specific processes are easy to construct using the contents of the *OPF* repository of reusable process components.
- The *OPF* process components are easy to tailor. During an endeavor, it is easier to tailor out what is not needed from a complete work product.

3.3.5 Reuse

The *OPEN Process Framework* greatly supports reuse.

3.4 OPEN Process Framework Underlying Theory

The *OPEN Process Framework (OPF)* is based on the following layered set of process engineering models:

1) Process Instance Layer (Layer 0):

Process Instance Layer contains the actual processes used on individual projects and programs of related projects.

2) Process Model Layer (Layer 1):

Process Model Layer contains the frameworks that are instantiated to produce the individual processes in layer 0 including the:

- Standard OPEN Default process frameworks for creating light, middleweight, and heavy processes.
- Company-specific process frameworks (variants of the *OPEN process frameworks*).
- Extensible repository containing all of the reusable classes of process components used in these frameworks.

3) Process Metamodel Layer (Layer 2):

Process Metamodel Layer defines the fundamental concepts and patterns that provide the foundation for the process frameworks in layer 1.

As illustrated on the right side of Figure 5, the *OPEN Process Framework (OPF)* consists of the:

- *OPEN* Metamodel
- *OPF* Repository of reusable Process Component Classes
- *OPF* Usage Guidelines

There are three types of arrows in this layered model:

- “→”: Inheritance
- “⇒”: Association
- “⊕→”: Aggregation

This layered model is represented by applying *Context Maps* technology on the left side of Figure 5:

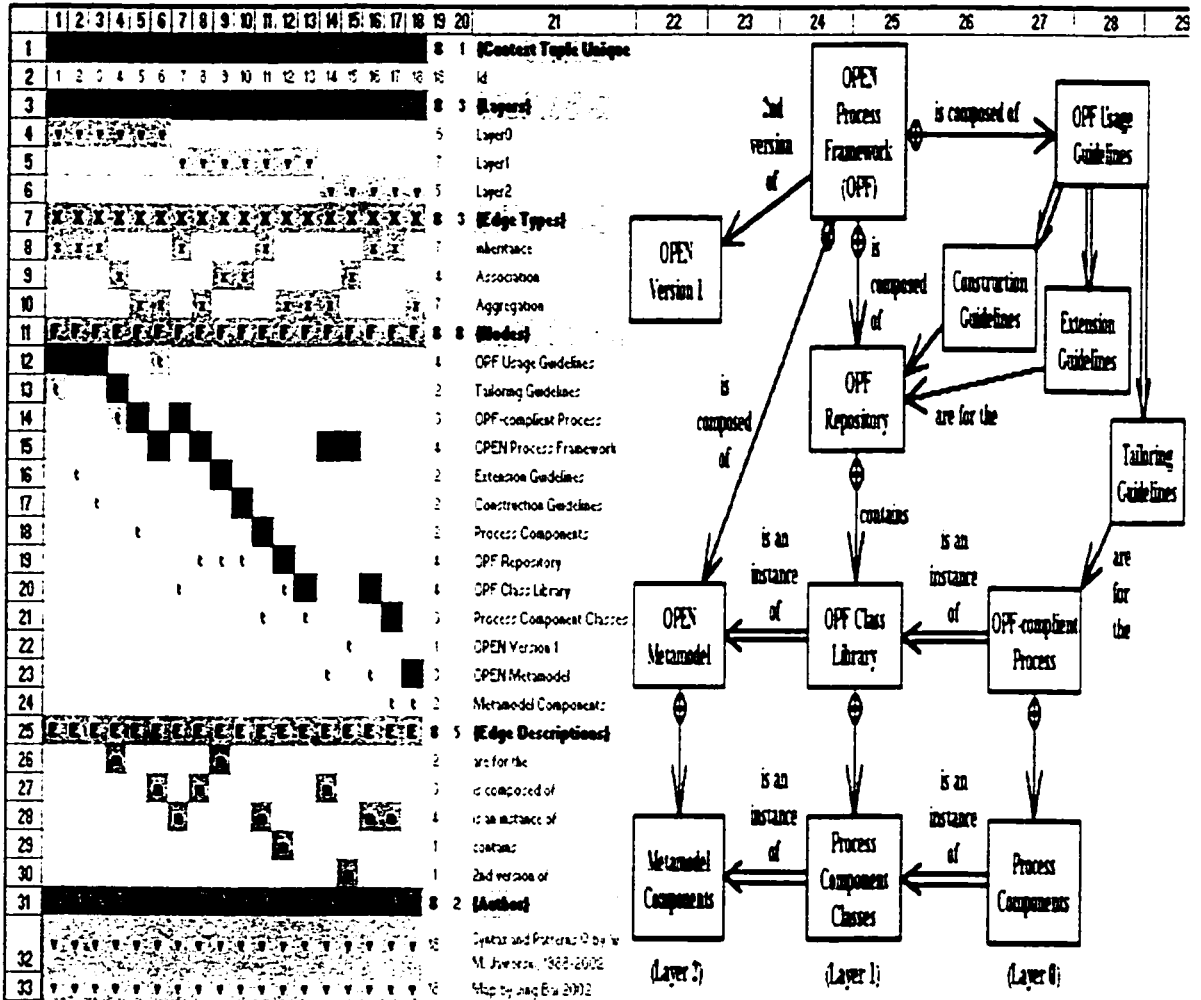


Figure 5 The *Context Maps* of OPF Layered Model

The *Context Maps* of the OPF component metamodel (Figure 4) and layered model (Figure 5) can be merged into one *Context Map*. Figure 6 presents the character of “Plug-able” in 3P-able notation:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	27	1	(Context Tuple Unique Id)		
																												27	27	Id	
v	v	v	v	v	v	v	v																					27	2	(Context View)	
										v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	9		OPEN Process Components	
																												18		OPEN Process Layered Model	
										v	v	v	v	v	v													27	3	(Layers)	
																v	v	v	v	v	v							6		Layer0	
																												7		Layer1	
																							v	v	v	v	v	5		Layer2	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	27	3	(Edge Types)	
																												7		Inheritance	
																												13		Association	
																												7		Aggregation	
f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	27	20	(Nodes)	
t																												2		Endeavors	
	t																											2		Stages	
		t																										4		Work Performed	
			t																									3		Producers	
				t																								3		Work Units	
					t																							3		Work Products	
						t																						1		Languages	
							t																					4		OPF Usage Guidelines	
								t																				2		Tailoring Guidelines	
									t																			3		OPF-compliant Process	
										t																		4		OPEN Process Framework	
											t																	2		Extension Guidelines	
												t																2		Construction Guidelines	
													t															2		Process Components	
														t														4		OPF Repository	
															t													4		OPF Class Library	
																t												3		Process Component Classes	
																	t											1		OPEN Version 1	
																		t										3		OPEN Metamodel	
																			t									2		Metamodel Components	
e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	27	13	(Edge Descriptions)	
																													1		are organized into
																													1		provide organization to the
																													1		consists of
																													2		involve
																													1		perform
																													1		impact
																													1		produce
																													1		are documented using
																													2		are for the
																													3		is composed of
																													4		is an instance of
																													1		contains
																													1		2nd version of
																													27	2	(Author)
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	v	v	v	v	27		Syntax and Patterns © by W. M. Jaworski, 1988-2002
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	v	v	v	v	27		Map by Jing Bai 2002

Figure 6 The Context Maps of Merging Figure 4 into Figure 5

3.5 Process Construction Guidelines

3.5.1 Process Framework

The *OPEN Process Framework (OPF)* recognizes that no two endeavors are exactly alike. Endeavors vary greatly in objectives, size, scope, complexity, criticality, and application domain. They also vary in terms of available schedule, resources, technology, and organizational culture. Therefore, no single process or methodology, no matter how tailor-able, will be optimal for all applications. Thus, the *OPEN* Consortium of methodologists, developers, tool vendors, and academics has produced a process framework rather than an individual process. In a sense, the *OPF* is not a development process, but rather an industry-standard process for producing development processes that meet the differing needs of specific endeavors.

3.5.2 Process Framework Repository

As a framework, the *OPF* contains a class library of reusable process components that is much too large and complete to be used as is on any project or even a program of related projects. Instead, only appropriate cost-effective process components are selected and tailored to produce a process that is appropriate for the specific needs of the endeavor. This framework is stored in a repository of reusable process component classes and instances.

3.5.3 Metamodel, Framework, and Process

As illustrated in Figure 7, the *OPEN* Consortium of methodologists, academics, CASE tool vendors, and industry experts constructs, extends, and maintains the *OPEN process* metamodel, which defines the process component classes and instances stored in the *OPEN Process Framework (OPF)* repository. The process team on an endeavor selects items in the repository and uses them to construct the endeavor-specific process. The process team can also extend the repository by adding new classes and tailor the endeavor-specific process by modifying its process components.

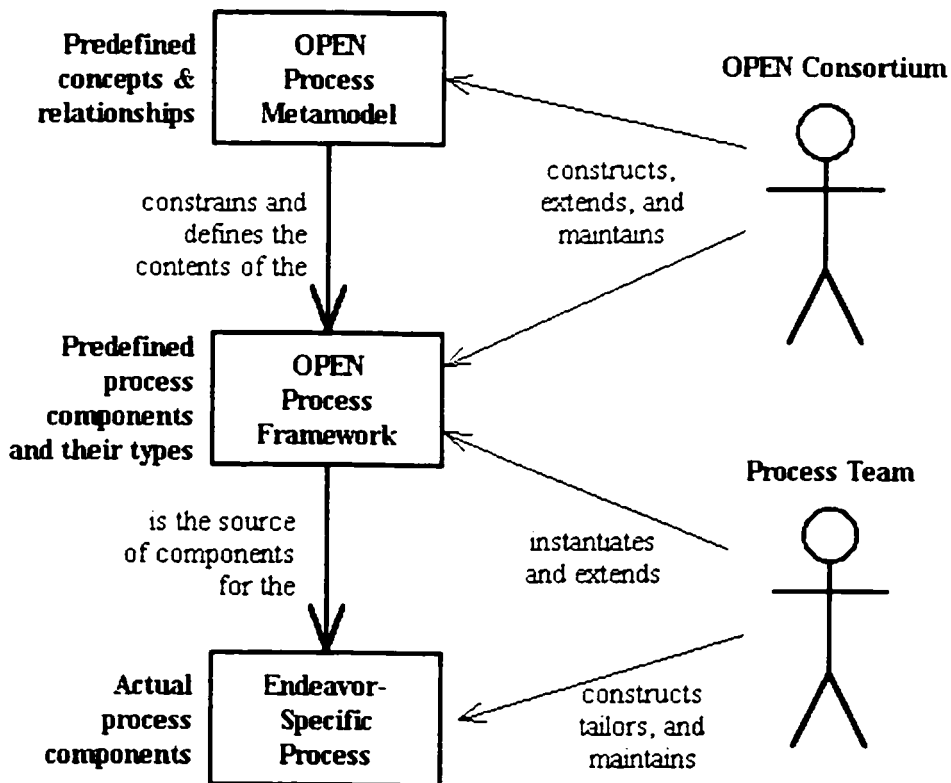


Figure 7 The Metamodel, Framework, and Process of OPF

3.5.4 Construction Process

To produce an endeavor-specific process, the following teams typically perform the following tasks iteratively and incrementally in roughly the following order:

1) Resource Management:

The endeavor's management team (e.g. program management team or project management) staffs the endeavor's process team.

2) Process Needs Assessment:

The process team assesses the endeavor's specific needs for process.

3) Process Construction:

The process team constructs an appropriate endeavor-specific process by selecting reusable process components from the process framework's repository.

4) Process Tailoring:

The process team tailors the constructed process (and its process components) to better meet the specific needs of the endeavor. The process team also continuously iterates the process based upon the results of analyzing inputs from the endeavor's staff.

5) Process Documentation:

The process team documents the endeavor's process in the associated process description document and conventions.

6) Training Delivery:

The training team provides appropriate training to both the endeavor staff and the customer organization in all aspects of the endeavor process.

7) Process Mandating:

The endeavor management team formally mandates the use of the endeavor process.

8) Process Consulting:

The process team supports the endeavor staff by answering process questions and mentoring staff members.

9) Quality Assurance:

The endeavor's staff and quality team continuously evaluate the effectiveness of the process in terms of work product quality and staff efficiency. Upon completion of the endeavor, they also capture process lessons learned.

10) Process Framework Iteration:

The process team continuously iterates the organizational process framework based on actual usage, technological advances, and academic research. This may include adding, deleting, or modifying process components.

Chapter 4

4. Converting Open Process Framework to Context Maps

The structure and contents of the *OPEN Process Framework* are clear and well organized; however, the relationships among specific elements are not obvious. The *OPF* is a container based on websites, all of the components are classified and hyper-linked together. In order to find out the relationship between two elements, we need to navigate through several hyperlinks. In addition, the website documenting the repository of process components are organized hierarchically, in accordance with the structure of the *OPEN* metamodel. Therefore, even when we find a certain relationship, the *OPF* still cannot summarize some problems, such as in which milestones a certain work product is produced or in which phases a milestone is involved. All these issues can be easily addressed by applying *Context Maps* technology.

4.1 Procedure of Converting OPF to Context Maps

After analyzing the organization of *OPEN Process Framework*, the relevant knowledge can be extracted from the website in logical level, and converted to *Context Maps* by using the following steps:

- 1) Identification of Sets
- 2) Enumeration of Set Members

- 3) Definition of Schema
- 4) Creation of Knowledge Tuples
- 5) Aggregation of Knowledge Tuples into Knowledge Views

4.1.1 Identification of Sets

Identify the “Sets” of the *OPEN Process Framework*, which are used to construct the schema of *Context Maps*. The top-level structure of the *OPF* consists of the following “Sets”:

- 1) {Context Tuple Unique Id}
- 2) {Context View}
- 3) {Cycles}
- 4) {Phase}
- 5) {Business Strategy}
- 6) {Business Optimization}
- 7) {Initiation}
- 8) {Construction}
- 9) {Delivery}
- 10) {Usage}
- 11) {Retirement}
- 12) {Work Products}
- 13) {Activities}
- 14) {Tasks}
- 15) {Organizations}

16) {Teams}

17) {Roles}

18) {Tools}

19) {Topics}

4.1.2 Enumeration of Set Members

Enumerate the “Set Members” of *OPEN Process Framework*, and classify relevant “Set Members” into the corresponding “Sets”. The semantic of the “Set Members” in the *OPF* schema is described as follows:

- 1) **{Context Tuple Unique Id}**: They are continuous numbers for each column to indicate data tuples.

- 2) **{Context View}**: It is the top-level structure of the *OPEN Process Framework*, which is consisted of two main sections:
 - Process Framework
 - Process

The difference between Process Framework and Process is as follows. Think of a process framework as a tool to build a process, or as a warehouse containing the parts needed to build a process, or better yet, as a factory for building processes. It is something like the difference between a car factory and the cars it produces. The factory contains the tools and the parts needed to build the cars, and the factory workers that build the cars are like the process engineers that take the process components and build processes. However, unlike a car factory that

produces large numbers of only a few car models, a process framework is used to build a different, endeavor-specific process each time.

A **stage** is an *OPF Process* component that models a formally identified period or point in time that provides structure to the work units of an endeavor-specific process. The *OPF* class library of reusable process components contains the following predefined stages:

- Cycles
- Phases
- Milestones

3) **{Cycles}**: A cycle is the top-level stage consisting of one or more related phases. The *OPF* repository of process components includes the following predefined cycles:

- Business Engineering Cycle
- Application Life Cycle
- Application Development Cycle

4) **{Phases}**: A phase is the kind of stage that forms a major part of a cycle. A phase consists of one or more builds, and it has one or more milestones. Each set member of {Phases} is decomposed into several milestones. The *OPF* repository of reusable process components contains the following phases:

- Business Strategy

- Business Optimization
- Initiation
- Construction
- Delivery
- Usage
- Retirement

A **milestone** is the kind of stage modeling a major scheduled point in time during the delivery process by which a cohesive set of significant objectives (e.g. set of tasks completed, set of work products delivered) is to be achieved.

5) **{Business Strategy}**: The business strategy phase is the first phase of the business engineering cycle, during which the development organization develops new strategies and architectures for all or part of the customer organization's business enterprise. The typical milestones of the business strategy phase of a business engineering project include:

- Project Mobilized
- Analyses Complete
- Business Vision Complete
- Strategies Complete
- Business Architecture Complete
- Business Strategy Phase Complete

6) **{Business Optimization}**: The business optimization phase is the second phase of the business engineering cycle, during which the development organization helps the customer organization optimize all or part of its business enterprise by implementing, communicating, and continually optimizing its new business strategies and architectures. The typical milestones of the business optimization phase include:

- Brand Communicated
- Reengineered Business Communicated
- Business Optimization Phase Complete

7) **{Initiation}**: The initiation phase is the first phase of the application development and life cycles, during which enough of the application's requirements and architecture are produced so that agreement can be reached between the development organization and the customer organization regarding the goals and scope of the following phases. The typical milestones of the initiation phase of an application development project include:

- Project Mobilized
- Significant Requirements Specified
- Initial Architectures Documented
- End-To-End Prototype Validated
- Following Phases Planned
- Initiation Phase Complete

8) **{Construction}**: The construction phase is the second phase of the application development and application life cycles, during which the development organization completes construction of an application and its associated deliverable work products. The typical milestones of the construction phase include:

- Requirements and Architecture Frozen
- Application Implemented
- Construction Phase Complete

9) **{Delivery}**: The delivery phase is the third phase of the development and life cycles, during which the development organization delivers the primary work products to the customer organization. The typical milestones of the delivery phase of an application development project include:

- Data Center Established
- Production Environment Established
- Application Deployed
- Delivery Phase Documentation Delivered
- Customer User Training Delivered
- Alpha Tests Passed
- Beta Tests Passed
- Acceptance Tests Passed
- Delivery Phase Complete

10) **{Usage}**: The usage phase is the fourth phase of the application life cycle, during which the application is used by its user organizations.

11) **{Retirement}**: The retirement phase is the final application life-cycle phase, during which the application is retired from service by the customer organization and use by the user organizations.

12) **{Work Products}**: A work product is an *OPF Process* component that models anything of value (e.g. document, diagram, application, software component, hardware component) that is produced by one or more collaborating producer during the performance of one or more task.

A **work unit** is an *OPF Process* component that models a functionally cohesive operation that may be performed by one or more producers as part of an endeavor-specific process. The *OPF* class library of reusable process components contains the following predefined instances of work units:

- Activity
- Task

13) **{Activities}**: An activity is the highest-level work unit consisting of a cohesive collection of one or more tasks that are performed by one or more collaborating

producers when producing a set of one or more related work products or providing one or more related services.

14) **{Tasks}**: A task is a mid-level work unit that models a functionally cohesive operation performed by one or more producers.

A **producer** is an *OPF Process* component that models something that performs tasks that:

- Produce (e.g. create, evaluate, iterate, or maintain), either directly or indirectly, versions of related work products.
- Provide, either directly or indirectly, one or more services.

The *OPF* class library of reusable process components contains the following predefined classes and instances of producers, organized firstly by inheritance and secondly composition:

- Organizations
- Teams
- Roles
- Tools

15) **{Organizations}**: An organization is a relatively large-scale indirect producer with the following characteristics:

- Endeavor: An organization is a major component of an enterprise.

- **Structure:** An organization consists of a cohesive collection of one or more related teams.
- **Management:** An organization is managed as a unit.

16) **{Teams}**: A team is a relatively small-scale indirect producer with the following characteristics:

- **Endeavor:** A team is a major component of a program or project.
- **Parent Producer:** A team is a major component of an organization.
- **Structure:** A team consists of a cohesive collection of one or more related roles and subordinate teams that collaborate to perform a cohesive set of tasks (i.e. typically an activity, although the tasks may comprise a major subset of an activity or a set of related activities).
- **Management:** A team is managed as a unit.

17) **{Roles}**: A role is a low-level direct producer that represents a part (i.e. a cohesive collection of responsibilities) that is played by one or more persons during an endeavor.

18) **{Tools}**: A tool is a direct producer that is a software application that is used by one or more persons playing one or more roles to create, modify, evaluate, or manage versions of work products.

19) **{Topics}**: Topics are the detailed definitions and descriptions for the corresponding set members.

4.1.3 Definition of Schema

Indicate the symbols of “Set Roles” and “Set Member Roles”, in terms of the *Context Maps* notation introduced in Section 2.5. In general, the schema provides the information of structure and size in *Context Maps*. The schema of the *OPEN Process Framework* can be obtained by hiding set members and irrelevant columns.

4.1.4 Creation of Knowledge Tuples

In *Context Maps*, each column represents the specific relationship of “Set Members” in logic level. The knowledge tuple is created by connecting relevant “Set Members” in terms of the relationships between *OPEN Process* components.

4.1.5 Aggregation of Knowledge Tuples into Knowledge Views

The knowledge tuples can be aggregated together to generate knowledge views according to the concepts and semantics of *OPEN Process* methodology. According to the above analysis, the schema view of *OPEN Process Framework* is represented by *Context Maps* as Figure 8:

	1	4	5	11	13	19	22	31	35	42	79	103	123	124	131	143	144	148	154	156	162	165	174	178	184	185	186		
1																									184	1	(Context Tuple Unique Id)		
2	1	4	5	11	13	19	22	31	35	42	79	103	123	124	131	143	144	148	154	156	162	165	174	174	184		1	Id	
3																									184	125	(Context View)		
4	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v										143		Repository Organization:	
5	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v										143		Process Framework:	
6	v	v	v	v	v	v	v	v	v																	41		Stages	
7										v	v	v	v	v	v											102		Objects	
8																v	v	v	v	v	v	v	v	v	v	41		Process:	
9	v															v										7		Cycles	
10		v															v	v	v	v	v	v	v	v	v	38		Phases	
11			v														v									12		Business Strategy	
12				v														v								4		Business Optimization	
13					v															v						12		Initiation	
14						v															v					6		Construction	
15							v															v				16		Delivery	
16								v															v			8		Usage	
17									v															v		14		Retirement	
18										v																37		Work Products	
56											v	v														44		Work Units	
57											v															24		Activities	
82												v														20		Tasks	
103												v	v	v	v											21		Producers	
104												v	v													8		Indirect Producers	
105														v	v											13		Direct Producers	
106												v														1		Organizations	
107													v													7		Teams	
115															v											12		Roles	
128																v										1		Tools	
129	F																									44	4	(Cycles)	
134		F															S									42	7	(Phases)	
142			F																							12	6	(Business Strategy)	
149				F																						4	2	(Business Optimization)	
152					F																					12	6	(Initiation)	
159						F																				6	3	(Construction)	
163							F																			18	9	(Delivery)	
173								F																		8	4	(Usage)	
178									F																	14	7	(Retirement)	
186										F											O	O	O	O	O	O	74	294	(Work Products)
481											F										S	S	S	S	S	S	61	158	(Activities)
640												F									S	S	S	S	S	S	57	259	(Tasks)
900													F								N	N	N	N	N	N	38	11	(Organizations)
912														F							N	N	N	N	N	N	44	63	(Teams)
976															F						N	N	N	N	N	N	49	79	(Roles)
1056																F					R	R	R	R	R	R	38	11	(Tools)
1068	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	184	248	(Topics)
1317																										184	2	(Author)	
1318	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	v	184		Syntax and Patterns © by W M Jaworski, 1988-2002
1319	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	v	184		Map by Jing Bai 2002

Figure 8 The Context Maps Schema of OPEN Process Framework

The notations of *Context Maps* used in the *OPEN Process Framework* are described as follows:

- **The symbols of Set Roles:**

I - (I)dentifier Tuple Unique ID

A - (A)ggregation of columns

F - (F)low graph nodes

L - (L)Flow graph with cycles

S - (S)equence

O - (O)bjects

N - (N)ode properties

R - (R)esources

- **The symbols of Set Member Roles:**

l ... * - numerical value

v - marker

f - (f)rom node component

t - (t)o node component

l - (l)oop node component

c - (c)reate

r - (r)ead

u - (u)pdate

d - (d)etele

The semantics of *OPEN Process* schema view are explained as follows:

- Each set member of {Cycles} (Row129) is the top-level stage consisting of one or more related {Phases} (Row134).
- Each set member of {Phases} (Row134) is the workflow involving a sequence of {Milestones} (Row142-178).
- Each set member of {Milestones} (Row142-178) is the kind of stage modeling a major scheduled point in time during the delivery process by which a cohesive set of significant objectives is to be achieved.
- Each stage of {Milestones} (Row142-178) produces one or more {Work Products} (Row186).
- Each stage of {Milestones} (Row142-178) contains a series of {Activities} (Row481), and {Tasks} (Row640).
- Each stage of {Milestones} (Row142-178) involves several {Organizations} (Row900), {Teams} (Row912), and {Roles} (Row976).
- Each stage of {Milestones} (Row142-178) makes use of different {Tools} (Row1056) to assist the development.

The following sections provide the detailed descriptions for each view of *Context Maps* in *OPEN Process Framework*:

4.2 Stages: Cycles and Phases View

The *OPF* repository of process components includes the following cycles and phases:

1. Business Engineering Cycle:

Business Engineering Cycle is the cycle that includes all phases, during which a business is (re)engineered:

- 1) **Business Strategy:** is the first business engineering cycle phase, during which the new strategies and architecture of the customer organization's business enterprise are developed.
- 2) **Business Optimization:** is the second business engineering cycle phase, during which the new strategies and architecture of the customer organization's business enterprise are placed into effect and continually optimized.

2. **Application Life Cycle:**

Application Life Cycle is the complete application cycle that includes all phases, during which a single application is produced, used, and retired:

- 1) **Initiation:** is the first application life cycle phase, during which the application's initial vision, major requirements, and partial architecture are captured so that the application's scope can be estimated.
- 2) **Construction:** is the second application life cycle phase, during which the complete version of the application is developed.
- 3) **Delivery:** is the third application life cycle phase, during which the new version of the application is delivered to the customer organization and placed into use by the user organizations.

- 4) **Usage:** is the fourth application life cycle phase. during which minor updates to the new version of the application are made while it is in use by the user communities.
- 5) **Retirement:** is the fifth application life cycle phase. during which the application is retired from use and its components are archived for future use.

3. **Application Development Cycle:**

Application Development Cycle is the cycle that includes all phases. during which a single application is developed:

- 1) **Initiation:** is the first application development cycle phase. during which the application's initial vision, major requirements, and partial architecture are captured so that the application's scope can be estimated.
- 2) **Construction:** is the second application development cycle phase. during which the complete version of the application is developed.
- 3) **Delivery:** is the third application development cycle phase. during which the new version of the application is delivered to the customer organization and placed into use by the user organizations.

Figure 9 illustrates the preceding cycles and their component phases, and Figure 10 demonstrates the state transition diagram of cycles and phases:

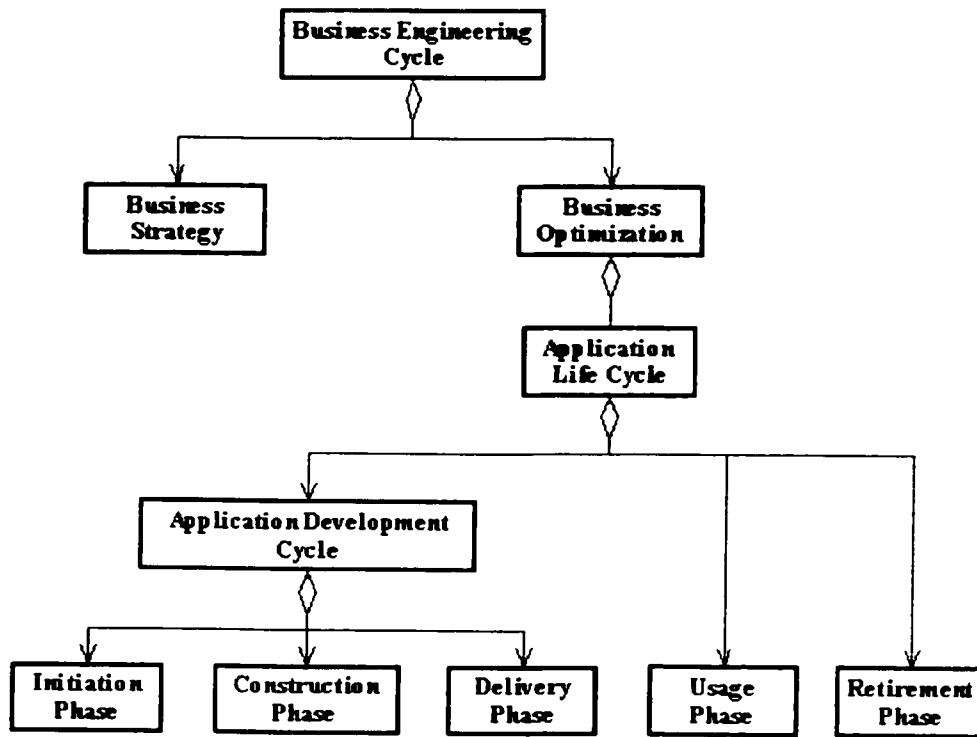


Figure 9 The Relationship Diagram of Cycles and Phases

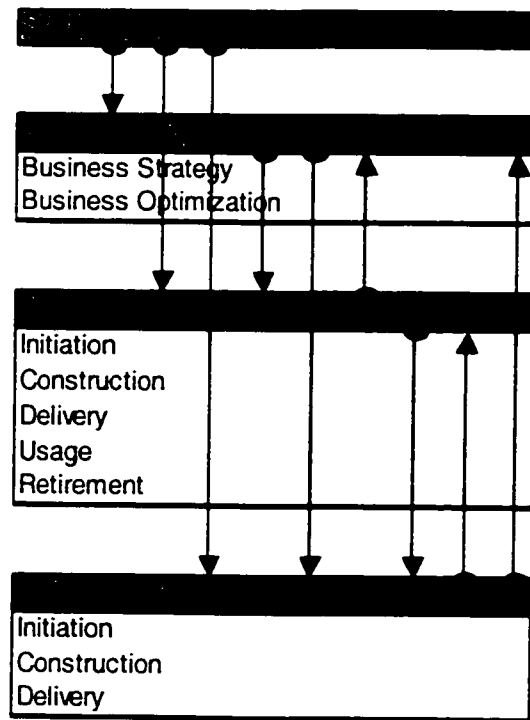


Figure 10 The State Transition Diagram of Cycles and Phases

Figure 11 demonstrates the *Context Maps* of {Cycles} and {Phases}:

	1	2	3	144	145	146	147	184	185	186
1								184	1	{Context Tuple Unique Id}
2	1	2	3	144	145	146	147	184		Id
3								184	125	{Context View}
4	v	v	v					143		Repository Organization:
5	v	v	v					143		Process Framework:
6	v	v	v					41		<u>Stages</u>
8				v	v	v	v	41		Process:
9	v	v	v	v	v	v	v	7		<u>Cycles</u>
129	F	F	F					44	4	{Cycles}
130								38		Start
131				t		t	t	13		<u>Business Engineering Cycle</u>
132				t	t		t	34		<u>Application Life Cycle</u>
133				t	t	t		31		<u>Application Development Cycle</u>
134				S	S	S	S	42	7	{Phases}
135					1			8		<u>Business Strategy</u>
136					2			4		<u>Business Optimization</u>
137						1	1	9		<u>Initiation</u>
138						2	2	6		<u>Construction</u>
139						3	3	12		<u>Delivery</u>
140						4		6		<u>Usage</u>
141						5		9		<u>Retirement</u>
1068	N	N	N	N	N	N	N	184	248	{Topics}
1069	v				v			2		<u>Definition</u>
1070	v				v			2		<u>Goals</u>
1071	v				v			2		<u>Objectives</u>
1072	v				v			2		<u>Phases</u>
1073	v				v			2		<u>Figure</u>
1074	v				v			2		<u>Guidelines</u>
1075		v				v		2		<u>Definition</u>
1076		v				v		2		<u>Goals</u>
1077		v				v		2		<u>Objectives</u>
1078		v				v		2		<u>Phases</u>
1079		v				v		2		<u>Figure</u>
1080		v				v		2		<u>Guidelines</u>
1081			v				v	2		<u>Definition</u>
1082			v				v	2		<u>Goals</u>
1083			v				v	2		<u>Objectives</u>
1084			v				v	2		<u>Phases</u>
1085			v				v	2		<u>Figure</u>
1086			v				v	2		<u>Guidelines</u>
1317								184	2	{Author}
1318	v	v	v	v	v	v	v	184		Syntax and Patterns © by W. M. Jaworski, 1988-2002
1319	v	v	v	v	v	v	v	184		Map by Jing Bai 2002

Figure 11 The *Context Maps* of Cycles and Phases

According to Figure 11, the semantics of {Cycles} and {Phases} view are described as follows:

The *OPF* repository of process components contains three {Cycles}: (Row129)

- 1) Business Engineering Cycle: (Row131)
 - Business Strategy Phase (Row135)
 - Business Optimization Phase (Row136)
- 2) Application Life Cycle: (Row132)
 - Initiation Phase (Row137)
 - Construction Phase (Row138)
 - Delivery Phase (Row139)
 - Usage Phase (Row140)
 - Retirement Phase (Row141)
- 3) Application Development Cycle: (Row133)
 - Initiation Phase (Row137)
 - Construction Phase (Row138)
 - Delivery Phase (Row139)

4.3 Milestones: Business Strategy Phase View

The business strategy phase is the first phase of the business engineering cycle. during which the development organization develops new strategies and architectures for all or part of the customer organization's business enterprise.

Figure 12 illustrates the order and approximate timing of preceding business strategy phase milestones:

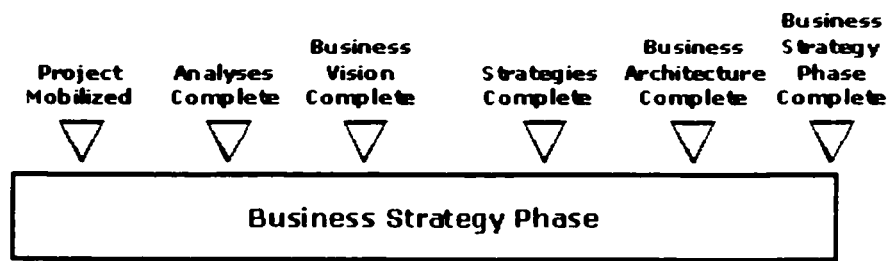


Figure 12 The Milestones Diagram of {Business Strategy Phase}

The relationships between the milestones of {Business Strategy Phase} in *OPEN Process Framework* are illustrated as the following state machine:

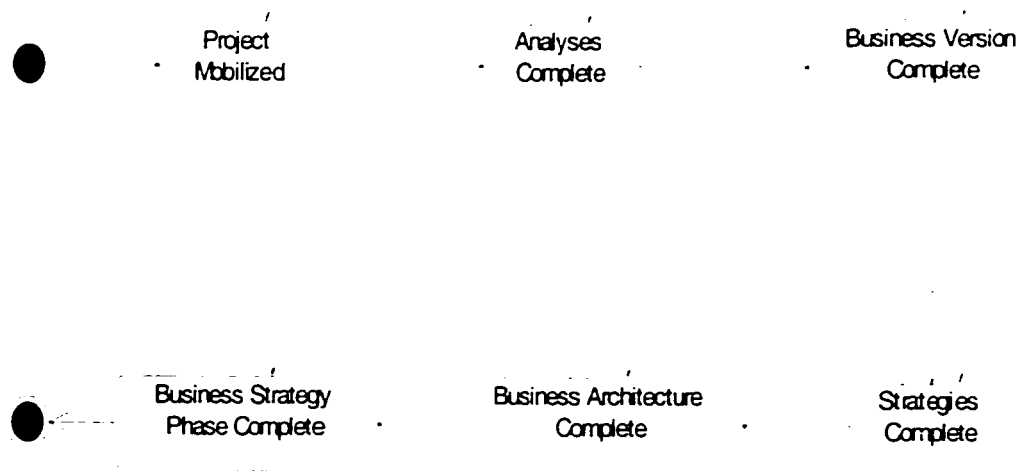


Figure 13 The State Machine of Milestones in {Business Strategy Phase}

Figure 14 illustrates the *Context Maps* of {Business Strategy Phase} in *OPEN Process Framework*:

	5	6	7	8	9	10	148	150	151	152	153	154	184	185	186
1													184	1	{Context Tuple Unique Id}
2	5	6	7	8	9	10	148	149	150	151	152	153	184		Id
3													184	125	{Context View}
4	v	v	v	v	v	v							143		Repository Organization:
5	v	v	v	v	v	v							143		Process Framework:
6	v	v	v	v	v	v							41		<u>Stages</u>
8							v	v	v	v	v	v	41		Process:
10							v	v	v	v	v	v	38		<u>Phases</u>
11	v	v	v	v	v	v	v	v	v	v	v	v	12		<u>Business Strategy</u>
129													44	4	{Cycles}
130							v	v	v	v	v	v	38		<u>Start</u>
131							v	v	v	v	v	v	13		<u>Business Engineering Cycle</u>
133							v	v	v	v	v	v	31		<u>Application Development Cycle</u>
134													42	7	{Phases}
135							v	v	v	v	v	v	8		<u>Business Strategy</u>
142	f	f	f	f	f	f							12	6	{Business Strategy}
143	f												2		<u>Project Mobilized</u>
144							-t						3		<u>Analyses Complete</u>
145								t					3		<u>Business Vision Complete</u>
146									t				3		<u>Strategies Complete</u>
147										t			3		<u>Business Architecture Complete</u>
148											t		3		<u>Business Strategy Phase Complete</u>
186													74	294	{Work Products}
187													7		Management Set:
190													3		• <u>Contract</u>
195													3		• <u>Job Description</u>
196													7		• <u>Management Plan</u>
199													3		• <u>Organization Chart</u>
201													7		• <u>Risk Management Plan</u>
202													3		• <u>Statement Of Work</u>
203													3		• <u>Status Report</u>
205													3		• <u>Work Breakdown Structure</u>
206													5		Configuration Mgmt Set:
207													7		• <u>Config Management Plan</u>
213													4		Quality Set:
219													4		• <u>Quality Plan</u>
222													3		Process Set:
223													3		• <u>Process Description</u>
225													3		Metrics Set:
226													3		• <u>Metrics Plan</u>
229													4		Environments Set:
230													3		• <u>Project Environment Description</u>
233													3		• <u>Development Environments:</u>
266													7		Requirements Set:
267													2		• <u>Actor Card</u>
268													2		• <u>Use Case Card</u>
269													2		• <u>Customer Analysis</u>
270													2		• <u>Customer Stakeholder Profile</u>
271													2		• <u>Market Analysis</u>
272													2		• <u>Competitor Profile</u>
273													2		• <u>Technology Analysis</u>

274										2		• User Analysis
275										2		• User Profile
276										2		• User-Task Matrix
280										2		• CRC Card
481					S	S	S	S	S	61	158	{Activities}
482					V					5		Management:
489					V					4		Configuration Management:
496					V					4		Risk Management:
512					V					4		Environments Engineering:
517					V					4		Metrics Engineering:
521					V					3		Process Engineering:
524					V					5		Quality Engineering:
542						V				3		Requirements Engineering:
583					V					7		Testing:
640					S	S	S	S	S	57	259	{Tasks}
649					V					5		Management Tasks:
650					V					3		Communications Management
652					V					6		Delivery Management
653					V					3		Management Planning
655					V					3		Relationship Management
656					V					3		Resource Management
659					V					4		Configuration Management Tasks:
660					V					3		Configuration Planning
666					V					4		Risk Management Tasks:
667					V					4		Risk Identification
668					V					4		Risk Analysis
669					V					4		Risk Documentation
691					V					4		Environments Engineering Tasks:
692					V					3		Environments Needs Assessment
693					V					3		Vendor And Tool Evaluation
694					V					3		Vendor And Tool Selection
695					V					3		Tool Acquisition
696					V					3		Environment Design
697					V					4		Environments Production
699					V					4		Metrics Engineering Tasks:
701					V					4		Metrics Planning
705					V					3		Process Engineering Tasks:
706					V					3		Process Needs Assessment
707					V					3		Process Framework Extension
708					V					3		Process Framework Iteration
709					V					3		Process Construction
710					V					3		Process Tailoring
711					V					3		Process Documentation
714					V					4		Quality Engineering Tasks:
715					V					3		Quality Planning
771						V				3		Requirements Engineering Tasks:
782						V				2		• Requirements Reuse
787						V				3		• Requirements Analysis
788						V				3		• Requirements Specification
833					V					7		Testing Tasks:
834					V					3		Test Planning
900					N	N	N	N	N	38	11	{Organizations}

901							v	v								18		Customer Organization
903							v									5		Maintenance Organization
905							v									4		Operations Organization
908							v									8		Subcontractor Organization
911							v									3		Vendor Organization
912							N	N	N	N	N	N	N	N		44	63	(Teams)
916							v									13		• Project Team
917							v									8		Management Teams:
918							v									8		• Enterprise Management Team
919							v									9		• Program Management Team
920							v									17		• Project Management Team
921							v									8		• Contact Center Management Team
922							v									8		• Data Center Management Team
923							v									8		• Reuse Center Management Team
924							v									6		• Change Control Board
925							v									6		• Configuration Management Team
931								v								7		• Business Strategy Team
932								v								7		• Technology Strategy Team
934								v								13		• Architecture Team
937							v									7		• Environments Team
939							v									14		• Independent Test Team
940							v									14		• Integration Team
941							v									11		• Metrics Team
943							v									8		• Process Team
944							v									4		• Quality Team
945								v								10		• Requirements Team
947							v									9		• Security Team
948							v	v								16		• Software Development Team
950							v	v								15		• User Experience Team
952							v									6		• Content Management Team
962							v									6		• Environments Inspection Team
965							v									7		• Management Inspection Team
966								v								8		• Requirements Inspection Team
969								v								8		• Strategy Inspection Team
976							N	N	N	N	N	N	N	N		49	79	(Roles)
1007								v								2		Domain Expert
1025							v	v								19		• Customer Representative
1027							v									7		• Subcontractor Representative
1030							v									7		• Vendor Representative
1035							v									7		• Program Manager
1036							v	v								18		• Project Manager
1041							v	v								19		• Technical Leader
1056							N	N	N	N	N	N	N	N		38	11	(Tools)
1068	N	N	N	N	N	N	N	N	N	N	N	N	N	N		184	248	(Topics)
1317																184	2	(Author)
1318	v	v	v	v	v	v	v	v	v	v	v	v	v	v		184		Syntax and Patterns © by W. M. Jaworski, 1988-2002
1319	v	v	v	v	v	v	v	v	v	v	v	v	v	v		184		Map by Jing Bai 2002

Figure 14 The Context Maps of {Business Strategy Phase}

According to Figure 14, the semantics of {Business Strategy Phase} are summarized as follows:

The typical milestones of the business strategy phase of a business engineering project include:

- Project Mobilized (Row143)
- Analyses Complete (Row144)
- Business Vision Complete (Row145)
- Strategies Complete (Row146)
- Business Architecture Complete (Row147)
- Business Strategy Phase Complete (Row148)

“Project Mobilized” is the first milestone of a project by which time the project staff should be mobilized to start work. The following {Work Products} (Row186) are typically delivered to the customer organization at or before this milestone:

- 1) Management Set: (Row187)
 - Contract (Row190)
 - Job Description (Row195)
 - Management Plan (Row196)
 - Organization Chart (Row199)

- Risk Management Plan (Row201)
 - Statement of Work (Row202)
 - Status Report (Row203)
 - Work Breakdown Structure (Row205)
- 2) Configuration Management Set: (Row206)
- Configuration Management Plan (Row207)
- 3) Quality Set: (Row213)
- Quality Plan (Row219)
- 4) Process Set: (Row222)
- Process Description (Row223)
- 5) Metrics Set: (Row225)
- Metrics Plan (Row226)
- 6) Environments Set: (Row229)
- Project Environment Description (Row230)
 - Development Environments (Row233)

The following {Tasks} (Row640) are typically accomplished by the development organization prior to this milestone:

- 1) Management Tasks: (Row649)
- Communications Management (Row650)
 - Delivery Management (Row652)

- Management Planning (Row653)
 - Relationship Management (Row655)
 - Resource Management (Row656)
- 2) Configuration Management Tasks: (Row659)
- Configuration Planning (Row660)
- 3) Risk Management Tasks: (Row666)
- Risk Identification (Row667)
 - Risk Analysis (Row668)
 - Risk Documentation (Row669)
- 4) Environments Engineering Tasks: (Row691)
- Environments Needs Assessment (Row692)
 - Vendor and Tool Evaluation (Row693)
 - Vendor and Tool Selection (Row694)
 - Tool Acquisition (Row695)
 - Environments Design (Row696)
 - Environments Production (Row697)
- 5) Metrics Engineering Tasks: (Row699)
- Metrics Planning (Row701)
- 6) Process Engineering Tasks: (Row705)
- Process Needs Assessment (Row706)

- Process Framework Extension (Row707)
 - Process Framework Iteration (Row708)
 - Process Construction (Row709)
 - Process Tailoring (Row710)
 - Process Documentation (Row711)
- 7) Quality Engineering Tasks: (Row714)
- Quality Planning (Row715)
- 8) Testing Tasks: (Row833)
- Test Planning (Row834)

4.4 Work Products View

A work product is an *OPF* process component that models anything of value (e.g. document, diagram, application, software component, hardware component) that is produced by one or more collaborating producer during the performance of one or more task. During the evolution of a work product, increasingly complete and higher quality versions of work products are created. Therefore, each work product has an associated time-ordered set of work product versions.

Figure 15 shows the *Context Maps* of {Work Products}:

																	184	1	(Context Tuple Unique Id)									
148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	184		Id
																	184	125	(Context View)									
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	v	41		Process:
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	v	38		Phases
v	v	v	v	v	v																					12		Business Strategy
				v	v																					4		Business Optimization
						v	v	v	v	v	v															12		Initiation
										v	v	v														6		Construction
																v	v	v	v	v	v	v	v	v	v	18		Delivery
																	44	4	(Cycles)									
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	v	38		Start
v	v	v	v	v	v	v																				13		Business Engineering Cycle
						v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	34		Application Life Cycle
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	v	31		Application Development Cycle
																	42	7	(Phases)									
v	v	v	v	v	v																					6		Business Strategy
					v	v																				4		Business Optimization
						v	v	v	v	v	v															9		Initiation
										v	v	v														6		Construction
																v	v	v	v	v	v	v	v	v	v	12		Delivery
																	12	6	(Business Strategy)									
																										2		Project Mobilized
t																										3		Analyses Complete
	t																									3		Business Vision Complete
		t																								3		Strategies Complete
			t																							3		Business Architecture Complete
				t																						3		Business Strategy Phase Complete
																	4	2	(Business Optimization)									
																										2		Brand Communicated
					t																					3		Reengineered Business Communicated
																	12	6	(Initiation)									
																										2		Project Mobilized
								t																		3		Significant Requirements Specified
									t																	3		Initial Architectures Documented
										t																3		End-To-End Prototype Validated
											t															3		Following Phases Planned
												t														3		Initiation Phase Complete
																	6	3	(Construction)									
																										2		Requirements & Architecture Frozen
												t														3		Application Implemented
													t													3		Construction Phase Complete
																	18	9	(Delivery)									
																										2		Data Center Established
																						t				3		Production Environment Established
																							t			3		Delivery Phase Documentation Delivered
																							t			3		Customer User Training Delivered
																								t		3		Application Deployed
																								t		3		Alpha Tests Passed
																									t	3		Beta Tests Passed

4.6 Producers View

A producer is an *OPF* process component that models something that performs tasks:

- Produce (e.g. create, evaluate, iterate, or maintain), either directly or indirectly, versions of related work products.
- Provide, either directly or indirectly, one or more services.

The *OPF* class library of reusable process components contains the following predefined classes and instances of producers, organized firstly by inheritance and secondly composition:

- **Indirect Producers:** which are producers that indirectly produce work products or provide services in terms of their component producers:
 - **Organizations:** which are relatively large-scale indirect producers.
 - **Teams:** which are relatively small-scale indirect producers.
- **Direct Producers:** which are producers that directly produce work products or provide services:
 - **Roles:** which are direct producers that model a cohesive collection of responsibilities representing a single part that a person can play within the process.
 - **Tools:** which are direct producers representing software applications (or possibly hardware devices) that are used by people playing roles to create,

modify, evaluate, or manage versions of work products (typically documents, software components, data components).

- **Persons:** which are direct producers modeling individual human beings that play one or more roles on an endeavor.

Figure 17 depicts the *Context Maps of {Producers}*:

																	184	1	{Context Tuple Unique Id}									
148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	184	125	{Context View}
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	v	v	41	Process:
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	v	v	38	Phases
v	v	v	v	v	v																						12	Business Strategy
				v	v																						4	Business Optimization
						v	v	v	v	v	v																12	Initiation
												v	v	v													6	Construction
																	v	v	v	v	v	v	v	v	v	v	18	Delivery
																	44	4	{Cycles}									
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	v	v	38	Start
v	v	v	v	v	v	v	v																				13	Business Engineering Cycle
						v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	34	Application Life Cycle
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	v	v	31	Application Development Cycle
																	42	7	{Phases}									
v	v	v	v	v	v																						8	Business Strategy
					v	v																					4	Business Optimization
						v	v	v	v	v	v																9	Initiation
												v	v	v													5	Construction
																	v	v	v	v	v	v	v	v	v	v	12	Delivery
																	12	6	{Business Strategy}									
																											2	Project Mobilized
t																											3	Analyses Complete
	t																										3	Business Vision Complete
		t																									3	Strategies Complete
			t																								3	Business Architecture Complete
				t																							3	Business Strategy Phase Complete
																	4	2	{Business Optimization}									
																											2	Brand Communicated
																											3	Reengineered Business Communicated
																	12	6	{Initiation}									
																											2	Project Mobilized
																											3	Significant Requirements Specified

4.7 Advantages of Converting OPF to a 3P-able Format by Context Maps

Context Maps is a powerful technology for representing systems architecture, structures, and processes. The merits of *Context Maps* are summarized as follows:

- **Plug-able:** *Context Maps* can easily integrate new contents and relationships into one map, even different kind of processes for comparison.
- **Process-able:** By using different syntax in spreadsheet, it is feasible and convenient to describe and process conceptual knowledge.
- **Pattern search-able:** By using the logical query of context tools, it is convenient to retrieve and extract the specific information that users expect to search from the maps.
- **Simple:** The syntax and semantics of *Context Maps* is very simple. Only minimal number of syntactical constructions is needed in the model.
- **Powerful:** By applying *Context Maps* technology, the information structure is rewritten from narratives into a knowledge frame, and create schema view of the *Context Maps* model. This provides *Context Maps* notation with modeling capability and power.
- **Multilingual:** The syntax and schema of *Context Maps* support multilingual modeling. *Context Maps* can incorporate instances, concepts, roles, knowledge

tuples, views and templates. It can be applied in many domains, such as modeling, mining, and evaluation of contexts, enterprises, methods, processes, projects, artifacts, databases, websites, information systems, and knowledge models with generic templates, domain experts, and proprietary notational technology.

- **Wide-spectrum:** *Context Maps* is a wide-spectrum notation, which from meta-level and macro-level to micro-level. The meta-level is used to represent deep structure of organizations and systems, and the micro-level is needed to define the executable specifications and source code. The notation supports all phases of system development process, including recovery and enhancement.
- **Machine-readable:** *Context Maps* is a kind of format, which can be easily read by machines.
- **Compactable:** Compared to diagrams, *Context Maps* is very compact to present a rich context within limited space.
- **Query-able:** By using context query tools, it is flexible to query on both high level design schema and specific elements.
- **Reusable:** Many elements and schema can be reused by modifications, and can be merged into other schemas.

- Transferable: If the data reach the limit of Excel spreadsheet, it can be transferred into other commercial databases, such as Access or SQL.
- Flexible: There are many different ways to representing the same knowledge. It is a challenge to find the easiest and simplest schema to represent the knowledge.
- Extendable: *Context Maps* allows users to create new notations and syntax according to their requirements.
- Comparable: It is more convenient for users to compare the similarities and differences between the processes by applying *Context Maps* technology than original materials.
- Consistency: The cardinality capabilities combined the well-defined notations makes it possible to check consistence in the schema.

Chapter 5

5. Comparing Open Process Framework with Rational Unified Process

5.1 Rational Unified Process Definition

The *Rational Unified Process* is a software engineering process, delivered through a web-enabled, searchable knowledge base [1]. The process enhances team productivity and delivers software best practices via guidelines, templates, and tool mentors for all critical software life cycle activities.

Note that *RUP* is a tailor-able process rather than an extensible and tailor-able process framework like *OPF* [3]. *RUP* is largely restricted to software development, and is therefore weak when it comes to systems development and business reengineering topics such as digital branding and content management. In general, *RUP* is less flexible and less complete than *OPF* [3].

5.2 Major Differences between OPF and RUP

The *OPF* is a process framework named after its open philosophy and volunteer consortium, while *RUP* is a process named after its for-profit company. The following

subsections and their associated tables summarize the differences between *OPF* and *RUP*:

5.2.1 Phase

The following table summarizes the major differences of {Phases} between *OPF* and *RUP*:

OPF	RUP	Rationale
Discovery	Missing	OPF covers entire delivery cycle
Strategy	Missing	OPF addresses customer's overall e-strategy
Inception	Inception	OPF emphasis on e-market systems
Elaboration	Elaboration	OPF emphasis on e-market systems
Construction	Construction	OPF emphasis on e-market systems
Transition	Transition	OPF emphasis on e-market systems
Usage	Missing	OPF covers entire delivery cycle

Table 1 Major Differences of {Phases} between *OPF* and *RUP*

5.2.2 Work Products

The following table summarizes the major differences of {Work Products} between *OPF* and *RUP*:

OPF	RUP	Rationale
Work Products	Artifacts	More traditional and understandable term
Strategy Document	Missing	OPF addresses customer's overall e-strategy
Application Vision Statement	Application Vision Statement	Roughly similar but OPF Application Vision Statement is more complete
System Requirements Specification	Use Case Model & Supplementary Specifications	More balanced treatment of operational and quality requirements
Project Glossary	Project Glossary	OPF differentiates domain and technical definitions
Domain Model Document	Requirement Document	OPF emphasis on e-market systems
System Architecture Document	System Architecture Document	OPF emphasizes systems development
Software Architecture Document	Software Architecture Document	Very similar

Table 2 Major Differences of {Work Products} between *OPF* and *RUP*

OPF has more types of “Work Products” than “Artifacts” in *RUP*. *OPF* is stronger in the areas of “Digital Branding”, “Content Management”, and “Business Engineering”. The following table summarizes the major differences of {Work Products} sets between *OPF* and *RUP*:

Work Product Sets (OPF Terminology)	OPF WPs	RUP WPs	Comparisons
Management	15	11	OPF strong
Configuration Management	6	3	OPF strong
Risk Management	2	2	TBD
Disaster Recovery	4	0	Not addressed by RUP
Training	6	1	OPF strong
Quality Engineering	5	1	OPF strong
Process Engineering	100+	40	OPF strong
Metrics Engineering	2	2	TBD
Environments Engineering	12	0	Not addressed by RUP
Security Engineering	2	0	Not addressed by RUP
Digital Branding	6+	0	Not addressed by RUP
Requirements	20	18	Both strong

Architecture	17	19	Both strong
Design	7	14	RUP strong
Implementation	11	1	OPF strong
Integration	4	2	OPF strong
Testing	15	8	OPF strong
Deployment	13	8	OPF strong
Content Management	2	0	Not addressed by RUP
Maintenance	0	0	TBD
Operations	1	0	Not addressed by RUP
User Support	0	0	Not addressed by RUP
Retirement	0	0	TBD
Model	4	0	RUP models are spread among requirements, architecture, design, and implementation

Table 3 Major Differences of {Work Products} Sets between *OPF* and *RUP*

5.2.3 Terminology

The following table summarizes the major differences of terminology between *OPF* and *RUP*:

OPF	RUP	Rationale
Work Product	Artifact	Work product is the traditional term; artifacts are what you have left after everyone dies
Role	Worker	Role is more descriptive; not every role is a worker
Process Framework	Single Process	OPF is more flexible
Systems	Software	OPF is more specific
E-markets Emphasis	General Purpose	OPF emphasis on e-markets
Delivery Cycle	Development Cycle	Must cover pre/post sales
Quality Requirements	Primarily Use Cases	Quality requirements impact architecture
Covers Roles and Teams	Only covers roles	OPF is more complete
Also XP, RDD, DBC	Three Amigos	OPF is more complete

Table 4 Major Differences of Terminology between *OPF* and *RUP*

5.3 OPEN Process Framework Strengths

OPF is recognized as having the following strengths relative to other development processes:

- *OPF* is supported by a large powerful corporation with large development and maintenance staffs, impressive sales, and marketing budgets.
- *OPF* is supported by a large web-based tool that is integrated into a set of related development tools.
- *OPF* is the de facto industry standard based on:
Sales of the *OPF* tool, numerous popular *OPF* books and articles, heavy direction setting and content input by the famous methodologists, significant partnerships with important consultants and methodologists.
- *OPF* helps standardize process terminology, a process metamodel, and associated notation (e.g. UML).
- *OPF* has a clear philosophical organization build around major process decisions:
OPF is use case driven.

5.4 RUP Weaknesses Addressed by OPF

RUP is widely controversial and seen as having the following weaknesses relative to the *OPEN Process Framework (OPF)*:

- Based on a static foundation of techniques and methodologies.
- Originally designed for software product development.
- Not easily tailored to suit e-market development.

- Insufficient support for teams and organizational structure.
- Insufficient component integration and reuse.
- Overemphasis of use cases.
- Too strongly oriented towards Rational tool set.

Chapter 6

6. Conclusion and Recommendation

6.1 General Conclusions

The following conclusions are drawn from the study of the major report:

- 1) *Context Maps* enables us to create virtual information maps for knowledge base system. It is a notation and method for representing system architecture, structure, processes, and reusable templates. *Context Maps* notation allows users easily to recovery and generates model of generic schema for objects, views, and processes of information systems.
- 2) *Context Maps* is a technology with a 3P-able notation, namely, Plug-able, Process-able, and Pattern search-able. *Context Maps* can easily integrate different knowledge into one consistent map. By using different syntax in spreadsheet, it is feasible to describe and process conceptual knowledge. By using the logical query of context tools, it is convenient to retrieve the specific information that users expect to search from the map.
- 3) *OPEN Process Framework* is a practical, public-domain, industry-standard, general-purpose management, and engineering process framework that is primarily intended for the object-oriented, component-based development of software-intensive systems.

This methodology can be converted to a 3P-able format by applying *Context Maps* technology based on concepts and relationships.

- 4) By applying *Contexts Maps* technology to represent *OPF* has a lot of advantages. It is more convenient for users to compare *OPEN Process Framework* with *Rational Unified Process*.

6.2 Recommendations for Future Works

From the results of this report, there are still some further works need to be carried out to refine and improve *Context Maps* technology. The following are recommended for the future enhancements:

- 1) There is much work for *Context Maps* to deal with complex systems. Future work is expected to improve a better and more complete theory of *Context Maps*.
- 2) Converting knowledge to *Context Maps* is still done by domain experts manually. There are possibilities to accomplish the transformations automatically in the future.
- 3) For large amounts of data, using Excel as an application environment of *Context Maps* has limitations, since only 256 columns are available in a spreadsheet. By considering this issue, develop a more efficient method for storing context tuples is another potential work.

Bibliography

- 1) **Rational Unified Process**, <http://www.rational.com>.
- 2) **W. M. Jaworski, Michailidis A. A.**, "Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", ATW94, University of Massachusetts - Lowell, Lowell, Massachusetts, June 1994.
- 3) **OPEN Process Framework**, <http://www.donald-firesmith.com>.
- 4) **W. M. Jaworski**, "Comp 657 Course Notes", Concordia University, 2000.
- 5) **W. M. Jaworski**, "*JMaps: Conceptual Spreadsheets for Data and Knowledge*". Warehousing, 1995.
- 6) **W. M. Jaworski**, "InfoMaps: Conceptual Spreadsheets for Data and Knowledge Warehousing", ATW95 - USA1995, University of New Hampshire, Durham, New Hampshire, May 31 - June 1, 1995.
- 7) **W. M. Jaworski**, et al. "The ABL/W4 Methodology for System Modeling". System Research Journal 4(1), 23 - 37, 1987.
- 8) **W. 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) **W. M. Jaworski**, et al. "Representing Processes, Schemata and Templates with *Context Maps*", Semiotica 125(1/3), 229 - 247, 1999.
- 10) **General Strategies Inc.**, <http://www.gen-strategies.com>.

11) **Context Maps,**

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

12) **W. M. Jaworski,** "System Analysis and Design in the Classroom: InfoMaps Teaching Factory", Modeling and Simulation Conference, Pittsburgh, Pa., May3 - 4, 1990.

13) **W. M. Jaworski,** "Cooperative Engineering Issues by Examples: Mapping of Mil-498 and NSDIR with *Context Maps*", ATW96 - USA1996, Electronic Systems Center, Hanscom Air Force Base, August 6 - 9, 1996.

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

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

16) **James Rumbaugh, etc.,** "Object-Oriented Modeling and Design". Prentice-Hall, Inc., 1991.

17) **John M. Pierre,** "On the Automated Classification of Websites", Linköping Electronic Articles in Computer and Information Science. Vol. 6(2001), Linköping University Electronic Press, Linköping, Sweden, 2001.

18) **Rob Kremer,** "A Concept Map Meta-Language".

<http://www.cpsc.ucalgary.ca/~kremer/dissertation/index.html>.

19) **Joseph D. Novak,** "The Theory Underlying Concept Maps and How to Construct them", <http://cmap.coginst.uwf.edu/info/printer.html>.

20) **Concordia University,** "Thesis Preparation and Thesis Examination Regulations", http://www-gradstudies.concordia.ca/SGS_WWW/publications.html.

Appendix

The *Context Maps* of *OPEN Process Framework* is too large to be viewed in one diagram. By querying the keywords in {Context View} set, it can be spilt into several figures. The following *Context Maps* are different views from *OPEN Process Framework*:

A-1 Stages: Cycles and Phases View

	5	6	7	8	9	10	148	150	151	152	153	154	184	185	186
1													184	1	(Context Tuple Unique Id)
2	5	6	7	8	9	10	148	149	150	151	152	153	184		Id
3													184	125	(Context View)
4	v	v	v	v	v	v							143		Repository Organization:
5	v	v	v	v	v	v							143		Process Framework:
6	v	v	v	v	v	v							41		<u>Stages</u>
8							v	v	v	v	v	v	41		Process:
10							v	v	v	v	v	v	38		<u>Phases</u>
11	v	v	v	v	v	v	v	v	v	v	v	v	12		<u>Business Strategy</u>
129													44	4	(Cycles)
130							v	v	v	v	v	v	38		Start
131							v	v	v	v	v	v	13		<u>Business Engineering Cycle</u>
133							v	v	v	v	v	v	31		<u>Application Development Cycle</u>
134													42	7	(Phases)
135							v	v	v	v	v	v	8		<u>Business Strategy</u>
142	F	F	F	F	F	F							12	6	(Business Strategy)
143	t												2		<u>Project Mobilized</u>
144		t					t						3		<u>Analyses Complete</u>
145			t					t					3		<u>Business Vision Complete</u>
146				t					t				3		<u>Strategies Complete</u>
147					t					t			3		<u>Business Architecture Complete</u>
148						t					t		3		<u>Business Strategy Phase Complete</u>
186													74	294	(Work Products)
187													7		Management Set:
190													3		• <u>Contract</u>
195													3		• <u>Job Description</u>
196													7		• <u>Management Plan</u>
199													3		• <u>Organization Chart</u>
201													7		• <u>Risk Management Plan</u>

A-5 Producers View

																	184	1	{Context Tuple Unique Id}											
148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	184		Id		
																	184	125	{Context View}											
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	v	v	41		Process:	
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	v	v	38		Phases	
v	v	v	v	v	v																						12		Business Strategy	
				v	v																						4		Business Optimization	
					v	v	v	v	v	v																		12		Initiation
										v	v	v																6		Construction
																	v	v	v	v	v	v	v	v	v	v	v	18		Delivery
																	44	4	{Cycles}											
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	v	v	38		Start	
v	v	v	v	v	v	v	v																					13		Business Engineering Cycle
						v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	34		Application Life Cycle
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	v	v	v	31		Application Development Cycle
																	42	7	{Phases}											
v	v	v	v	v	v																							6		Business Strategy
					v	v																						4		Business Optimization
						v	v	v	v	v	v																	7		Initiation
											v	v	v															6		Construction
																	v	v	v	v	v	v	v	v	v	v	v	12		Delivery
																	12	6	{Business Strategy}											
																												2		Project Mobilized
t																												3		Analyses Complete
	t																											3		Business Vision Complete
		t																										3		Strategies Complete
			t																									3		Business Architecture Complete
				t																								3		Business Strategy Phase Complete
																	4	2	{Business Optimization}											
																												2		Brand Communicated
																												3		Reengineered Business Communicated
																	12	6	{Initiation}											
																												2		Project Mobilized
																												3		Significant Requirements Specified
																												3		Initial Architectures Documented
																												3		End-To-End Prototype Validated
																												3		Following Phases Planned
																												3		Initiation Phase Complete
																	6	3	{Construction}											
																												2		Requirements & Architecture Frozen
																												3		Application Implemented
																												3		Construction Phase Complete
																	18	9	{Delivery}											
																												2		Data Center Established
																												3		Production Environment Established

