2007

Unified data mining engine as a system of patterns

Somenath Das
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Das, Somenath, "Unified data mining engine as a system of patterns" (2007). *Master's Theses*. 3440.
DOI: <https://doi.org/10.31979/etd.uc9c-8bb3>
https://scholarworks.sjsu.edu/etd_theses/3440

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

UNIFIED DATA MINING ENGINE AS A SYSTEM OF PATTERNS

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Somenath Das

August 2007

UMI Number: 1448865

INFORMATION TO USERS

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 bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send 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.

UMI[®]

UMI Microform 1448865

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

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

© 2007

Somenath Das

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER
ENGINEERING

Dr. Sigurd Meldal

Dr. Mohamed Fayad

Dr. Dan Harkey

APPROVED FOR THE UNIVERSITY

ABSTRACT

UNIFIED DATA MINING ENGINE AS A SYSTEM OF PATTERNS

by Somenath Das

The main goal of this thesis is to design and implement an efficient data mining engine based on the software stability model. This thesis follows the following important steps to build the system in a very systematic manner: first, multiple scenarios are selected (different data mining applications) to understand the problem. Then, the thesis will attempt to find out the core logic behind the data mining engine after analyzing those scenarios, which are the knowledge maps or knowledge core sets, and which also describe the topology of the problem in more stable way.

From the knowledge map, this thesis derives the EBTs (Enduring Business Theme) and the BOs (Business Object) to design the core engine; finally, this thesis describes use cases, where it shows how applications are developed based on the engine by using hooks (connection point between the application and the core engine).

ACKNOWLEDMENTS

I would like to express my sincere gratitude and appreciation to my supervisor Dr. M. E. Fayad for his active involvement, constant guidance, encouragement, and inspiration, without which this research work would not be carried out successfully. Every week he took some time out from his busy schedule to discuss my progress with the thesis work.

I also deeply thank Professor Dr. Sigurd Meldal and Professor Dr. Dan Harkey for serving on my graduate committee.

Special thanks to my lovely wife Soma for her help in editing the thesis and also acting as the father to our six-month old beautiful daughter Anaisha. I would also like to say sorry to my parents whom I could not call regularly due to the extensive work related to the thesis.

TABLE OF CONTENTS

LIST OF FIGURES	VIII
LIST OF TABLES	X
1 INTRODUCTION.....	1
1.1 Introduction	1
1.2 Problem	3
1.3 Solution	5
1.4 SSM overview	7
1.4.1 Enduring business themes (EBTs) – Establish the data mining goals	9
1.4.2 Business objects (BOs) – Identifying the workhorses for achieving the goals.....	9
1.4.3 Industrial objects (IOs).....	10
1.5 DM engine overview	11
1.6 Research methodology	15
1.7 How research is conducted.....	19
1.8 Thesis contributions	19
1.9 Thesis layout	21
2 COMPARATIVE STUDY	24
2.1 Introduction	24
2.2 Why data mining?	27
2.3 Existing data mining tools and environment	28
2.3.1 SPSS Clementine.....	32
2.3.2 KXEN.....	33
2.3.3 TreeNet.....	34
2.3.4 SAS data integration tool	35
2.3.5 Oracle data mining tool	36
2.4 Data mining process.....	37
2.5 Comparison of different approaches	38
2.6 Analysis.....	38
2.7 Conclusions	43
3 DM ENGINE BASIC BLOCKS	45
3.1 Introduction	45
3.2 Goals (EBTs).....	49
3.2.1 Discovery analysis pattern.....	49
3.2.2 Classification analysis pattern	51
3.2.3 Visualization analysis pattern.....	52
3.3 Capabilities.....	55
3.3.1 AnyInteraction design pattern	55
3.3.2 AnyPrecision design pattern.....	57
3.3.3 AnyModel design pattern	59
3.3.4 AnyCriteria design pattern	61
3.3.5 AnyData design pattern	62
3.4 What is software pattern?	65

3.5	What is knowledge map?	65
3.6	Implementation issues	71
3.7	Conclusions	72
4	THE DM ENGINE.....	74
4.1	Introduction	74
4.2	The data mining engine overview	76
4.2.1	Classification based on processes.....	77
4.2.2	Classification based on techniques	79
4.2.3	Classification based on domains	80
4.2.4	Classification based on outcomes.....	81
4.3	Data mining engine architecture.....	82
4.4	Describe elements of architecture	86
4.5	DM engine applications	89
4.5.1	Example application – Text mining for customer support email.....	91
4.6	The impact of DM engine	92
4.6.1	Data mining in business	93
4.6.2	Data mining in education	94
4.6.3	Data mining in medicine	95
4.7	Conclusions	96
5	CASE STUDY	98
5.1	Introduction	98
5.2	Scenario template	100
5.2.1	Use case: Data extraction	104
5.2.2	Use case: Discovery of features and clusters.....	109
5.2.3	Use case: Document classification	115
5.3	Conclusions	121
6	ANALYSIS & MAJOR FINDINGS	123
6.1	Introduction	123
6.2	Major findings	124
6.3	Conclusions	128
7	FUTURE WORK & CONCLUSIONS	129
7.1	Introduction	129
7.2	Future development.....	129
7.3	Conclusions	131
	REFERENCES	134
	APPENDIXES	137
	Appendix A: The Classification stable analysis pattern.....	137
	Appendix B: The Visualization stable analysis pattern.....	165
	Appendix C: The AnyPrecision stable design pattern.....	195
	Appendix D: Implementation & code	223
	Appendix E: Demo & instructions	247

LIST OF FIGURES

Figure 1 - Software stability model	8
Figure 2 - Layers in SSM.....	11
Figure 3 - Data mining engine processes.....	13
Figure 4 - Data mining engine development methodology	16
Figure 5 - Traditional data mining tool.....	42
Figure 6 - Data mining application based on UDME	42
Figure 7 - Data mining patterns	47
Figure 8 - Discovery analysis pattern	50
Figure 9 - Classification pattern.....	52
Figure 10 - Class diagram for the Visualization pattern	54
Figure 11 - Class diagram of AnyInteraction	56
Figure 12 - Class diagram for the AnyPrecision pattern.....	58
Figure 13- Class diagram for the AnyModel pattern	60
Figure 14 -Class diagram for AnyCriteria pattern	62
Figure 15 - Class diagram for AnyData pattern.....	64
Figure 16 - Knowledge map	68
Figure 17 - Application 1	69
Figure 18 - Application 2.....	70
Figure 19 - Data mining application	71
Figure 20 - Knowledge map	83
Figure 21 - Simple process diagram	85
Figure 22 - Complex process model	86
Figure 23 - Data mining patterns	87
Figure 24 - Data mining engine architecture	88
Figure 25 - Communication between engine and message bus	89
Figure 26 - Class diagram for a text mining application used in CRM	92
Figure 27 - Major steps in text mining.....	100
Figure 28 - Participant classes in the scenario	102
Figure 29 - Term/Feature extraction process.....	105
Figure 30 - Participant classes in the use case	107
Figure 31 - Scenario diagram.....	109
Figure 32 - Clustering process	111
Figure 33 - Class diagram	112
Figure 34 - Sequence diagram	115
Figure 35 – The classification process in text mining	116
Figure 36 - Class diagram	117
Figure 37 - Sequence diagram	120
Figure 38 - Class diagram of the Classification pattern.....	146
Figure 39 - Diagram of hurricane classification system based on SS scale.....	153
Figure 40 - Sequence diagram for hurricane classification	156

Figure 41 - Class diagram of a terrorist classification system.....	157
Figure 42 - Sequence diagram of terrorist classification system	160
Figure 43 - Class diagram of the Visualization pattern	175
Figure 44 - Using Visualization analysis pattern in GIS	182
Figure 45 - Sequence diagram for finding nearest hospital	185
Figure 46 - Using Visualization analysis pattern in health care application.....	186
Figure 47 - Sequence diagram for monitoring heartbeat of a patient	189
Figure 48 - Class diagram of the AnyPrecision pattern.....	204
Figure 49 - Class diagram of a precision navigation system for a submarine	211
Figure 50 - Sequence diagram of a precision navigation system for a submarine	214
Figure 51 - Class diagram of a text mining application.....	216
Figure 52 - Sequence diagram of a precision navigation system for a submarine	219
Figure 53 - Class diagram of a hurricane classification system.....	223
Figure 54 - Main application frame	248
Figure 55 - Category edit menu	249
Figure 56 - Hurricane data frame.....	250
Figure 57 - Hurricane screen after the data is classified	251

LIST OF TABLES

Table 1 - Comparison of existing data mining tools and UDME	41
Table 2 - List of goals and capabilities for data mining engine.....	47

Chapter 1

Introduction

1.1 Introduction

Data mining is the discovery of useful knowledge by analyzing enormous sets of data, by extracting their real meaning, and then later predicting their future trends. Data mining also helps us to find out secret information from large databases, and helps companies and firms to make proper decisions based on the available knowledge and information.

Stability patterns have emerged as a promising technique that introduces the innovative concept of the stability in the requirements, analysis, design, architecture, and testing of any knowledge. Design patterns concentrate mainly on a number of special techniques; usually, such techniques couple very tightly with their main code. On the other hand, the stability model provides a stable system of patterns that represents the core knowledge or knowledge map of any given domain (i.e., data mining). This core engine is highly stable over a period, and provides flexibility, reliability, and reusability to the software, thus successfully saving billions of dollars in repair and maintenance.

A typical data mining tool of the day has some common core logic, which is very independent of the available data and the applications; but most of existing implementations simply try to ignore that fact and try to concentrate on the specific problem. In this way, the engine becomes limited to a particular set of data for a specific application. To avoid this unseemly problem, this thesis has used the software stability

model pioneered by Dr. Fayad [B1, B7,B8, B9, B14, B15, and B17], to design the data mining engine that will work with many different sets of data, and is quite useful in any given domain.

Our research further expands on previous and ongoing research work on the concept of software stability model. Software stability model is a very novel idea in object-oriented design techniques. It differs entirely from any traditional model, by providing a fair amount of stability in software design. With the help of this modeling technique, this thesis has attempted to separate out the enduring business themes in data mining, which are different analysis patterns that tie together to provide a robust and stable data mining engine. SSM also helps to find out the business objects that reside inside the data mining engine, which could later build the core business logic. With the help of EBTs and BOs, this thesis is also able to design a data mining engine, which is very stable and flexible.

Software stability model has three essential layers: the Enduring Business Themes (EBTs) or goals layer, the Business Objects (BOs) or capabilities layer used to achieve the goals layer, and the Industrial Objects (IOs) layer or the application layer. EBTs always capture the enduring themes of the problem, which does not change over time; BOs encapsulate the business logic, which are stable internally, but externally adaptable; while, IOs implement the application specific logic, which changes over time and from application to application. Software stability contains a number of quality factors implicitly, such as testability, reusability, adaptability, and scalability. The system groups into many types specifically based on the nature and stability of the classes.

This chapter specifies the exact nature of problem that the thesis is trying to solve, and then explains the proposed solution. Next, the chapter provides an overview of the Software stability modeling technique, along with the proposed data mining engine. Later, it explains in detail, how the research is conducted along with the methodology used for the purpose. The end of this chapter provides many important contributions of this thesis in minute detail. The end section of this chapter also presents the layout of other chapters in this thesis.

1.2 Problem

Data mining is all about finding interesting patterns in the available set of data. The main challenge for any data mining engine is to find out ways how to apply different algorithms or different techniques, on different sets of data to find all interesting patterns, which are useful to businesses. However, it is quite difficult to arrive at some predefined and standard ways of analyzing the data. The stupendous volume and the sheer complexity of the data make it almost impossible to run same algorithms on different datasets. Nowadays, there are many vendors who are trying to solve this unique problem, but mostly they support a subset of different algorithms. None of them has designed any single stable engine, which can work in any data set and in any domain.

In the last decade or so, the improvement in storage and CPU speed have created wonderful opportunities for different data mining applications ranging from CRM to medical health care applications. Right now, it is very difficult to design and develop a single application, which can effectively take care all of these problems. It is a pure

dream even to think of an application that can iterate through any data to find meaningful patterns. Data mining also deals with useful patterns, not just patterns; now, whether a pattern is useful or not depends solely on the context where it is applied. Present day tools also depend on types of algorithms to apply and ways to analyze the output, because most of them are generic in nature, and there is no context specific logic attached to the application.

The problems of the existing data mining tools are:

- a) Difficult to use: Existing data mining tools try to cover every data mining applications, which subsequently makes them very difficult to configure and run.
- b) Need expert to run the tool: A given tool never binds any domain or problem specific logic; this will require an expert to run the tool and analyze the available result.
- c) Difficult to add new functionality: Because of the sheer size and complexity of each tool, it is very difficult to add new feature to the tool.
- d) Difficult to integrate: In many cases, algorithms developed by other companies are not compatible with the tools; hence, such algorithms are quite difficult to integrate with the available tools.
- e) Short lifetime: A given tool does not have any stable component, and with time, the tool will become obsolete. As new tools invade the market, changing the existing tool to incorporate new feature is difficult and requires lot of architectural changes.

- f) Limited number of algorithms: Existing tools just provide a limited number of algorithms, and in many cases, use of multiple algorithms is severely limited.
- g) Need lot of resources: It is very difficult to optimize the existing tools for any given specific application; thus, these tools need many other resources, such as runtime memory, hard disk etc.

1.3 Solution

This thesis proposes to analyze the underlying problem, by using the software stability model pioneered by Dr. M. Fayad [B1, B7, B8, B9, B14, B15, and B17]. Later, this thesis will attempt to arrive with a suitable architecture consisting of EBTs and BOs, as a system of patterns, which is stable over a period, and provide a very simple way to apply different data mining and data analysis algorithms on different sets of data in any given domain.

However, the questions that lie in front of us are how this thesis can achieve its main goals, and what it can do and perform that is entirely different from others. Existing vendors are more concerned with specific problems, and their tools focus mainly on specific formats of data, by tightly coupling and hemming their tools with the application logic. That is the reason why DM tools are not really stable and generic enough for different uses, whereas, our thesis focuses more on the stable part of the system, rather than the application specific logic. This thesis also designs the classes based on the principles of enduring themes and business logic. Each one of the application specific

logic is easily attachable to the engine to develop an application that is specific to a particular context or problem.

Here is a summary of some benefits of the proposed solution:

- a) Ease of use: It is easy to develop multiple tools, by focusing on some specific problems, because they all can share the same core services, thus avoiding the development of a large tool, which tries to solve all problems.
- b) No need of expert to run the tool: A particular tool can implement domain specific knowledge, such as verification and selection of tool, all the while developing the tool.
- c) Easy to add new functionality: The application specific logic is quite separate from the core logic, therefore, new application specific functionality can be added very easily, without making any change in the core logic.
- d) Easy to interface: The design relies on a system of independent patterns, and third party vendors can develop them.
- e) Long lifetime: The engine also depends on stable core logic, which has a long lifetime, while the application logic is very loosely connected, which can change over time.
- f) Multiple algorithms: The engine can support any number of algorithms.
- g) Fewer resources: By connecting several patterns from the knowledge map, it is quite easy to develop an engine. Depending on the application domains, the engine can also use different patterns, which are necessary, and that need fewer resources, when compared to existing tools.

The tools also provide extra benefits such as:

- a) **Stable:** The engine is specifically based on EBTs and BOs, as a system of patterns, which is stable over a period, and provides a simple way to apply different data mining and data analysis algorithms, on different sets of data in any given domain.
- b) **Isolation of application logic:** The research explained in this thesis has successfully isolated the stable knowledge from any application specific logic; therefore, different applications can use the same core knowledge, that never need repeated changes and modifications.
- c) **Independent component:** The design relies on a system of patterns, developed as independent components; thus, it is possible to develop them in a parallel manner.
- d) **Less maintenance cost:** Maintenance cost is minimal and low. Whenever there is a new requirement for an additional feature, the existing IOs are easy to replace by adding the new one, without making any changes in the core engine. Thus, there will be no maintenance task associated with the engine.

1.4 SSM overview

Software stability concept mainly uses a layer approach for developing stable software systems. Under this particular concept, the essential classes are the Enduring Business Themes (EBTs) layer, the Business Objects (BOs) layer, and the Industrial Objects (IOs) layer, as shown in Figure 1.

Figure 1 - Software stability model

Under the EBT layer, the classes always provide the enduring and core concepts of the underlying industry or business. BO layer always map the EBTs of the system into more clear and concrete objects. IO layer map the BOs of the system into many physical concrete objects. BOs are semi-conceptual and externally stable in nature, but they can change internally. More detail about EBTs, BOs, and IOs are described by Dr. Fayad in [B8].

1.4.1 Enduring business themes (EBTs) – Establish the data mining goals

EBTs represent the enduring business concepts of the system that will never change. They have the following properties as described by Dr. Fayad [B8]:

- a) Externally and internally stable: EBTs essentially encapsulate the core knowledge of the system. They are quite stable over time, and should never change, externally or internally.
- b) Conceptual objects: EBTs always represent concepts those are not tangible.
- c) Commonality to the domain: EBTs are very common to all of the applications of the domain, in which the EBTs considered as identified and labeled.

1.4.2 Business objects (BOs) – Identifying the workhorses for achieving the goals

BOs are the special classes that encapsulate the underlying business logic; they actually implement the concepts driven by EBT, into more concrete and definable objects. BOs are tangible and externally stable, but they are internally adaptable. BOs characterize a number of specific and special properties as described by Dr. Fayad [B8]. These properties are:

- a) Internally stable and externally adaptable: BOs can change externally but should never change internally.
- b) Semi-tangible objects: Compared to the EBTs, the BOs of the system are quite tangible; however, they are not concrete objects, therefore, considered semi-tangible objects.

- c) Commonality to the domain: BOs are very common to all of the applications of the domain, in which the BOs considered as identified and labeled.

1.4.3 Industrial objects (IOs)

IOs are the classes that map the BOs of the system into concrete and definable physical objects. IO has the following properties:

- a) Unstable: IOs present the physical objects of the system; and, it is possible to replace, add to the system, or even removed from the system, all the while, not affecting the core of the system as described by Dr. Fayad [B9].
- b) Concrete objects: The IOs are fully tangible objects. They usually represent real world objects.
- c) Commonality to the domain: IOs always stay at the outmost layer in the given SSM model. IOs can also be completely different from one application to another, even within the same domain.

The three layers in SSM are described in Figure 2.

Figure 2 - Layers in SSM

The next big task is the identification of the EBTs and BOs. To identify the EBTs and BOs, the engine is broken into different pieces, as independent domains. Different scenarios are then studied to find the common core logic, which in turn shapes the EBTs and BOs of the system, which can be referred as independent domains of knowledge, as knowledge map, which is binding logic of different component inside any application.

1.5 DM engine overview

Data mining is probably the hottest topic among the business circles. Why is this so important and critical? One of the main reasons is that one can predict the future by using the data mining concept, which often varies from stock market analysis to search

patterns. How can we predict future? Can we really do so? The simple approach here is going to an astrologer to seek the future truth! The most complex approach is designing a data mining engine that finds patterns in the existing data, and then predicts the future based on the discovered knowledge. This thesis depends on complex approaches, which aim at developing a data mining core engine, which is usable under different domains and in their applications areas. The implementation of the application specific logic on top of the core engine is as simple as going to an astrologer. To design such an engine, different processes need a thorough investigation, which is involved with the data mining process as shown in Figure 3.

Figure 3 – Data mining engine processes

The main goal of data mining is to find and detect the unseen patterns in the available data. Next question is from where these data usually come. There can be different sources based on many applications. Most of the applications usually store data in commercial databases. Those databases can be RDBMS, MMDBS, ODBMS, or just flat files. These data sets are not directly usable by the data mining engine. Therefore,

the first step is cleaning the available data and later the algorithms. The data mining suite also provides essential tools for performing all of these operations.

We have already mentioned earlier in this thesis that there is no generic engine designed to work with the different data mining algorithms. Most of them just focus on a particular problem. To solve that specific problem, this thesis develops an engine that not only provides support for different systems and applications, but also provides stability over a period, with the help of software stability modeling techniques, which are always stable over time, and usable by different applications.

Rather than simply designing a data mining engine that focuses on specific problem, the approach described in this thesis differs entirely from previous works in this area, by developing an engine that is usable in any data mining applications, as the core layer of logic. It effectively hides all the business logic in the form of more generic EBTs and BOs; therefore, depending on the nature of the application, IOs can be developed at a later stage, which can use this engine, with the help of hooks provided by us. This may lower the complexity and time needed for developing any data mining application from the ground level.

With this new and innovative approach, it is easy to build applications within a short period. As, time to market, a product is the main key to the success, and with the data mining engine described in this thesis, companies and firms can develop and market new applications quite quickly; in that way, they will be able to capture major market share. As time passes, market will also change, and with this changing trend, a particular

company can design and develop new IOs. By considering the new market trend, they can continue to use the proposed data mining engine.

1.6 Research methodology

In the research hypothesis, the thesis has used the software stability modeling techniques to find the basic building blocks of the data mining engine. SSM is the next level of components driven architecture. It also helps to find the stable components of the system, as EBTs and BOs. Identifying EBTs or the BOs is not an easy task; it always needs previous experience and proper understanding of the field.

There are four main steps in this research thesis:

- a) First, the problem area divides itself into multiple patterns. Each pattern has its own EBTs and BOs, which form a specific knowledge map that is independent of any application logic. To achieve this goal, we have had several rounds of discussions, both in college and in the company, by meeting many different people, who are familiar with the fields such as data mining or pattern oriented software designs.
- b) The second important step is how to find the EBTs and BOs for those patterns. Each pattern will study different scenarios to find the stable core logic, expressed as the EBTs and BOs.
- c) The third task is to put all these patterns together to develop a stable engine.

- d) The last and final step is the verification of the knowledge map, which is, writing different applications in different domains by using the same engine to prove the stability of the DM engine.

Figure-4 represents the lifecycle of the proposed data mining engine.

Figure 4 - Data mining engine development methodology

The lifecycle of the proposed engine has three major phases such as, application development, engine configuration, and running the application for data mining purpose.

The development section has multiple phases with very specific tasks like:

Analysis phase – Analysis phase has the following tasks:

- a) The main task in this phase is to identify the goals and EBTs of the system.

This phase is critical, because this phase dictates other component that exists within the system. If EBTs are improper, then the application will lack functionality as needed or desired.

- b) The final task in the analysis phase is to develop analysis patterns based on the EBTs.

Design phase – Design phase has the following tasks:

- a) The main task in this phase is to identify the BOs and develop design patterns, which would describe the capability of the system.
- b) Once developers properly identify EBTs and BOs, the knowledge map is developed, which represents the heart or the core of the system. Knowledge map is the core stable knowledge about the system.
- c) Another important task in the design phase is to develop the data mining engine framework, which can use the knowledge map, to develop application based on group of patterns that are identifiable by the application requirements.

There can be several rounds of iterations between design phase and the analysis phase. It is definitely possible that, while identifying capabilities of the system, the designer will find new goals and objectives. Under such a case, the designer may need to go back to analysis phase, and include the EBT in the system.

Development phase – Development phase has the following assigned tasks:

- a) At the beginning of the development phase, application specific requirements are correctly gathered.
- b) Based on the requirement or specification, it is possible to identify IOs in a proper manner.
- c) The next step is to implement the patterns that developed in the analysis and the design phases, along with IOs specific to the target application.

Integration phase – Integration phase has the following assigned tasks:

- a) Integration phase connects several patterns to build a specific application, as per the requirement.
- b) The compiled code will help a developer to build an effective application.

Configuration phase – Configuration phase has the following assigned tasks:

- a) Configuration phase tests the applicability of the developed application with real data. If there is any disconnection between the requirement and analysis, this phase will catch those anomalies to provide more inputs to the analysis phase.

Run phase – Run phase has the following assigned tasks:

- a) The final task is to run the application itself. This task is the test process of previous work. As the core knowledge and domain specific requirement builds an application, decision-making capabilities are already taken care of. Any user should be able to run this task, without any help from data mining experts.

1.7 How research is conducted

It is possible to conduct the research in the following manner:

- a) As mentioned earlier, the scope of this thesis is enormous, and separating the core knowledge from the application logic, and then developing software stability pattern needs plenty of detailed iterations and discussion processes.

The author of this thesis always used to meet with the advisor every week, and sometimes twice a week, to discuss the design, and the authors went through a number of iterations on the pattern presented in this thesis, before compiling the final draft. The authors have spent almost 200 hours in discussions and iterations.

- b) Lots of research conducted in data mining helped us to understand different stages in data mining and classification techniques, as so precisely defined by industry experts.
- c) I have organized several group discussions with many friends to design and develop the knowledge map in data mining.

1.8 Thesis contributions

The contributions of this thesis are twofold: It provides a generic flexible stable framework for the data mining engine, and it provides an implementation of the stability framework, for a complex problem like data mining.

The contributions of this thesis are:

- a) This thesis has developed a system of patterns, which are suitable for developing a stable data mining engine, used by different data mining applications, as the main core logic.
- b) We have developed the knowledge map for the data mining engine, which is the tangible representation of relationships of knowledge related to data mining. The thesis also describes with illustrations, how patterns derived from knowledge map can be connected together to develop a meaningful application.
- c) The knowledge map also proposes a new classification system. Most of the present classification relies on the same process. The thesis has also shown that data mining is independent of process; some applications can have two process, while, some others can follow several repetitive process that are based on the application requirement. Classification based on knowledge map classifies any data mining application, based on its main goals and capabilities. Processes might change, new algorithms found, but knowledge map is quite stable. Therefore, the thesis claims that classification based on knowledge map is very stable and flexible.
- d) Chapter 3 describes the main goal of data mining engine, as a number of analysis patterns obtained, are directly related to the data mining engine. Short description is given for each of them in chapter three, and detailed descriptions for two analysis patterns are provided in the Appendix A and Appendix B. The chapter three also describes the capabilities of data mining

engine as a number of design patterns. Chapter 3 also provides short description for each of them in, while Appendix C provides detailed description for one specific pattern.

- e) The Chapter 4 has shown the architecture of the data mining engine, which shows how different design patterns and analysis patterns could define the core of the engine. In very simple words, the engine is a collection of design and analysis patterns. Engine provides the interaction among these patterns through the message bus. Application specific logic is applicable as Industrial Objects, which is always outside the engine.

1.9 Thesis layout

This thesis comprises nine chapters and four appendices. The organization of this thesis will be as follows:

Chapter 1: Introduction

Chapter 1 presents two ideas: data mining and software stability modeling, and then explains how to use a software stability model to develop a stable core engine, for use any data mining application, which is independent of any domain.

Chapter 2: Comparative study

This chapter carries a detailed comparison of different data mining tools that are available today. These chapters also review the strength and weaknesses of five very popular tools, to check how each one is the best for particular types of needs. Then, the chapter also

discusses about the problems that encountered by the available tools today, and the thesis proposes to solve that.

Chapter 3: DM engine basic blocks

This chapter provides an in-depth analysis of the basic building blocks; that is, what are the goals and capabilities of a data mining engine; and what consists of the knowledge map. All EBTs and BOs are identified in this chapter, along with short description of few important ones.

Chapter 4: The DM engine

This chapter describes existing classification of data mining engine, and proposes a new classification based on the software stability model. It also explains about how to apply SSM techniques to develop a data mining engine that are usable as a set of independent components, that are applicable to any data mining problem, without making any changes to the core engine.

Chapter 5: Case study

This chapter describes a detailed case study about how the DM engine developed in Chapter 3 and Chapter 4 can be used to develop a data mining application, in a specific context or domain.

Chapter 6: Analysis & major findings

This chapter describes major contributions and findings of the thesis.

Chapter 7: Future work & conclusions

This concluding chapter suggests specific areas for future work in the short and long term. It will also provide many inputs that highlight how a full feature engine is developed based on the design as proposed in this thesis.

Appendix A: The Classification stable analysis pattern

This section of the thesis describes the stable pattern analysis-classification in considerable detail and the Chapter 3 provides this pattern analysis.

Appendix B: The Visualization stable analysis pattern

This section of the thesis describes the stable pattern analysis-visualization in considerable detail and the Chapter 3 provides this pattern analysis.

Appendix C: The AnyPrecision stable design pattern

This section of the thesis describes the stable pattern analysis AnyPrecision in considerable detail and the Chapter 3 provides this pattern analysis.

Appendix D: Implementation & code

This section of the thesis has the code for a sample application based on the data mining engine described in the thesis.

Appendix E: Demo & instruction

This section of the thesis describes a demo application and the instructions to execute the application.

Chapter 2

Comparative study

2.1 Introduction

Data mining is very critical for organizations in order to arrive at an efficient decision-making process. Various data mining tools find meaningful patterns and shifting trends in the data, by analyzing a large amount of data. Right now, there are many commercial data mining tools available on the market that are specially designed to meet the decision-making requirements of the business organizations; but, choosing one among many available data mining software is actually a tough decision. Many tools are very expensive, while the others do not accurately mine the data, which makes it difficult to get the right kind of information for use in business decision-making process. Some well-known data mining tools may have very useful data mining and modeling capabilities, but they are very difficult to use. On the other hand, a few of the mining tools could be quite easy to handle, but they might not possess the required capabilities, to mine the available data as per the requirement and immediate needs.

Strangely, many commercial mining tools are not platform independent, and they cannot read data format from different platforms. Using such data mining tools is a time consuming process, as the data feed process to the tools has to be utterly clean and entirely complete, and it requires users to spend precious time in preparing data for modeling. Furthermore, they also need to fill every bit of missing data, because commercial data mining tools present on the market simply cannot critically analyze data

with some missing data; neither do they have the capability to interpret and analyze those missing pieces of data. As of now, there are not data mining tools that could be termed as the best, as each one of them possess its own advantages and severe limitations. A tool that is capable of fulfilling one particular requirement of the business organization may not simply have the real-time capability to meet the critical needs of some other institutions. In this thesis, we will attempt to review five popular data mining tools, by highlighting their strengths and weaknesses and by checking how each one of them works under some specific conditions.

Data mining is defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" as described by W. Frawley [B12] and "the science of extracting useful information from large data sets or databases" as defined by Hand et al [B18]. Data mining is all about finding hitherto unknown patterns in data. The ideal data mining engine should preferably be independent of the source of the data or the type of the data. It is possible to source the data from any business or biological sources, like DNA patterns. Data could be single dimensional or multidimensional; it can also be in tabular form, or also as an image. Possessing all these diverse data formats in its bowel, a data mining engine has the toughest job to perform, which is finding unknown relations or pattern in the existing data. Any given software works very well only with known things. Software can neither think nor predict; it can do or perform only, what is specifically required. In this case, the designer of the software tool does not know what to find, how to find, and where to find. It is here where the wonderful data mining engine comes to help, by rescuing and salvaging the user. The

main job of a data mining tool is to design and develop a set of proper rules or to define a set of known steps to help find the unknown.

Though the concept of data mining is universal, many well-known experts provide many different types of definitions. Gartner group has defined data mining, as a process of discovering meaningful new relations and patterns - “Data mining is the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies, as well as statistical and mathematical techniques” [B34].

According to Dr. Fayad, the data mining is an essential part of the knowledge discovery process:

Finding useful patterns in data is known by different names (including data mining) in different communities (e.g., knowledge extraction, information discovery, information harvesting, data archeology, and data pattern processing).” The term “data mining” is used most by statisticians, database researchers, and more recently by the MIS and business communities. Here we use the term “KDD” to refer to the overall process of discovering useful knowledge from data. Data mining is a particular step in this process—application of specific algorithms for extracting patterns (models) from data. The additional steps in the KDD process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining ensure that useful knowledge is derived from the data [B37].

With all these definitions, it is very safe to conclude that the data mining is a process of finding interesting and useful patterns. Special care should be give to the word “useful.” The usefulness of a pattern depends mainly on the specific business need of the domain, where the results end up in practice. Different data mining applications use the same set of data to find different patterns based on a number of business needs. For example, a Midwest grocery chain used the data mining capability of Oracle software

system, to probe and analyze the local buying patterns of consumers. The results discovered revealed some interesting patterns; when, men purchased diapers on Thursdays and Saturdays, they showed an inclination to buy beer. Further analysis also showed that these shoppers typically preferred Saturdays as their shopping day! However, they bought a few number of items on Thursdays. The retailer, thus, was empowered to draw a conclusion that customers purchased beer for consumption during the weekends. It was also possible for the grocery chain to use these newly discovered bits of information, in many several profitable ways to increase their revenue. For example, they were able to move the beer display very closer to the diaper display. In addition, they also decided that it is possible to earn more revenues by selling beer and diapers at their full prices on all Saturdays.

This chapter discusses the existing data mining tools: their merits, advantages, and disadvantages. Further, it will also debate about the architecture of the data mining engine, and later explain how the engine described in this thesis is much better than the other commercial data mining tools.

2.2 Why data mining?

The needs and requirements for data mining have grown up recently because of a wide variety of factors:

- a) The stupendous growth in data collection and data warehousing opens up the windows of huge opportunities to analyze data, which were hitherto very difficult to perform. For example, Wal-Mart always captures “point-of-sale”

transactions from over 2,900 stores in six different countries, and continuously transmits this critical data to its massive 7.5 terabyte data warehouse. During 1995, Wal-Mart computers processed more than 1 million complex data queries and instructions.

- b) The perceived easy access to the stored data over the internet makes it possible for data mining applications to gather and collect necessary data from different locations scattered around the world. For example, a data mining application for analyzing the buying pattern in a local store can access the available census data over the internet to find out the earning pattern in a particular demographic area.
- c) Globalization has resulted in an increased competition, which makes companies and businesses look for additional business advantages, just to increase their market share.
- d) Off-the-shelf data mining applications make it easier for companies to perform data mining, rather than developing an in-house product, which often takes lot of time to develop, and it also needs a lot of resources.
- e) The growth in computing power and parallel processing makes it feasible to analyze huge data in a short period.

2.3 Existing data mining tools and environment

Because of the popularity in the use of data mining products for different purposes, companies and businesses always provide new products to augment the

existing market share. Some of these products are very popular, while the rest just disappear from the market with in no time. Right now, there are many commercial data mining products available on the market.

Some of them are:

- a) Corpora
- b) Cubist
- c) Connexor
- d) Cypher
- e) Compare Suite
- f) DocMINER
- g) DolphinSearch
- h) DtSearch
- i) DS Dataset
- j) Excel
- k) Enkata
- l) Entrieva
- m) Files Search Assistant
- n) FreeText Software Technologies
- o) TextQuest
- p) TeSSI®
- q) TextPipe Pro
- r) TextQuest

- s) Intellexer
- t) Insightful Miner
- u) Inxight
- v) ISYS:desktop
- w) Klarity (part of Intology tools)
- x) Leximancer
- y) Lextek Onix Toolkit
- z) Lextek Profiling Engine
- aa) Megaputer Text Analyst
- bb) Monarch
- cc) Recommind MindServer
- dd) SAS Text Miner
- ee) SPSS LexiQuest
- ff) SPSS Clementine
- gg) SWAPit
- hh) Fraunhofer-FIT's Text
- ii) Readware
- jj) Quenza
- kk) VantagePoint
- ll) VisualText™

Apart from these paid software products, there are a number of freeware and shareware software products available as free downloads, and some of them are INTEXT, S-EM (Spy-EM), and Vivisimo/Clusty.

As per a recent popular poll conducted by Kdnuggets.com website, some of the most popular data mining products were SPSS Clementine, SAS, Salford CART/MARS, and Excel.

A 2006 poll conducted by the same web site reveals that CART/MARS/TreeNet, SPSS Clementine, SPSS, Excel, and KXEN are among the first of a few popular data mining tools. By comparing the 2004 and 2006 poll, it is obvious that SPSS Clementine is still the most popular data mining software; however, CART/MARS/TreeNet, which was number one software in 2006, missed out in its rankings and dropped out of the top three during 2004. As a result, data mining software market is fiercely competitive, because the existing firms always try to retain their market share with the launch of new products and by adding enhancements to existing products.

On the other hand, new entrants try to sneak into the market by introducing new and innovative product ideas; but, such products are always “application oriented” in their structure and they simply fail to provide practical solutions to the ever-changing business needs. All new firms further develop their existing products with the addition of new enhancements and product features.

2.3.1 SPSS Clementine

SPSS Clementine is typically an analytical software product, which helps users to obtain key details from unstructured data culled from different websites. It also assists many companies for obtaining different types of data from inventory, accounting, ERP, and manufacturing. It will also classify, cluster, analyze, forecast, and predict different patterns based on its findings. The platforms currently used by this software are Sun Solaris, HP/UX, Windows NT, and IBM AIX. All these products provide a good and feature rich graphical interface. Its highly innovative GUI approach makes it very easy for the users to follow the available functions to obtain the desired result. A flow of functions is generated by using the drag and drop feature of the software.

Some of the advantages of using SPSS Clementine are described bellow from the SPSS website [B23]:

- a) Improving customer relationship management: Clementine is a very effective tool for all web based companies and businesses. It will mine web data and helps web sites to provide more effective and meaningful services to its customers. A SPSS product called Customer Centric also uses Clementine, which is analytical customer relationship management software that increases customer turnover to the web site, and helps to retain customer for a long time. It also helps to identify and detect profitable customers through effective data mining, and retain them by providing a high quality service.
- b) Reducing risk and fraud: Organizations using Clementine can use the knowledge, to identify the potential risks and detect instances of fraud well in

advance, by recognizing the ensuing customer behavior. Understanding the customer behavior will also help organizations to make web site designs better, and provide online services in an efficient manner.

- c) Reducing operational costs: It can also efficiently manage and maintain the operational costs, as the new feature selection algorithm used in the software can easily recognize the most and the least relevant of all data attributes. This will ensure more organized and a guaranteed analysis procedure, by easily generating simple and intuitive models.

The known limitations of Clementine, according to its traditional users, are:

- a) Very little availability of statistical analysis capacity.
- b) Output are neither well described nor elaborated.
- c) One type of solution will not handle different kinds of situations.

2.3.2 KXEN

This is yet another popular data mining tool. It can fulfill common needs of the decision makers, and it allows them to make better business decisions. The main advantage of using this software is its interactive model, and a script specially generated to reiterate the model development.

Some of the reasons for using KXEN tool are [B21]:

- a) The algorithms are very accurate and powerful.
- b) Easy to use – based on existing situations, it can provide solutions very quickly and easily.

- c) The API is very simple and clean.
- d) This software is good in automatic variable selection and model building.

Limitations mentioned by its users are:

- a) It cannot incorporate models from other sources.
- b) While interacting with the software, the user may need to use an expression language a “create column” node.
- c) For coding complex problems, users are unable to use a scripting interface.

2.3.3 TreeNet

TreeNet (a brainchild of Salford Systems) provides a capability to deal with different types of confusing data sets. In TreeNet, structured data is obtainable by gradually analyzing the different layers of information. TreeNet architecture contain many small trees, which contribute minute adjustments to the overall model; hence the name TreeNet. Diverse financial services, like credit risk scoring, targeted marketing, fraud detection and document classification services use this special software.

According to Salford Systems, the major benefits of using TreeNet software are:

- a) It can differentiate real data structure, from highly inconsistent and unorganized data sets.
- b) It has the real capacity of handling the missing values, without resorting to any repairs.
- c) It can also deal with very large numbers of candidate predictors, without any problems, and can differentiate those that matter from the rest.

- d) This software can be rooted in different kinds of computing situations.

2.3.4 SAS data integration tool

Taking and adapting efficient and right business decisions will largely rely on the capacity of the organization to obtain the right type data. However, obtaining the right and reliable data can be overtly challenging and very costly, for most of the business enterprises. Getting the accurate information from largely unorganized and scattered data is a big problem for the companies and firms. Due to the presence of large number of platforms, it is also very difficult for the companies to recognize and detect the repetitive and inconsistency in the available data.

According to SAS, its enterprise intelligence platform provides an integrated platform that overcomes these challenges, and delivers enterprise intelligence, all the while extending the value of existing IT architectures described in SAS architecture [B33]. The modules and tools of SAS Data integration (which is also one of the SAS data mining tools) are an interactive data integration and process development environment, metadata management facilities, read/write capabilities to third-party database, network connectivity facilities, deployment and management facilities, and quality and cleansing capabilities.

Some of the advantages as stated by the SAS for its data integration tools are:

- a) Platform independent: One of the major advantages of using SAS Data

Integration is that it does not become exclusively dependent on SQL. It can read and mine all data formats from different available platforms. The large

number of built-in transformation functions can run on all major operating systems, such as MVS, UNIX, and Windows, which finally reduces the large cost involved in data transfer.

- b) High data performance: For the purpose of data connectivity, it provides superior performance capabilities for data connectivity, data quality, ETL, which are also extracting, transforming, and loading data, data migration, data synchronization, and data federation.
- c) Easy data integration: These tools have a “drag and drop wizard” facility that helps quicker formation of customizable data integration strategies, without needing to possess substantial programming knowledge. It also makes the maintenance of thousands of lines of custom code much easier.
- d) More control over data integration: SAS Data Integration provides an improved accountability and control, which makes managing data integration processes, tracking process changes and managing metadata much easier for the companies. Overall, it also helps in obtaining greater control over all features of the data integration process. A metadata help prevents time-consuming research to figure out the method of obtaining the data elements, and it also takes out any kind of assumptions that are involved in deriving data.

2.3.5 Oracle data mining tool

Data mining is possible within Oracle database, by using Oracle Data Mining (ODM) tool. This tool works on the principle of relational tables and it helps users

extract the required information, without transferring data to a standalone tools, or specialized analytical server, thereby reducing the overall cost and trouble for the user of the tool. Because of its highly sophisticated integrated architecture, ODM works more efficiently and helps in managing data with a fair degree of accuracy. It allows users to model data and mine in the available database, and later transfers the available results and methodologies into full-fledged application components, so that the business establishments could make sound business decisions. It further helps analysts to take full advantage of all aspects of Oracle technology stack as part of an application [B26].

Oracle data mining also supports many of the supervised and unsupervised functions. Supervised functions may include classification of the data based on the class to which the item belongs, and by recognizing the attributes, which are relevant in predicting the pattern and in finding out abnormal data. The unsupervised functions are figuring out the similar types of data and generating new features from the existing features.

2.4 Data mining process

Different vendors of data mining tools usually follow different processes. SAS divides the process into five main stages: sample, explain, manipulate, model, and assess. Insightful Miner has the process modeling functions combined in their user interface. In such a case, user can choose different processes needed for their problem-using tool. In earlier days, most of these models did not provide any feedback mechanisms. CRISP

process models created by Clementine, always try to address this underlying issue, by providing feedback mechanisms in the process loop.

2.5 Comparison of different approaches

It is very difficult and cumbersome to compare different data mining tools, because the usefulness of the tool always depends on the specific problem in hand. One particular tool may be very good in analyzing an online search criterion and at the same time, this particular tool may be less useful in analyzing a set of genetic data for biological research. The accuracy of a tool entirely depends on the algorithm used for the purpose, and in absence of the right algorithm, the final prediction from the tool could be entirely misleading. Thus, there is no such thing called the best tool. The next section of this thesis provides a comparison between the proposed engine and the existing tools.

2.6 Analysis

Robert A. Nisbet has composed a very good article, on choosing the right type of data mining suite for a particular task [B24]. According to him, choosing the right type of tool is not an easy task, there are absolutely no tools such as the best and most complete; each one has its own, well-defined advantages and disadvantages.

The quality of any tool depends on the following factors:

- a) Easy and flexible to use.
- b) Provides acceptable accuracy (even though, not the highest accuracy available).
- c) Ability for carrying out tasks those are more common.

This thesis discusses the quality of the unified data mining engine, based on the metric provided by Robert A. Nisbet, as described above.

- a) Ease of use: Most of the tools of the day, lack the ease and flexibility of use, which are two of the most important factors. One tool will try to handle and manage everything, thus the UI provided by tool, becomes so complex and difficult that it needs an expert to run that tool. At times, the cost of training and sourcing an expert is more expensive than the cost of the tool itself. The main objective of data mining engine is to develop an efficient core engine, by embedding application specific logic within the engine, as and when needed. Using this framework, the cost of developing application becomes minimal, and it is more often like assembling the entire set of components based on the specific requirement. The data mining expertise should be needed to run such an application, because they focus on just one single problem, and all details such as configuration of different algorithm, what kind of data needs to be collected, how to clean the data, and the type of algorithm to run should already be implemented in the application logic. Therefore, running this tool will be as simple as pressing some simple buttons.
- b) Accuracy: The thesis has succeeded in forcing the accuracy factor in the core itself. An assembler of the tool can specify accuracy level needed for the purpose based on the specific usage of the tool; then, all other components like algorithm and data collection will ensure a fair degree of accuracy. It is

also possible to monitor accuracy based on the needs and requirements of the applications.

- c) Tasks coverage: The UDME (Unified Data Mining Engine) also provides direct application abilities; therefore, it will perform all tasks that needed for a particular application.

Few other quality factors, which are important in the context of a data mining tool, are:

- a) Easy to add New functionality: In UDME, the application specific logic is entirely separate from the core logic, and thus adding new application specific functionality is very easy, without making any changes in the core logic.
However, the usage pattern of the tool never links to a particular quality, but this definitely helps the tool developer to add new functionality, which might be useful for customers.
- b) Easy to interface: The design is a system of independent patterns, developed by different vendors or service integrators.
- c) Long lifetime: The engine also provides stable core logic, which has a long lifetime, while the application logic is loosely connected, and such application logic can change over time.
- d) Numbers of algorithms supported: The engine can support any number of algorithms.
- e) Fewer resources: The proposed engine connects several patterns from the knowledge map. Depending on the application domain, the engine can use a

number of patterns, which are necessary, therefore, needs fewer resources compare to existing tools.

Table 1 - Comparison of existing data mining tools and UDME

Criteria	Tools					
	UDME	SPSS	KXEN	TreeNet	SPS	oracle
Ease of use	*****	***	***	***	***	***
Accuracy	*****	*****	***	***	***	*****
Tasks coverage	*****	*****	***	***	***	***
Easy to add new functionality	*****	***	***	***	***	***
Easy to interface	*****	***	***	***	***	***
Long lifetime	*****	***	***	***	***	***
Numbers of algorithms supported	*****	*****	*****	***	***	***
Fewer resources	*****	***	***	***	***	***

Figure 5 and Figure 6 provide simple illustrations, to show the difference between the traditional approach and the approach mentioned in this thesis.

Figure 5 - Traditional data mining tool

Figure 6 - Data mining application based on UDME

The Figure 5 shows a complex application that is taking care of all different problems under different domains, where each problem has its own set of data, algorithm, and configuration. To serve all these different applications, it has to consider all different combination, thus the application becomes complex and very difficult to maintain. On the other hand, the Figure 6 shows how one application can be developed to solve a specific problem, therefore, the application is very easy to use and maintain.

2.7 Conclusions

Data mining is a continuously evolving area; the scientists from all over the world are inventing everyday, new techniques and algorithms. New business use cases are making an appearance to find the usefulness of data mining. Large enterprise to small companies are already beginning to understand the need of data mining, and how data mining can help to increase customer satisfaction or profit. Several types of different data mining tools are available today, but most of them focus simply on running specific algorithms, and they are not easy to operate and hence needs lot of expertise and skill to run them. As more numbers of new features get into the tool everyday, it becomes more complex to understand, manage, and handle. The main reason is that most of these tools are monolithic applications, which simply try to do everything at the same time, and there is no separation or layer between domain specific logic and the core data mining concept. Right now, there is no real concept of data mining engine that exists even today. This

thesis takes the first sure step forward to solve this problem, by developing a core engine specifically designed and developed for data mining.

Almost all dicey problems are easy to solve by a “Divide and conquer” policy – which is breaking the larger problem into smaller chunks of independent problems. Such independent problems are very easy to solve both individually and separately. Data mining, in fact, is a big problem; it is difficult to design one single and complete tool, which will run in any data set. To find such an interesting pattern, it may be very easy to dream and visualize, but practically almost impossible to implement. Moreover, data mining is all about finding interesting and useful patterns, and not just any simple pattern. Without knowing the domain or the specific requirement, it is not possible to determine and decide if a pattern is interesting or not. The definition of data mining itself restricts and binds the developer to design a single data mining application, which is usable in any application scenario.

That’s the reason why, most of the present day data mining tools need the assistance of an expert to decide, which algorithm to run and which pattern is interesting or not. One way to solve this problem is to write an application based on the specific problem, which is again not possible, because of the cost of the development and maintenance involved in the exercise. The approaches here is separating the core logic of data mining from the application logic, and develop an engine that is based on the core logic, where application specific component can be easily plugged and embedded into. This process will lower the cost of application development, and will provide the flexibility of having specific application based on the particular problem requirements.

Chapter 3

DM engine basic blocks

3.1 Introduction

Traditional software development is a combination of specific application logic modules; once the requirements are gathered for the given application, the design and coding is later performed for that specific application. However, this model does not scale very well, for a company, which is developing hundreds of products. Often, resources used for developing the same set of codes duplicate themselves, by different developers at a different point in time and at different locations scattered across the world. It is very difficult to separate out a particular piece of functionality from the application, and later use it for different purposes, when the given application is tied with many requirements and specifications. Software stability model specifically focuses on developing software based on domain knowledge, rather than on specific applications.

It first establishes useful knowledge maps by using domain knowledge. Then, depending on the specific needs and requirements, it is quite easy to develop specific applications by joining multiple knowledge maps. Software stability model also focuses on developing independent software component based on the knowledge map in a particular domain. These components are independent to each other and can work with any remote component in any given domain. The job of application developer is to choose different components based on the requirement, and put them together. To design

the UDME, the research efforts described in the thesis have divided the development effort into several sections that deals specially with:

- a) Identify the goals or the core enduring knowledge.
- b) Identifying capabilities within the system.
- c) Developing the knowledge map for data mining engine.
- d) Providing development scenarios for the model.
- e) Identifying deployment patterns in the system.

Figure 7 provides an overview of the architecture of the data mining engine. The bottom layer in the Figure 7 consists of the goals of the systems, like discovery, tuning, and learning, which form the analysis patterns. The layer above it describes the capabilities of the system, which also help to achieve the goals mentioned in the bottom layer, and these are design patterns. Table 2 lists all the analysis and design patterns that are found for data mining engine.

Figure 7 - Data mining patterns

Table 2 - List of goals and capabilities for data mining engine

Goals/Capabilities	Class type	Description
Classification	EBT	Dividing a large group into small sub groups.
Analysis	EBT	Dividing whole into small parts, to understand each parts separately.
Mining	EBT	The process of digging useful things from the core.
Discovery	EBT	The critical act of finding the unknown.
Visualization	EBT	Ability to see the future prospects and opportunities.
Tuning	EBT	The process of introducing many different techniques to improve the functioning capacity.

AnyParty	BO	This class represents any person or group.
AnyMechanism	BO	AnyMechanism can range from mathematical algorithm, such as binary tree to any mechanical procedure.
AnyCriteria	BO	A standard, rule, or test, on which a judgment or decision can be based.
AnyCategory	BO	This class represents the category generated from the classification process. Entities are group into different categories based on common properties. For example, when a user is classified based on the age then young, middle age, old are the categories, which has the age range associated with each of them.
AnyType	BO	Type can be a particular characteristic, or set of characteristics.
AnyEntity	BO	An entity is something that has a distinct, separate existence.
AnyPresentation	BO	This class represents the image, chart, or animation generated from the Visualization process. This class can also be considered as the connector of the two ends of visualization, where one end creates the view and the other end understands the view.
AnyRequirement	BO	This class defines different requirements provided by the actors, which is by the other pattern like AnyMechanism to run proper algorithm.
AnyActor	BO	Actors are participants of communication process.
AnyMedia	BO	This class represents means of communication, as radio and television, newspapers, and magazines.
AnyMetric	BO	Metric is a system of parameters or ways of quantitative and periodic assessment of a process.
AnyData	BO	Unorganized facts those become meaningful, when organized properly.
AnyModel	BO	Representation of an object or system from a particular viewpoint.
AnyPrecision	BO	Measurement with certain degree of accuracy.

AnyCollection	BO	This class represents group of objects.
Any Condition	BO	This class represents a particular mode of being of a person or thing.
Any Pattern	BO	Pattern is a template or a model, which follows a set of rules that can be used to infer or generate things or part of things.

3.2 Goals (EBTs)

Goals are analysis patterns that are very specific to the system. Goals mainly deal with EBT, which represents the requirements or the problem space of the core knowledge of the system. A data mining engine can provide a number of analysis patterns. In the next section, we will provide short descriptions for each of these analysis patterns. Please refer to the Appendices section for detailed descriptions. Table 2 gives an overview of the main goals of the data mining engine.

3.2.1 Discovery analysis pattern

Discovery analysis pattern attempts to describe the act of discovery process and such patterns could be applied to any domain. The core logic behind any discovery process can be unfolded with this pattern.

Name: Discovery

Context: Discovery pattern is usable in any problem context, where finding new object or idea is in focus. It uses different factors like knowledge, classification, pattern etc.

There can be several entities like object, idea, actor associated with the discovery process.

Problem: The Discovery analysis pattern is applicable in many different contexts as well. It is very difficult and complex to design a generic model and to cover all these different

scenarios. Moreover, to model a pattern, which takes into consideration of all different kind of discovery processes, is a serious and dicey challenge.

Solution and participants: Figure 8 shows the class diagram for the Discovery pattern.

Figure 8 - Discovery analysis pattern

Source: Adapted from Fayad's pattern archive

3.2.2 Classification analysis pattern

Classification is placing an object or idea into a group based on common properties among the objects or ideas.

Name: Classification pattern

Context: It is possible to create classifications for objects or ideas and concepts. The minimum requirement is that, the participating members need to convert into a set, and they should have some good properties to compare. With this minimum requirement, Classification can be applied to a vast majority of problems, ranging from classifying some movies as action movies to classifying people based on their country of origin.

Problem: The main problem of modeling classification is trying to find a common ground to describe the property of an object or an idea. There can be different kind of objects with different type of properties; it is always very difficult to express them in generic terms. Stability model also helps to describe object and properties in generic form, that model the classification.

Solution and participants: Figure 9 shows the class diagram for the pattern.

Figure 9 - Classification pattern

Source: Adapted from Fayad's pattern archive

3.2.3 Visualization analysis pattern

Visualization is the essential process of presenting data in a special and an easy-to-understand form and relationship, which are not evident from the raw data.

Name: Visualization pattern

Context: It is possible to create visualization for objects or ideas and concepts. The main idea behind visualization is to make something that looks very simple and ordinary, whereas the internal representation could be a very complex one. Visualization can also convey the ideas or the findings in an easy to understand way.

Problem: One of the problems of modeling Visualization is to find a strong generic base to represent an idea, objects, or complex raw data. There can be different kinds of data, the relationship among them that can vary from one to another, and expressing them can be cumbersome in generic terms. Stability model also helps to describe object and properties in generic form, to model the goals and capabilities that lie behind Visualization.

Solution and participants: Figure 10 shows the class diagram for the pattern.

Figure 10 - Class diagram for the Visualization pattern

Source: Adapted from Fayad's pattern archive

3.3 Capabilities

Capabilities are essentially design patterns, which describe the properties of the system, with respect to the knowledge map. During the classification process, any identified patterns will exhibit knowledge map related properties of the system. The Table 2 lists all the design patterns that related to data mining goals.

3.3.1 AnyInteraction design pattern

AnyInteraction is a design pattern, which shows the framework for interaction between two or more parties.

Name: Interaction pattern

Interaction is a type of action that occurs between two or more objects, and has an effect upon one another, as a subsequent result.

Context: Interaction can happen between any two given objects; it can be a living being or machine. For example, when somebody drives a car, the driver interacts with the car through the steering wheel, and the clutch and brake. In a classroom, student interacts with teacher by asking question or giving response to teacher.

Problem: The diversity of interaction makes it very hard to implement a stable model. There is no definite way or mode, how one interacts with another one. The pattern or medium of interaction changes widely depending on the type or place. Traditional model does not provide any technique to model such diversity or complexity. The following section attempts to create a model for generic interaction, with a stability model that can be applied anywhere.

Solution and participants: Figure 11 shows the class diagram for the Interaction pattern.

Figure 11 - Class diagram of AnyInteraction

Source: Adapted from Fayad's pattern archive

3.3.2 AnyPrecision design pattern

This section provides a detailed description of the AnyPrecision design pattern in the context of data mining pattern language. The pattern AnyPrecision characterizes the degree of mutual agreement among a series of measurement or values.

Name: AnyPrecision pattern

Context: Precision is mainly associated with measurement, sometimes, also related to abstract things, by expressing ideas precisely and to provide precise answers to a question. In the context of measurement, the factor of precision could also be engineering in nature, which describes how precisely different objects are put together to build a structure, for example, designing different components to build a bridge. In the subject of data mining precision, it describes the accuracy of the available data and the accuracy of the analysis made.

Problem: Precision can also be associated with the measurement, answers, and many different things in different domains. The problem here is to describe a generic model that can work in all different domains, where the aspect of precision is of paramount importance.

Solution and participants: Figure 12 shows the class diagram for the pattern.

Figure 12 - Class diagram for the AnyPrecision pattern

Source: Adapted from Fayad's pattern archive

3.3.3 AnyModel design pattern

This section provides a detailed analysis of the AnyModel design pattern used in the context of data mining pattern language. The pattern AnyModel also describes the nature of models within the given application. The model is a pure representation of the data embedded within the application.

Name: AnyModel pattern

Context: A model is also a specific representation of the data. It can deal with many objects, such as machine, houses, and bridges, all of which have visible, physical existence. On the other hand, software modeling is the representation of application logic in a compact format.

Problem: This model is very common and frequent in our daily life. Sometimes, it is a derived form of physical objects, such as building a model of a house, or sometimes based on abstract ideas or thought. It can be in so many different forms that are very hard and tough to identify within a generic pattern. With stability design patterns, any model is a BO called AnyModel, which is the most generic pattern of any possible model.

Solution and participants: Figure 13 shows the class diagram for the pattern.

Figure 13- Class diagram for the AnyModel pattern

Source: Adapted from Fayad's pattern archive

3.3.4 AnyCriteria design pattern

This section provides a detailed description of the AnyCriteria design pattern used in the context of data mining pattern language. The pattern AnyCriteria describes the rules, standards and the other external factors under consideration, before carrying out a given task.

Name: AnyCriteria pattern

Context: Criteria can be in any form or manner, and it is always usable different domains and contexts. For example, a stock market analyst criterion can be as simple as finding a stock, which is performing exceedingly well. On the other hand, criteria for selecting an algorithm for neural network could be very complex.

Problem: It is also possible to use many criterions in different domains like finance, software, medical etc. All theses fields have their own distinctive way of identifying criteria and representing them. It is very difficult task to find a common ground for modeling all theses patterns. By using stability, it is quote easy to develop a AnyCriteria model, which can used with all different kinds of criteria and under different fields.

Solution and participants: Figure 14 shows the class diagram for the pattern.

Figure 14 -Class diagram for AnyCriteria pattern

Source: Adapted from Fayad's pattern archive

3.3.5 AnyData design pattern

This section provides a detailed description of the AnyData design pattern that is usable in the context of data mining pattern language. The pattern AnyData characterizes

the representation of information in such a way, which is usable by any mechanisms or processes, and in a structured form.

Name: AnyData pattern

Context: AnyData is also usable everywhere and in any field of study. Data represents information, which can be simply inputs, as well as outputs of any system processes.

Problem: In reality, data can be any sort of information, but its structure may vary from one context to the other. In the case of statistics or engineering, data is mostly in mathematical and analytical form, but in other areas, it is often in textual form. The variable nature of the representations makes it very difficult and cumbersome to design.

Solution and participants: Figure 15 shows the class diagram for the pattern.

Figure 15 - Class diagram for AnyData pattern

Source: Adapted from Fayad's pattern archive

3.4 What is software pattern?

In software engineering areas, a pattern has two specific purposes; it clearly defines a problem, which can occur in several applications, and then it proposes a generic solution to the said problem. In object-oriented programs like C++ or Java, a pattern is executable, as the contract among several classes. To design the basic building blocks of the DM engine, the thesis initially found out several stable design patterns, and such stable analysis patterns entirely based on the methodology as defined by software stability model. Figure 16, Figure 17, Figure 18, and Figure 19 describes how different applications could arise from several independent patterns.

3.5 What is knowledge map?

Knowledge map is the tangible representation of different relationships of knowledge. The proper definition of knowledge is yet to be well defined. Different people have different way of explaining and defining it. According to most people, three important criteria need critical consideration, before any information or detail referred to as knowledge: justified, true, and believed; but, these criteria by themselves are not well defined. The base of justification can vary in nature; some people might believe in something, which can be wrong to others. It is important to have a proper definition, which people can follow and understand that will also have the same meaning to everyone. Therefore, the thesis tries to follow a simple definition that is easy to understand and follow. According to Wikipedia:

Knowledge is information of which a person, organization, or other entity is aware. Knowledge is gained either by experience, learning, and perception or through association and reasoning. The term knowledge is also used to mean the confident understanding of a subject, potentially with the ability to use it for a specific purpose.

In the context of software stability, knowledge is well-known understanding of any context, objects, event, or information. The thesis focuses more on “what and how,” rather than the “why” aspect. The thesis also tries to distinguish and differentiate knowledge in three different categories: knowledge, which are stable over time, knowledge of business specific logic, and the application specific knowledge.

Knowledge can be of different types, depending on the type of pattern used to express the knowledge. SSM uses two different types of pattern like, analysis patterns called goals, design pattern called capabilities. Every piece of knowledge is expressed as patterns, which is then expanded in three different layers of classes based on software stability model, which are Enduring Business Themes (EBTs), Business Object (BOs), and Industrial Objects (IOs). For gaining a better picture of this, please refer to Figure 7.

The main goal is to find out the relationships or interaction among knowledge in different domains, or in the same domain. This thesis refers this relationship as a knowledge map.

Figure 16 shows the overall picture of any knowledge map. Knowledge map is defining the relationships among several patterns from different knowledge domains. Here are few characteristics of knowledge map described by Dr. Fayad [B1]:

- a) Knowledge map is quite independent of specific applications or requirements.

It relies on the enduring knowledge of the system, via the use of stable

patterns for a set of recurrent problems. It is possible to create a large number of applications by using the same knowledge map and by reducing the cost of development, which ultimately means a faster ROI.

- b) Knowledge map also describes the knowledge as four main concerns: analysis concerns called goals, design concerns called capabilities, development concerns called development scenarios, and deployment concerns also called by the same name. Later, each piece of knowledge is expanded into three layers of concerns in software stability concepts', which are Enduring Business Themes (EBTs), Business Object (BOs), and Industrial Objects (IOs). The paper "Stable Pattern Languages: Knowledge Map Introduction" discusses more about knowledge map in detail [B31].
- c) Knowledge map also shows the relationship among several independent patterns within a single domain or in remote domains. This means that a knowledge map provides a better understanding of patterns in a specific domain, by describing the interaction and dependency among them, and it identifies the traceability of knowledge over several existing domains.
- d) It is also possible to develop several applications by using the constraints or relationships as defined by the knowledge map. Knowledge map also describes the knowledge as a set of patterns. Each pattern has its own set of constraints and well-defined interfaces, which are usable by the application for implementing interaction among several knowledge domain, and without exposing application specific logic.

- e) Knowledge map does not contain patterns; it defines the relationships among several patterns. In other words, a knowledge map does not own the patterns it describes the relationship among patterns over several domains.

Figure 16 - Knowledge map

Figure 16-19 shows how the same knowledge map helps us to develop different applications. Figure 17 shows several patterns sourced from different domains. The rectangle box represents the domain and the small circle inside the box represents patterns. Each pattern is consists of EBTs and BOs. The line joining two circles shows the live interaction between two patterns.

Figure 17 - Application 1

Figure 17 shows how different patterns can be connected together to develop an application. Red circles are the patterns, which take an active part in the application. Similarly, in Figure 18, the image shows how different pattern combinations develop a different set if application. In Figure 18, blue circles forms a completely different application, when compared to the one mentioned in Figure 19.

Figure 18 - Application 2

This illustration conclusively proves that the patterns developed on a stability model, can easily identify core patterns in any domains that are independent of any application logic. Depending on the requirements and the needs, it is also possible to develop different sets of applications by connecting these patterns. Figure 19 shows an example of an application using data mining patterns.

Figure 19 - Data mining application

3.6 Implementation issues

The main implementation issues that are related to the implementation of EBT and BO are discussed below.

- a) Separation of concerns: Software stability will separate out the patterns into three different layers as EBTs, BOs, and IOs, dealt in individual cases, with explicit dependencies among themselves. EBTs are implemented as interfaces,

while BOs are partly interfaces and partly business logic, whereas, IOs are the concrete classes, which implement the application specific logic.

- b) Hooks: Hooks are the extension points on the BOs that are easily adaptable by IOs. It is also possible to implement hooks in many ways based on the capability or ability provided by the language. In general, hooks describe a structured and an uniform method to allow the application developer to determine, methods to use the capabilities that are provided by the pattern
- c) Identifying data collection techniques: The data collection techniques are not part of the core data mining engine, which needs consideration during the implementation of any application. Application designers should find out the limitation, advantage, and disadvantages of the available data source, and then should choose the best scheme suitable for the specific problem.

3.7 Conclusions

The main objective of the this chapter is to describe, how the building blocks of a unified data mining engine could be developed, based on the software stability model, which focuses specifically on developing software based on core knowledge, rather than on specific applications. This chapter also explains how the SSM focuses on developing independent software component based on the knowledge map in a particular domain. This thesis has explained in detail the way of designing data mining (DM) engine, by dividing the development efforts into several sections that specially deals with identify

the goals, capabilities, developing the knowledge map for data mining engine, providing development scenarios and identifying deployment.

Further, a short template explains the analysis pattern and its goals such as Discovery analysis pattern, Classification analysis pattern, and Visualization analysis pattern. This chapter also discusses many design patterns and capabilities in considerable detail. The chapter also lists all the design patterns related to data mining goals, while problems and their corresponding solutions for each of these patterns also provide a valuable insight. To clearly understand and comprehend the topic, the concepts like software pattern and knowledge map provide information that is more detailed.

Chapter 4

The DM engine

4.1 Introduction

Extracting and culling out meaningful information from data warehouse has become extremely important for all business organizations and other institutions to ensure effective decision-making. Different companies have designed and developed different kinds of data mining tools for obtaining meaningful information and relationships among various data that were previously unknown to the decision makers. However, the biggest problem is in choosing the right one that would best serve the business needs. Some tools available on the market are very expensive, because of their huge development cost. Many of these tools are platform dependent, thus does not have the capability to interpret data from other platforms.

Moreover, the existing tools are very difficult to use and employ, as they do not analyze data with all the missing fields. Therefore, this chapter discusses in detail, how to develop a data mining engine by using software stability concepts. The DM engine described in this thesis is very stable that would not require any upgrade overtime, thus reducing the cost of development, also making it available to the users at much lesser price. The engine would also make it possible for the developers to share codes, which would subsequently reduce repeated tasks of adding codes based on specific requirements. Moreover, complexity and maintenance cost of the software will be almost zero, as it is not necessary to include new codes to the existing application in order to fulfill different tasks.

Software stability concepts have recently demonstrated a new way of developing software applications, which are stable over time and does not need any upgrade over time. On the other hand, other software development processes often start with a very narrow insight and a focus that solves a complex problem, based on the present requirement; that is the reason why, they start with requirement analysis and high-level design of the system. In this process, the design and implementation aspect binds and couples with the requirements, based on the present needs for the specific context, and in a very limited domain. Thus, the resultant output is neither stable nor can be applied for different applications. Designers often solve same problem repeatedly. Few issues with the present software development are listed below:

- a) Present software development starts with the requirement gathering and performs a design based on the given requirement. In spite of developers using several helper libraries during the software implementation, most of the works are specific to the target application.
- b) Over time, existing applications will include new and advanced codes, thereby adding more complexity and maintenance cost to the software.
- c) Different applications cannot share the code related to the same business logic.
- d) A code written based specifically on the requirement always changes constantly, so there is no stability in the design.

Software stability concept always attempts to address these issues, by providing a solid framework for developing software entirely based on the stability of the system. It also introduces three different layers or classes in the software development, using

object-oriented languages like java or C++. These essential layers are EBT, which is the most stable part of the system, which defines the core behavior or goals; then BO, which encapsulate the business logic and these can change internally, but fixed externally; next IO, which are industrial objects that are specific to the application.

EBT and BO are the stable part of the system, where IO can change over a period. In the stability model, IO is the only component, which is based on the requirements or application, whereas, EBT and BO are developed based on the knowledge map. We strongly believe that any knowledge is stable, and what changes, is how to relate them or use them. A knowledge map is the sole basis for developing stable analysis patterns or stable design patterns. It is always possible to mix such patterns to design and develop several new applications.

The next section provides details about how one can apply these concepts to develop a data mining engine, developed as a set of independent components that are applicable to any data mining problem, without making any changes in the core engine.

4.2 The data mining engine overview

This section of the thesis attempts to develop the building blocks of the data mining engine. It first starts with the classification, and after that, it discusses about different components that form the core of the architecture. Available classifications grouped into several categories, are common in a market place and some examples are:

- a) Classification based on processes
- b) Classification based on techniques

- c) Classification based on domains
- d) Classification based on solutions

4.2.1 Classification based on processes

The most popular classification relies on the CRISP model. CRISP-DM (Cross Industry Standard Process for Data Mining), was founded in 1997, and funded by the European Commission. The main idea was to define an industry standard for data mining.

4.2.1.1 CRISP

CRISP model describes data mining as a hierarchical process, by defining tasks in four different levels. A number of phases make up for the top level. Each phase in the top level has second level generic tasks that are associated with it. The second level tasks are always stable and flexible, which should be able to incorporate any future changes, and cover the core processes of data mining for any data mining applications. The third level has tasks, which describe, in what way the generic tasks needs to perform in certain specific situations. For example, if the second level has a task called “clean data,” then the third level will have other tasks like clean numerical value or clean categorical value. The fourth level records the decision and result of the analysis. “CRISP-DM 1.0 - Step-by-step data mining guide” discusses about the CRISP model in detail [B5].

All these phases, tasks represent a sequence of events, performed in different order and repeatedly, if required. CRISP classifies any data mining process into six different phases:

- a) Business understanding: In this phase, data mining engine understands the project objectives and requirements in the business context, and then transforming that knowledge into data mining problem definition.
- b) Data understanding: The data-understanding phase involves data collection and first insight to data.
- c) Data preparation: In data preparation, phase data is prepared for the modeling tools, which includes attribute selection, transforming and cleaning of the data.
- d) Modeling: The modeling phase will apply various modeling techniques and specifies the requirement for different techniques.
- e) Evaluation: This phase will evaluate the model, chosen earlier, and review all the steps executed previously to construct the model.
- f) Deployment: In this stage, the customers place the model into action.

4.2.1.2 Six Sigma

Another popular definition of data mining process is from a popular data modeling methodology called “Six Sigma.” This methodology defines a sequence of steps called DMAIC:

Define → Measure → Analyze → Improve → control

4.2.1.3 SEMMA

There is another approach, that is similar to this one is proposed by SAS institute called SEMMA.

Sample → Explore → Modify → Model → Assess

As CRISP is the first real approach to standardize the data mining process, it does a very good job, by clearly defining each steps of the process. This task-based approach helps data analysts to divide the complex process into several measurable small tasks. Some of the negative side or shortcomings of process-based classification are listed below.

- a) Process-based-classification does not talk about how a specific application hooks itself with those generic processes.
- b) To keep everything flexible, it does not enforce any rules in the process; therefore, it is very easy to commit any mistake.
- c) This classification does not draw any line in between, what can be implemented in software and what can be judged by the analyst.

4.2.2 Classification based on techniques

So many data mining experts attempt to classify data mining, based on different techniques to solve a given problem. According to Berry, Michael:

Data mining comes in two different flavors— directed and undirected. Directed data mining attempts to explain or categorize some particular target fields, such as income or response. Undirected data mining attempts to find patterns or similarities among groups of records, without the use of a particular target field or collection of predefined [B3].

Classification based on techniques does not really resonate with the main goal of data mining, which is all about finding a meaningful pattern. Techniques are the tools to achieve that particular goal. If there is any system designed based on this particular

classification, they will not be flexible to accommodate future needs. The fragile nature of techniques will also make the system very weak.

4.2.3 Classification based on domains

Sometimes, data mining definitions also rely on the available domains. One such approach is in the book, *Web Data Mining and Applications in Business Intelligence and Counter-Terrorism*, written by Thuraisingham, Bhavani [B35]. According to him, the key technology behind data mining is the database management and data warehousing. Apart from this, there are several other technologies like statistical method and machine learning. All these technologies are integrated quite well, to provide an effective data mining solution. For example, machine learning is mainly about learning rules and patterns that are analyzing the data. Now, to machine learn, one needs special statistical methods to find the pattern, and there are other technologies like visualization, parallel processing, while decision support is used to execute the task for reporting purpose. In this way, the mode of classification shows the manner, in which data mining is growing and how it is incorporating other important fields of study. However, it does not provide the true pictures of the data mining engine, its goal, its purpose and ways to achieve them.

4.2.4 Classification based on outcomes

In many cases, data mining process has been classified based on the outcomes i.e. classification, sequence detection, deviation analysis, and dependency analysis.

- a) Classification: An input data integrates itself well into many meaningful subclasses in this process by depending on some commonly occurring properties. For example, an auto manufacturing company has a set of data that shows most of the buyers of model X, is in age group 20-25. From this data, they can assume that the age group falling within 20-25, likes model X, more than the other models, even if they are not in their list.
- b) Sequence detection: In this process, the sequence is determined, by observing the pattern in the data. An example of sequence detection: Tom goes to a bank, before going to the bar.
- c) Deviation analysis: A third outcome of mining is deviation analysis. For example, Tom goes to a bank on weekdays, but does not go to bar, instead, he goes to office. With this type of mining, anomalous instances and discrepancies are easy to find.
- d) Dependency analysis: A fourth outcome of data mining is data dependency analysis. In this process, factors like dependencies, relationships, or associations between different data items provide invaluable information to form meaningful analyses. For example, if mortgage rate is high, houses sold is very low.

4.3 Data mining engine architecture

The classification mentioned in this thesis takes a different approach than any of the existing classifications. Existing classification rely on tools, techniques, methodology domains, which can evolve, or can change; therefore, the classification based on those ideas will also change over a period. The use of data mining is always changing because of what tools used and where they are used. Therefore, to design a stable data mining engine, this thesis focuses on ‘what’, rather than ‘how’ and ‘where’.

This thesis also tries to answer the question “What is data mining?” What is the core idea behind data mining? What is the tangible part in data mining? Is it possible to draw a line between core data mining logic and application logic? Data mining is the collaboration among multiple knowledge maps, as shown in Figure 20. The application is by connecting several patterns from different domains, depending on the application specific requirements. In that way, applications become very specific to the existing need. They do not contain unnecessary and unwanted codes, rather they only contain codes, needed to perform the jobs as described by the requirements, therefore making them lighter and easy to maintain.

Figure 20 - Knowledge map

In the heart and core of the data mining engine, there are basic core knowledge bases like data, analysis, reporting, discovery; statistics aspects tie close together in the context of data mining. There can be several remote knowledge maps based on applications or domains, where the processes of data mining are applied. Each knowledge map has several patterns. It is easy to develop applications by connecting these patterns. This way of classification covers all aspect of data mining, while providing the most stable architecture. There are several reasons, why the architecture mentioned in this thesis is stable, because:

- a) This classification does not dictate techniques that are applicable to the data, which can change over time.
- b) The engine couples very loosely with the domains, in the form of remote knowledge map; therefore, applying this engine in a new domain would mean connecting the core pattern with a remote knowledge map.
- c) This classification does not impose any sequence or order, because those are essentially application specific and tends to change all the time.
- d) This classification does not define the process explicitly, but use them as patterns hidden or closed under the knowledge map. This provides much deeper control rather than defining some superficial process.
- e) The application specific logic is not part of the engine, as that logic is applicable as Industrial Object, which is outside the core engine. Thus, it draws a clear line between enduring business logic and application logic.

- f) The most important part of this classification is that, one can implement and validate the classification during implementation stage itself.

The process based on this methodology can be very simple or sometime very complex as well, based on the type of application process. The engine defines the relationships among different pattern in the knowledge map in a precise manner, but the application always manipulates the sequence and its order.

Figure 21 shows a process diagram for a simple basic data mining application. Whereas, Figure 22, shows a complex model that can support application of multiple modeling tools based on the result.

Figure 21 - Simple process diagram

Figure 22 - Complex process model

4.4 Describe elements of architecture

Figure 23 provides the architecture of the data mining engine, which shows different design and analysis patterns, which further defines the core of the engine. In simple words, the engine is a collection of design and analysis patterns. An engine provides the interaction among these patterns through the message bus. Application specific logic is easy to implement through the Industrial Object, which resides just outside the engine.

Figure 23 - Data mining patterns

Figure 24 shows how different patterns from different knowledge map talk to the engine through the message bus.

Figure 24 - Data mining engine architecture

The data mining engine drives the communication among different patterns. Figure 25 shows the messages between mining engine and other pattern through the message bus.

Figure 25 - Communication between engine and message bus

4.5 DM engine applications

Data mining is not just finding pattern, it is also finding meaningful patterns in the context. When a stockbroker uses data mining engine to find a pattern in stock market variation for oil stock, the main purpose is to gain enough money by predicting the future of the market. Here, the purpose of data mining engine is the future prediction and the input to data mining engine, which is the historical data of stock market. Now, the stockbroker needs to talk to the expert about what kind of other data he/she needs and the experts recommended to add data for oil production rate from different countries.

All data mining tools require an expert to suggest the type of data needed and type of algorithm to use. The data mining engine described in this thesis, overcomes this by including the expert logic into a number of applications specific IOs. Different applications for different purposes are easy to synthesize by using the same data mining engine, thus the thesis provides the groundwork for most generic and flexible engine for data mining.

As different applications need knowledge specific to the needs, a general application will not entirely serve them. The next section of this thesis will describe how patterns described elsewhere help us to create a simple application. The following steps need close observation, while developing applications by using the data mining engine:

- a) Step 1 - Define the goal of the application: In order to find the goal of the application simply answer the question. What is the real problem? What will be the suggested solution? The output of this step is the list of analysis and design patterns needed to solve the problem.
- b) Step 2 - Select the operation needed by the application: Choose the operation or mechanisms needed for the application. This step makes sure that the application does not contain any unnecessary codes.
- c) Step 3 - Developing IOs needed by the application: develop the IOs by using the hooks that are provided by the pattern. Decide what is best, whether to use inheritance or aggregation.

- d) Step 4 - Developing IOs as needed by the application: Build the IOs along with patterns that are provided by the engine. The output of this step is the final application.

4.5.1 Example application – Text mining for customer support email

The following paragraph describes comprehensively, how an application is possible to develop, by using several patterns from the data mining engine. This application analyzes the email received from customer, and tries to find out any similarity in the complain pattern by discovering any cluster, and then categorize them accordingly in a proper manner. Pattern matching techniques and lexical analyzer are the methods used to discover phrases and relationships in the e-mail texts. These methods will also show how efficiently that piece of knowledge defines the categories, used by the classification process to group emails, based on some common properties. Figure 26 describes the patterns used in this application, along with the IOs and their interactions among them.

Figure 26 - Class diagram for a text mining application used in CRM

4.6 The impact of DM engine

The impact of data mining engine is described below.

4.6.1 Data mining in business

Data mining is very useful for running businesses, from where it will obtain hidden patterns and visible trends out of a large set of collected data. Patterns and business strategies formed, by using various types of algorithms and mathematical & statistical techniques, are robust and stable in nature

Most of the banks use effective data mining techniques to detect credit card fraud in a better and effective way. Sophisticated data mining techniques help bank authorities to check out customer accounts and later establish meaningful patterns. If there is any visible deviation or an abnormal pattern, then the customers may need to verify their credentials and recent transactions.

Bank of America is currently using data mining techniques to price their financial products. They are known to use Discovery data mining software from HYPER parallel to find prospective customers, who are in the process of applying for the lending products. The bank determines the sequence of events, which will lead to purchasing, by putting in the profiling parameters to the software resulting in determining the final list of quality prospects. Later the concerned bank uses this list for direct marketing, which ultimately results in successfully selling of their financial products as mentioned by Van, Barry [B39].

Beside banks and other similar institutions, several other businesses are using the concept of data mining as well. Data mining also helps companies and firms to figure out the customers' buying patterns, their tendencies, and the prospective lead. Many companies and firms also use data mining to choose customers for direct mailings or

giving shelf-space to a particular type of product, at the particular place. They also use a DM process to determine the marginal customers and frame policies to attract those customers.

4.6.2 Data mining in education

Data mining could be very useful and handy in predicting, which students would chose to go for higher studies, and those who would need financial assistance for pursuing higher studies, or students, who would need special assistance to perform better in their studies. Several information and details like students' grades and report cards are stored in the college database; from here, educational institutions will be able to deduce a meaningful trend, by separating out all hidden patterns from the student database. Later, all derived patterns can accurately predict the students' behavior, and help institutions to allocate resources accordingly and attract more students to enroll for higher studies. The DM engine described in this thesis could easily develop such applications that could determine the students' behavior, without many modifications or changes to the core application.

An efficient data mining engine will help academicians and researchers to detect instances of plagiarism. An anti-plagiarism web site called Turnitin.com is a web-based plagiarism determination application, which uses data mining to index and store a large collection of papers electronically submitted to them. It uses many data mining algorithms to develop a technology called "document source analysis" that makes a digital "fingerprint" of any text document, and then compares it against millions of other

papers existing on the Internet, which will ultimately help to find the occurrences of plagiarism in the submitted papers.

4.6.3 Data mining in medicine

Medicine is an exciting and thrilling area, where the principles of data mining has been successfully used and implemented. In a field called evidence-based medicine, a patient's critical medical data will be subjected to a through data analysis to perform advanced research in the field. The main aim of this exercise is to prevent any medical condition, before it occurs, or diagnose possible health problems and conditions by using the solid medical evidences. By using the process of data mining, some meaningful patterns could be determined in finding out certain diseases and conditions, which are yet link with certain symptoms.

To meet different requirements, different types of applications are required and developed urgently, as they base their processes on tools, techniques, methodology domains, which can evolve and change overtime and at any time; the aforementioned cases will testify to these facts. Therefore, to avoid the work and effort that is involved in developing different applications to meet different requirements, the DM engine is obviously the best choice, because core knowledge that is involved to design a stable data mining engine does not change overtime.

4.7 Conclusions

In the present chapter, the concept of software stability will provide a detailed explanation regarding the DM engine. Similarly, the architecture of the data mining engine will describe about itself in length, soon after explaining existing classifications based on processes, technique, domains, and solutions. This chapter also provides several advantages and disadvantages related to these classifications.

Under the section ‘classification based on process’, the CRISP model has been described in detail, along with its four different levels. Six different phases of CRISP, such as business understanding, data understanding, and data preparation, modeling, evaluation and deployment also provide a repository of information in a comprehensive manner. At the end of the chapter, a new system of stable classification will provide new insights, and this system relies on existing classifications that always change over time and that do not provide any stable way of a data mining classification.

To design a stable data mining engine, our thesis focuses mainly on the ‘what’ aspect, rather than ‘how’ and ‘where’. The thesis has also neatly classified data mining as a meaningful collaboration among multiple knowledge maps, like data, analysis, reporting, discovery and statistics, which are practically attached together in the overall context of data mining. The chapter also shows how the presented DM engine is different from any other data mining tools, which requires an experienced person to suggest the type of data needed, and also type of algorithm that should be used in the future. On the other hand, the presented data mining engine overcomes this lacuna or deficiency, by including the expert logic into applications. The thesis has also tried to

show how different types of application for different purposes can be developed, by using an efficiently endowed, generic, and flexible data mining engine.

Chapter 5

Case study

5.1 Introduction

Today, the sheer magnitude of available information can be overwhelming; therefore, it is almost difficult to retrieve any important piece of information in a quick manner and within a short time. As the price of memory modules are becoming cheaper day by day, companies and firms are storing all types of available data, no matter whether they are found useful or not. Through an internet facility, anybody can add any type of content, thus making it hard for a reader to find the correct materials. Text mining offers a great solution to this unique problem, by helping reader to find the proper and right content that he/she is looking for. It involves analyzing a large volume of data to discover information, which can be either relationship or patterns that lay buried in the pile of data, and would have been impossible to discover by ordinary human eyes. In this case-study, a text mining application is proposed based on the data mining engine that is specially presented in this thesis

Text mining is described by M. Hearst as “the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources” as [B19]. Text mining is quite different from web search. In a web search, the user already knows what to look for, and the search engine readily provides the materials, which are relevant with the search text, whereas, text mining is all about finding new and unknown patterns by relying on an automated process. Many people

view text mining quite differently from data mining, but text mining is a special form of data mining, which uses unstructured natural language text as the main input data.

The major challenge in text mining is the representation of the natural language used by human in a form understood very easily by the computer. For this purpose, different types of techniques are used. Any text mining process can be broken down to the four major steps as shown in Figure 27.

a) Information retrieval: Documents need correct identification during this phase.

This process helps to narrow down the input data set, by providing relevant data that is specific to the problem.

b) Natural language processing: This phase utilizes the grammar data provided by the language to classify word into different categories.

c) Information extraction: In this phase, key phrases and relationships are identified in the text, by using pattern matching techniques.

d) Reporting: final stage is present the result in a visual form.

Figure 27 - Major steps in text mining

Next section describes two scenarios in a text mining application by using Fayad's scenario template.

5.2 Scenario template

Scenario id: 1

Scenario name: Discovering categories

Robot/Tool names: NA

Description of the scenario: Before performing text mining, it is absolutely important to analyze the text to identify different terms such as names, numbers, abbreviations, currency dates etc, and then placing them into different categories. For example, name extraction is very important and it provides valuable clues to the content of a single document or a group of similar documents. The name extraction process also provides a dictionary of all occurrences of that name in the text and it further specify the type of that name, whether the name refers to human, place, organization etc. Often, different representations refer to the same name, for example, the terms “Albert Einstein,” “Einstein” “Great Scientist Einstein” always refer to the same person. The feature extraction process should be able to find such cases, where different names refer to the same person or object.

This scenario shows how key phrases and relationships are discovered in the text, by using pattern matching techniques and a lexical analyzer, and then how that knowledge can be used to define the categories that can be used by the classification process, to group document based on some common properties. Figure 28 describes the class diagram for this scenario.

Figure 28 - Participant classes in the scenario

Name all the IOs:

Feature

Document

Name

Term

Date

Number

LexicalAnalyzer

LanguageGrammer

FeatureDataSetCriteria

SpecificFacts

DocumentCluster

Science

TextMiningAgent

ClusteringAlgorithm

Additional hardware: NA

Required stable patterns:

AnyCriteria

AnyEntity

AnyMechanism

AnyType

AnyParty

AnyEvidence

AnyDiscovery

AnyCategory

Classification

Discovery

Challenge: The main challenge in this particular case is to relate the specific knowledge, discovered through the discovery process to the classification process. Another important challenge is also to determine the accuracy of the algorithm or mechanism used to identify any entity. It is always possible to identify date or numbers, but it is not possible for any system to identify these names with 100% accuracy all the time; but the system should always keep in track, the accuracy level that is associated with different entities, while using different mechanism.

5.2.1 Use case: Data extraction

To provide better clarity, use case along with the sequence diagram are provided, which shows how terms are extracted from a document to provide the base for the analysis.

Figure 29 displays a term extraction process:

.

Figure 29 - Term/Feature extraction process

Use case id: 1.0

Use case title: Extract terms from the document.

Actors	Roles
AnyParty	Site Admin
AnyParty	Language specialist

Class	Type	Attributes	Interfaces
Classification	EBT	title, description, type, mechanism	classify()
AnyParty	BO	name, responsibility	Initiate()
AnyMechanism	BO	name, type, criteria, category	run()
AnyCriteria	BO	name, description, condition, result	readCriteria() applyCriteria()
LanguageGrammer	IO	language, grammar rules	analyzeGrammer()
LexicalAnalyzer	IO	Language rules, language	analyze()
Term	IO	type, value	validate() findType()
Document	IO	name, type, category, words	readDocument() extractTerms()
Name	IO	Property	isName()

Enduring Business Themes (EBT): Classification

Business Objects (BO): AnyParty, AnyMechanism, AnyCriteria

Industrial Objects (IO): Name, Document, Term, LexicalAnalyzer, LanguageGrammer

Figure 30 describes the class diagram and Figure 31 describes the sequence diagram for this use case.

Figure 30 - Participant classes in the use case

Description of the use case:

- a) AnyParty<<BO>>, which can be a site Admin<<IO>> or Language Specialist<<IO>>, collects a document, and initiate the analysis by defining the Term<<IO>> into proper AnyCategory<<BO>> for Classification<<EBT>> purpose.
- b) AnyParty<<BO>> requests Classification<<EBT>> to classify each Term<<IO>> based on LanguageGrammer<<IO>>.
- c) Classification<<EBT>> runs AnyMechanism<<BO>>.
- d) AnyMechanism<<BO>> read the criteria from AnyCriteria<<BO>>.
- e) AnyCriteria<<BO>> retrieve the criteria mention in the LanguageGrammer<<IO>>.
- f) AnyMechanism<<BO>> extract the terms from the Document<<IO>>.
- g) AnyMechanism<<BO>> pass the Term<<IO>> and LanguageGrammer<<IO>> criteria to LexicalAnalyzer<<IO>>.
- h) LexicalAnalyzer<<IO>> check the Term<<IO>> for each category defined such as Name<<IO>>. If a match is found, it assigns the category to the term<<IO>>.

Sequence diagram:

Figure 31 - Scenario diagram

5.2.2 Use case: Discovery of features and clusters

Clustering process divides multiple documents into several groups. The document in each group shares some common properties. A detailed analysis of the text in the given document will help the user to find some common properties. One good approach is to find the list of terms, which are common in the documents.

An example of clustering in text mining would be to analyzing customer complains to find the similar complains, and later group them based on number of complains; this process can help customer support to prioritize the list of complains to find out which problem is really effecting the customers.

To provide better clarity, use case along with the sequence diagram are provided, which shows how terms are extracted from a document to provide the base for the analysis. The clustering process involved in text mining is shown in Figure 32.

Figure 32 - Clustering process

By using the IOs, it is possible to design an application. Figure 33 shows the class diagram and Figure 34 describes the sequence diagram.

Figure 33 - Class diagram

Use case id: 2.0

Use case title: Discover document clusters

Actors	Roles
AnyParty	Site administrator
AnyParty	Text mining agent

Class	Type	Attributes	Interfaces
Discovery	EBT	title, description, type, mechanism	classify()
AnyParty	BO	name, responsibility	initiate()
AnyMechanism	BO	name, type, evidence	run()
AnyEvidence	BO	name, description, condition, result	readCriteria() applyCriteria()
AnyDiscovery	BO	name, description, type, property	assignCategory()
AnyType	BO	Name, identifier, property	
SpecialFacts	IO	Name, type, property	matchWindSpeed()
ClusteringAlgorithm	IO	data, cluster, property	compare()
DocumentCluster	IO	name, type, cluster	recordWindSpeed()
TextMiningAgent	IO	name, type, document	collectData() assignCategory()
DocumentCategoryType	IO	name, type, property	

Enduring Business Themes (EBT): Discovery

Business Objects (BO): AnyEvidence, AnyParty, AnyMechanism, AnyDiscovery, AnyType

Industrial Objects (IO): SpecialFacts, ClusteringAlgorithm, DocumentCluster, TextMiningAgent, DocumentCategoryType

Steps of the Use case:

1. Discovery<<EBT>> starts the discovery process, by running AnyMechanism<<BO>>.
2. AnyMechanism<<BO>> notifies the AnyParty<<BO>>, which can be TextMiningAgent<<IO>>.
3. TextMiningAgent<<IO>> describes the DocumentCategoryType<<BO>> and then provides SpecialFacts<<BO>> about the Discovery<<EBT>>.
4. TextMiningAgent<<IO>> then runs ClusteringAlgorithm<<BO>> that creates the DocumentCluster<<IO>> and returns the result back AnyMechanism<<BO>>.
5. AnyMechanism<<BO>> provides AnyEvidence<<BO>>.
6. AnyEvidence<<BO>> uses SpecialFacts<<IO>> to describe the evidence of the Discovery<<EBT>>.
7. Discovery<<EBT>> then qualifies the AppDiscovery<<IO>>.

Sequence diagram:

Figure 34 - Sequence diagram

5.2.3 Use case: Document classification

A classification process can assign documents to a number of predefined categories. The main difference between classification and clustering is that categories are predefined with certain properties in classification process, whereas, categories and common properties are not known before hand in the case of clustering. Some very

common classification examples would be, how email filters act to store incoming emails into different folders based on some well-defined properties.

To provide better clarity, use case along with the sequence diagram are presented, which shows how terms are extracted from a document to provide the base for the analysis. The classification process involved in text mining is:

Figure 35 – The classification process in text mining

By using the IOs, it is possible to design an application. Figure 36 shows the class diagram and Figure 37 shows the sequence diagram.

Figure 36 - Class diagram

Use case id: 3.0

Use case title: Classify documents

Actors	Roles
AnyParty	Site administrator
AnyParty	Language specialist

Class	Type	Attributes	Interfaces
Classification	EBT	title, description, type, mechanism	classify()
AnyCategory	BO	name, description, type, property	assignCategory()
AnyCriteria	BO	name, description, condition, result	readCriteria() applyCriteria()
FeatureSetCriteria	IO	feature, category	create() validate()
TextMiningAgent	IO	name, responsibility	initiate()
DocumentClassifier	IO	name, type, criteria, category, document	run()
Document	IO	name, type, category	classify()
Science	IO	name, type, document	classify as()

Enduring Business Themes (EBT): Classification

Business Objects (BO): AnyCategory, AnyCategory, AnyCriteria

Industrial Objects (IO): FeatureSetCriteria, TextMiningAgent, Science,
DocumentClassifier

Description of the use case:

- a) AnyParty<<BO>>, which can be a TextMiningAgent <<IO>>, defines AnyCriteria<<BO>> such as FeatureSetCriteria<<IO>> for Classification<<EBT>> purpose.
- b) TextMiningAgent <<IO>> sends a request to Classification<<EBT>> to define AnyCategory<<BO>> based on the AnyCriteria<<BO>>.
- c) Classification<<EBT>> creates named category such as Science <<IO>> and define the classifier for the category.
- d) Once the categories are defined TextMiningAgent<<BO>> request for classification of new Document<<IO>> by running DocumentClassifier<<IO>>.
- e) DocumentClassifier<<IO>> validate the FeatureSetCriteria<<IO>> and then classify the Document<<IO>> as Scientist<<IO>> if it matches the AnyCriteria<<BO>>.
- f) Classification<<EBT>> assigns the proper category to the Document<<IO>>.

Sequence diagram:

Figure 37 - Sequence diagram

5.3 Conclusions

Text mining is all about obtaining precious knowledge from a large volume of text data and finding a number of hidden patterns in its bowel. It is also about analyzing text to discover hidden relationship from a large pile of data, which otherwise would have been impossible to identify with the human eyes. It also makes it possible for data mining algorithms to obtain meaningful information from completely unorganized text, this further helps data mining engine to classify data easily, for its better utilization in decision-making process. Previously unknown information now used by various fields, in developing good plans.

In the case study, a text mining application, specifically based on the data mining engine is proposed. This case study also provides many major steps of text mining and several process diagrams. In each of these steps, several use cases, as demonstrated by using stable design and analysis patterns, described in the earlier chapters in the context of the data mining engine.

The main purpose of this use case is to show, “how an efficient data mining application is developed, by using the core data mining engine that consists of stable patterns.” The use of the UDME is possible by developing the application on top of the exiting patterns, without having to develop the application from its scratch. This case study also demonstrates how the UDME assist us in developing the similar kind of data mining applications, in much lesser time thereby reducing the overall investment, time, and effort. The core engine also provides the core knowledge of data mining concepts

that gives opportunity to the application developers to concentrate on the application specific logic only, by saving their time, during design and implementation stages.

Chapter 6

Analysis & major findings

6.1 Introduction

The present research results display, how a system of patterns can help us to develop a stable and robust core engine for data mining. This thesis helps develop the knowledge map in data mining in a unique and special fashion, and further describes them as patterns, which not only helps developer to use them in any type of data mining application, but also provides a proper and thorough understanding of knowledge base, like classification, mining, and precision, all in the context of data mining. Furthermore, the patterns developed in this thesis, also describe how those patterns successfully extract and document the core knowledge of any system. It also deals with the aspect of the architecture related with the data mining engine that will also provide stability to various systems and applications in the near future. In the past, a detailed study conducted on the stability model provided a solid framework to the already finished work. The thesis further explains how EBTs and BOs can make the stability design pattern possible. Now, it has been explained that application IOs could be developed, by using the stable data mining engine in a much shorter period.

6.2 Major findings

The major findings of this thesis are:

- a) This thesis introduces a stable architecture for a data mining engine, which not only provides solid support for different systems and applications, it also provides unlimited stability over time. This goal can be achieved with the help of stable framework that accrue many novel concepts like EBT and BO, which are quite stable over time and which can be used by different applications as well.
- b) Our research expands on previous and ongoing work on software stability model. Software stability model is very novel and innovative idea in object-oriented design techniques. It differs from any other traditional models, by providing the stability in software design. With the help of these modeling techniques, the thesis has attempted to separate the enduring business themes in data mining, which are in fact, different analysis patterns that tie close together to provide the stable data mining engine. SSM also helps to find the business objects that reside inside the data mining engine, which later builds the core business logic. With the help of EBTs and BOs, the thesis designs an efficient data mining engine, which is stable and flexible.
- c) Instead of simply designing a data mining engine, which focuses mainly on a specific problem, the approach mentioned in this thesis, differs vastly from previous research in this area, by developing an engine, used in any data mining applications, more importantly as the core layer of logic. It hides all

the business logic, in the form of more generic EBTs and BOs, therefore, depending solely on the nature of the application, IOs can be developed, which can use this engine with the help of hooks provided by the engine; therefore, lowering the complexity and time for developing any data mining application from the ground level. With this new approach, it is possible to build applications in much lesser time. As time to market is the main key issue that contributes to the success with the data mining engine, companies can develop and market new applications very quickly and in this way, they will be able to capture the bulk of the market share. As time passes, market will also change, and with these shifting trends, the company can develop new IOs even with changing market trends.

- d) This thesis also presents a stable data mining engine, used by different data mining applications, by acting as the main core logic. Then, it develops different applications by using the data mining engine, which demonstrates the effectiveness of the approach mentioned in the thesis, in improving the speed of the development, and the quality of the software design modeling with respect to stability, reliability, management, and maintenance.
- e) We have developed the knowledge map for the data mining engine, which is the tangible representation of relationships of knowledge related to data mining. The thesis has also shown with graphical illustrations, how patterns from knowledge map could be connected together to develop an application.

- f) We have proposed a new classification, solely based on knowledge map.

Many of the present day classifications rely on this process. This thesis has shown that data mining is independent of process, and some application can have two different processes, while some other applications can follow several reparative process purely based on the application requirement. The classification technique based on knowledge map classifies any data mining application based on its goal and process capabilities. Process might change, and new algorithm found, but knowledge map is always stable. Therefore, the thesis claims that classification based on knowledge map is highly stable and flexible.

- g) Chapter 3 describes the goal of data mining engine, as a number of analysis patterns are possible by relating them to the data mining engine. Short descriptions for each of them, are in chapter 3, while detailed descriptions for two analysis patterns are in the Appendix A and Appendix B.
- h) Chapter 3 also describes the capabilities of data mining engine, as a number of design patterns. Short descriptions are given for each of them in chapter 3, while detailed descriptions for one design pattern, are provided in the Appendix C.
- i) Chapter 4 has shown the architecture of the data mining engine, which displays how different design pattern and analysis patterns define the core of the engine. In very simple words, the engine is a collection of design and analysis patterns. Engine provides the interaction among these patterns

through the help of a message bus. Application specific logic is possible through the Industrial Object, which is always outside the engine.

- j) Appendix A describes the Classification analysis pattern that models the core knowledge of classification in an effective way usable in many different applications, where classification concept is required, rather than repeatedly building the concept from a scratch. This pattern forms the core building blocks of the design of classification of all entities in any given domain, where application specific logic employs this core knowledge to implement domain specific requirements. Several scenarios have been explored and probed to show the applicability and reusability of the pattern.
- k) The thesis has also developed an analysis pattern for Visualization, mentioned in Appendix B. The visualization pattern can easily extract and document the core knowledge of visualization, as an analysis pattern planned to provide a recurring solution for the problem of visualization in any given domain or in any application. The pattern developed relies solely on software stability Model introduced by Dr. Fayad [B7]. Two case studies provided here, prove the applicability of the pattern.
- l) The thesis describes AnyPrecision design pattern that essentially models the core knowledge of precision in a domain independent way, and this pattern is usable in any application, where the degree of precision is quite important for the measured data. One can use this pattern as the core logic for precision in any application, rather than implementing it as a part of the application. The

“Software Stability Model” (SSM) [B7] can specifically identify and isolate the core knowledge of precision from the application specific knowledge.

Several scenarios also show the applicability and reusability of this pattern.

6.3 Conclusions

In this thesis, we provide a number of contributions in the data mining field by describing knowledge that is associated with data mining concepts, and how software patterns are used to describe the core concept behind data mining. The research further describes how data mining engine could be developed based on a given software stability model. Software stability model is a unique design technique for developing stable software patterns, which describe the core goal, abilities and capabilities of a system. Data mining is a generic concept not tied with any application and in any domain, but it is a challenging task to separate the core knowledge from application specific knowledge. The research work carried out in this thesis shows how software stability model can help in separating the core knowledge. This thesis also discusses different use cases to show and demonstrate how the applications developed based entirely on the patterns are stable over time.

Chapter 7

Future work & conclusions

7.1 Introduction

In this thesis, we present a unique model of a unified data mining engine for the benefit of readers, used in creating different applications like the core engine. Users can easily hook application logic, based on the requirements of the specific problem. Thus, the development of a large software application can be avoided, which tries to deal with all different possibilities, and takes into account several unrelated requirements, which can be developed as IOs and hooked into the core engine with least hassles. Since, this was not an easy task and a lot of case studies were needed to test the concept, the thesis focuses mainly on developing knowledge map, which can be used in the core engine and how they can be put together to build the engine. The next paragraph mentions possible future developmental work, which can take the UDME to the next higher level.

7.2 Future development

This thesis has also discussed few important designs and analysis patterns that are usable as the core data mining engine. To discuss all possible patterns and implementation of such an engine, it requires considerable analysis, which is out of the purview of this thesis. As the next essential step, following efforts are the possible future developmental face of the engine.

- a) The next logical step will be to move to a full-scale engine development, which may implement all the patterns that have been identified in this thesis, and put them together to make it work with several different applications under different domains.
- b) The engine should be tested and analyzed, considering all possible scenarios, to make sure that it covers all the interactions among the components of the data mining engine.
- c) The user interface of the engine should develop, by keeping in mind the usability and flexibility of adding different algorithms to the mechanism container.
- d) Users may need to implement and test all design and analysis patterns mentioned in this thesis, in a thorough and complete manner.
- e) The patterns mentioned in this thesis are usable in any application process, where the knowledge of the pattern is required. The way and the mode in which, each pattern developed was to keep them as independent modules, and this ensured that they did not tie with data mining and hence used in any other areas of study.
- f) As data mining is an ever-evolving subject, it is possible to incorporate several new ideas and technologies to the engine to consider all the necessary changes.

- g) It is also possible to develop stable patterns in the areas of animation and visualization; such patterns will assist in interpreting and comparing patterns found in the data mining process.
- h) Users can also develop an auto-selection process for selecting the right algorithms based on the domain and available data format for each applications.
- i) Unification of data is possible by developing patterns, within the available domains that ensure different kinds of operation on them.
- j) An integration engine, which considers the data format, domain, and application requirement, could be developed, as necessary inputs, and further automatically select the algorithm and convert the input data into the data format that is supported by the algorithm.
- k) Users can also develop a web-service based framework by using the engine, where customers could specify their requirements and the format of the available data. Now, the integration engine can select the appropriate algorithm for the specific problem, and provide efficient web services, to run on the customer data remotely.

7.3 Conclusions

This thesis introduces an idea of architecture for a data mining engine, used as the core of any data mining application, and it can provide stability over a period, by using a software stability model. This unique architecture delivers new concepts like EBT and

BO, which are stable over a period, that are reusable by different applications. The intended research also expands and builds on previous and ongoing research work on software stability model. In fact, this line differs from any other traditional model, by providing the stability in software design, and separating the enduring business themes and business objects, from the application specific logic.

Unlike existing data mining tools, the engine mentioned in this thesis, does not specifically focus on a specific problem. In the approach followed in the thesis, it develops an efficient engine based on concepts of core knowledge that is usable in any data mining applications. It also hides all the business logic in the form of more generic EBTs and BOs; therefore, depending on the nature of the application, IOs can be developed with the help of application hook, thus considerably lowering the complexity and time for developing any data mining application from the ground level. With this new and novel approach, it is quite easy to build applications in much lesser time.

To develop such an engine, a knowledge map is probably the best suited for a data mining system, which is the tangible representation of relationships of knowledge related to data mining. Several illustrations have been shown to prove how patterns from knowledge map, could be connected together to develop an application process.

This thesis also proposes a new classification system based on knowledge map, rather than the process itself. Data mining as a concept is entirely independent of process and the process changes over time, while the core knowledge remains the same. Thus, the classification based on knowledge map is stable over a period, and it classifies any data mining application based on its overall goal and capabilities.

As the scope of data mining engine is enormous, and as it is, out of the purview of this thesis, we have selected a few important patterns here just to show procedures of developing patterns, as a necessary part of the core engine.

The Appendix section also describes in considerable detail, the stable analysis patterns such as Classification, Visualization, and AnyPrecision, all of which models the core knowledge in a special way by using them in many different application areas; it is obvious that the idea of concept is required here, rather than repeatedly building the concept from scratch.

Briefly, this thesis has developed the architecture of a unified data mining engine specifically based on the usage of stability patterns. This thesis has also discussed the knowledge map used in the context of data mining, and later, it has developed the data mining engine entirely based on the knowledge map. Later, this thesis has described a number of patterns for the core of the engine. Finally, a case study will show how the engine is usable in a typical real problem scenario. In the conclusion, this thesis has designed, developed, and tested a unique core data mining engine that is usable as the core platform for developing any data mining application in the near future.

REFERENCES

- [B1] Mahdy, A. and Fayad, M.E., "A Software Stability Model Pattern." *ECOOP 2002, Workshop on Model-based Software Reuse*, Malaga, Spain, June 2002.
- [B2] "An Introduction to Data." Mining." <<http://www.thearling.com/text/dmwhite/dmwhite.htm>>.
- [B3] Berry, M. J., *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*, Hoboken, NJ, USA: John Wiley & Sons Incorporated, 2004, pp. 35.
- [B4] Cabena, P., Pablo H., Rolf S., Jaap V., and Alessandro Z., *Discovering Data Mining: From Concept to Implementation*, Upper Saddle River, NJ, Prentice Hall, 1998.
- [B5] "CRISP-DM 1.0 - Step-by-step data mining guide." <<http://www.crisp-dm.org/CRISPWP-0800.pdf>>.
- [B6] "Data Mining: What is Data Mining?" <<http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm>>.
- [B7] Fayad, M.E., and Altman, A., "Introduction to Software Stability," *Communications of the ACM*, Vol. 44. No. 9. Sept. 2001.
- [B8] Fayad, M.E., "Accomplishing Software Stability," *Communications of the ACM*. Vol. 45. No. 1. Jan. 2002, pp. 95-98.
- [B9] Fayad, M.E., "How to Deal with Software Stability," *Communications of the ACM* Vol. 45. No. 4. Apr. 2002, pp. 109-112.
- [B10] M.E. Fayad. Stable Analysis Patterns. Work in Progress.
- [B11] Fayyad, Usama, Gregory P., Shapiro, and Padhraic S., "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *COMMUNICATIONS OF THE ACM*. Vol. 39, No. 11. Nov. 1996.
- [B12] Frawley, W., Piatetsky-Shapiro G., and Matheus, C., "Knowledge Discovery in Databases: An Overview," *AI Magazine*, Fall 1992, pp. 213-28.

- [B13] Friendly, M., Statistical Consulting Service and Psychology Department, York University, Oct.1999. <<http://www.math.yorku.ca/SCS/Gallery/>>
- [B14] Hamza, H., "A Foundation For Building Stable Analysis Patterns," Master Thesis, University of Nebraska-Lincoln, 2002.
- [B15] Hamza, H., "Building Stable Analysis Patterns Using Software Stability." *4th European GCSE Young Researchers Workshop 2002 (GCSE/NoDE YRW 2002)*. Erfurt, Germany, Oct. 2002.
- [B16] Hamza, H., and Fayad, M.E., "Model-based Software Reuse Using Stable Analysis Patterns," *ECOOP 2002, Workshop on Model-based Software Reuse*, Malaga, Spain, June. 2002.
- [B17] Hamza, H., and Fayad, M.E., "A Pattern Language for Building Stable Analysis Patterns," *9th Conference on Pattern Language of Programs (PLoP 02)*, Illinois, USA, Sept. 2002.
- [B18] Hand, D., Mannila, H., and Smyth, P., *Principles of Data Mining*, Cambridge, MA, MIT Press, 2001.
- [B19] Hearst, M., "What is text mining?" (2004). <<http://www.sims.berkeley.edu/~hearst/textmining.html>>.
- [B20] *KDnuggets News*. <<http://www.kdnuggets.com/software/text.html>>.
- [B21] KXEN Inc. <<http://www.kxen.com/infocenter/integrators.php>>
- [B22] Larose, D. T., *Discovering Knowledge in Data: An Introduction to Data Mining*, Hoboken, NJ, USA: John Wiley & Sons Incorporated, 2005.
- [B23] "Maximize Your Returns with Data Mining and Predictive Analysis." Clementine. SPSS.com.<<http://www.spss.com/clementine/>>.
- [B24] Nisbet, Robert A., "How to Choose a Data Mining Suite," *DMReview published in DM Direct Special Report*, Mar. 23, 2004. <http://www.dmreview.com/article_sub.cfm?ArticleID=1000465>.
- [B25] Nisbet, Robert A., "Data Mining Tools: Which One is Best for CRM?" *DMDirect Special Report*, Jan.24, 2006.
- [B26] Oracle Data Mining, 10g release, B14339-01, *Oracle Data Mining Concepts*. Oracle, June 2005.

- [B27] Owen, Scott G., "Visualization Education in the USA," *Journal of Computers and Education*, 1993, Vol. 8, pp. 339-345.
- [B28] Ramaswamy S., et al., "Multiclass cancer diagnosis using tumor gene expression signatures." *Proc. Natl. Acad. Sci. USA*. 98(26):15149-15154,(2001).
- [B29] Salford Systems Inc, "TreeNet," <<http://www.salford-systems.com/doc/TreeNetTrifold.pdf>>
- [B30] Senkbeil, Jason C., and Scott Christopher Sheridan, "A Postlandfall Hurricane Classification System for the United States," Kent State University, Sep. 2006. <http://www.allenpress.com/pdf/coas_22_518_1025_1034.pdf>
- [B31] Sanchez, Huascar A., "Stable Pattern Languages: Knowledge Map Introduction," May 4, 2005.
- [B32] Sabasteanski, Anna, "Patterns of Global Terrorism 1985-2005: U.S. Department of State Reports with Supplementary Documents and Statistics," Berkshire Publishing Group, Sep. 2005.
- [B33] The SAS® Enterprise Intelligence Platform:SAS® Data Integration, "Three steps to faster ROI from your information assets," <http://www.sas.com/offices/europe/switzerland/pdf/solutions/EIP_Data_Integration.pdf>.
- [B34] The Gartner Group. <www.gartner.com>.
- [B35] Thuraisingham, Bhavani M., *Web Data Mining and Applications in Business Intelligence and Counter-Terrorist*, Boca Raton, FL, Auerbach Publishers, 2003, pp 43.
- [B36] Tkach, Daniel, "Text Mining Technology Turning Information Into Knowledge -A White Paper from IBM," IBM Software Solutions, Feb.17, 1998.
- [B37] Usama Fayyad, Gregory Piatetsky - Shapiro , and Padhraic Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *COMMUNICATIONS OF THE ACM*, Nov. 1996/Vol. 39, No. 11.
- [B38] Van, Jarke J, "The Value of Visualization," Mathematics and Computer Science Technische Universiteit Eindhoven. < <http://www.win.tue.nl/~vanwijk/vov.pdf>>
- [B39] Van, Barry. "Managing Tons of Data," *Computerworld*, April 23, 2001.

APPENDIXES

Appendix A: The Classification stable analysis pattern

“Science is the systematic classification of experience.” - George H. Lewes

A.1 Abstract

The main objective of this chapter is to extract and document the core knowledge of the classification technique, as an analysis pattern that is usable in many different applications, where classification concept is required, rather than repeatedly building the concept from its scratch. The motivation behind the development of this pattern is to write a meaningful generic pattern, based entirely on the core classification concept, which is applicable in any application and under in any domain, where classification is required. To achieve this goal, this thesis uses the concept of “Software Stability Model” (SSM) to identify and isolate the core knowledge of classification from the application specific knowledge. This thesis also explores and probes several different scenarios to show the applicability and reusability of the pattern. This section also provides a detailed documentation of the proposed pattern and demonstrates a number of benefits and advantages, by demonstrating two case studies, which describe two completely different applications that use classification to solve specific problem in a specific domain.

A.2 Introduction

Greek philosopher Aristotle developed the earliest known classification system, in 4th century BC (384 to 322 BC). He classified living organisms, as a plant or an animal, and animals into blood and bloodless. He further classified animals into different categories, like three groups, according to how they moved - walking, flying, or swimming (land, air, or water). Recently, the use of the classification in different problem has grown rapidly; for example, the diagnosis of complex genetic diseases like cancer relies on the tumor tissue type, pathological features, and clinical stages.

Right from the classification of animals and to the classification of cancer, classification has always found its usefulness and applications, in many different areas, and in different domains. Therefore, it is very important to understand the core knowledge of classification, without referring to any particulate domain or application, because the subject of classification is always evolving and changing, based on the new findings or technological advancement.

According to the classical view, in any given classification, categories are the discreet entities, characterized by a set of properties, which should be clearly defined and mutually exclusive. Therefore, any entity of the given classification should also belong to unequivocally to one, and only one, of the proposed categories. Eleanor Rosch and George Lakeoff have presented a different view of the classification, during the year 1970. According to them, categorization or classification are viewable, as the process of grouping and segregating things based on their prototypes - the idea of necessary and sufficient conditions presented in the classical view has almost never met in categories of

naturally occurring things. Although, there are several methodologies that have been used to describe how classification are supposed to be used, the core knowledge of classification still relies heavily on the fact that classification is a necessary procedure, in which individual entities are placed into groups based on one or more characteristics or properties inherent in the entities. It also relies on a list of items, previously categorized into categories based on properties of the defined categories.

Classification is the categorization of multiple entities into different smaller subsets, based on some commonly occurring properties. The main purpose of classification is to build a special model to classify the existing data into different categories, which could be further used to categorize unclassified data. For example, if the requirement of classification is to divide the customers into different age groups, then the first obvious step is to find the class variable, which can be in turn used to classify the data. In this example, the class variable can be “younger than 20”, “20 to 30”, “30 to 45”, “45 to 60”, and “over 60”. Once the class variables are well defined, then the available data can be easily grouped and segregated into different categories.

There are many different design techniques developed to model several classification problems; however, even though they try to model classifications, such efforts mainly focus on the domain knowledge. Therefore, the implementation varies from one another largely. This thesis also tries to overcome this problem, by separating the core knowledge from the application specific logic, into an analysis pattern by using the concept of “Software Stability Model” (SSM). In the remaining part of the paper, a

documentation of the classification pattern along with several typical scenarios will show the applicability and reusability of the pattern.

A.3 Pattern documentation

A.3.1 Name: Classification pattern

This pattern is required to model the core knowledge of any classification technique that is independent of the application specific logic; hence, the name is Classification.

Sometimes, the process of categorization is loosely referred to as “just classification,” though there is a lot of difference between the two concepts.

Categorization is all about defining categories of equivalence classes, such as sets of object, abstract things, processes, and a number of events treated as the same in the specific context. Whereas, the classification, in reality, is a two-steps process- first, a model of categorization is build based on pre-determined set of principles and secondly, the resulting model will classify entirely new sets of data.

The clustering always shares some form of similarities with the classification, but clustering is a type of unsupervised learning process, where class labels of the training data are entirely unknown, and clustering mainly focuses on finding existing clusters in the dataset. In that particular sense, classification is a type of supervised learning, where class labels are determined from the training set, which are applied to classify new data. Prediction has the same goal as that of classification, but prediction is frequently used to

predict missing or unknown data, whereas classification predicts categorical class labels, and it also classifies data based on the classification attributes. Other similar terms can be partition, division, and classification type etc.

A.3.3 Context

Classification is a generic process of classifying data into different categories. The core knowledge of the classification may never change, solely based on the data and its properties. This pattern has taken into account the generic concept of classification, which is grouping any data based on some common properties. The analysis pattern developed is applicable to any domain and any application. The application logic does not tie with the pattern; the pattern provides simple hooks, by which application logic can use the core knowledge of the classification.

Classification could come handy and useful in many different applications. For example, an insurance company can use the classification techniques to group different customers to determine the rate of premium. From the previously available accidental data, they can find a definite pattern, such as “singles have higher risk of accident, than those who are married,” or “rough drivers prefer sports car, whereas safe drivers prefer a family car.” All these different datasets are obtainable through a detailed survey, which come beneficial to classify any future customers. In the medical field, the risk of getting any particular diseases is deductible based on the age of the patients or their lifestyle status, such as the “smokers are prone to certain diseases, than nonsmokers.” Recent studies have also shown that the classification based on microarray gene expression, will assist in

diagnosing early stages of different types of cancers. This particular example shows the importance and life saving benefit of the classification in modern days.

Nowadays, in almost every field, the subject of the classification is readily applicable, either knowingly or unknowingly. The Classification analysis pattern provides a solid base to understand how the classification helps to understand the data in a better way, or how any type of application will benefit by classifying the input data.

A.3.4 Problem

There are couples of major problems that are associated with the present day classification model:

- a) **Flexibility:** Classification is usable in different domains, ranging from customer data to genetic information. All these applications use different training set, different properties, and different class variables. The existing classifications are limited to a specific problem; it is very hard to use just one classification for a different problem.
- b) **Evolving nature:** A classification is always evolving in nature; within the given context, a classification will always display greater capacity to grow and develop in the future. As new experiences are gathered, the pattern should also incorporate new changes easily.
- c) **Extensibility:** Any pattern on classification should provide novel ways, so that they are further extendable to provide the functionality needed for the specific domain.

- d) Accuracy: The main goal of the classification is to classify entities into different categories with a fair degree of accuracy, based on the criteria just defined before. When a criterion is ill defined and used, then the classification may produce an inaccurate result.
- e) Complete coverage: Classification can be of any entities based on similar properties. Any pattern on the classification should provide solution to all theses different entities having different characteristics.

Given the above highlighted issues, the problem here is, how to develop a model based on the core knowledge of classification, which are reusable by different applications, and in different domains and be able to extend as and when needed.

A.3.5 Constraints and challenges

The Classification pattern should resolve the following constrains and challenges:

Challenges:

- a) Classification is the categorization of any data or entity based on many common properties; therefore, any generic pattern on classification should be flexible enough to accommodate any entity or an object.
- b) Classification always depends on property; for example, classification of patients in a particular hospital relies on their age or on their health status. However, the design should not loosely couple with a specific property; rather it should allow any property to be the main criteria of the classification.

- c) There can be different constraints that are associated with a property, while classifying entities, for example, if the classification takes into consideration the factor of age, and then there should be an age constraint associated with the each category. The pattern should allow the presence of constraint on the property.
- d) The overall goal of the classification is to classify the unclassified data into different categories, based on the prior analysis of the data. There should not be any limit on the number of categories or the type of categories.
- e) Classification can also use different mechanisms to categorize the unclassified data into proper category by depending on the type of the application and the format of the data. The Classification pattern should allow different algorithms or mechanisms to work on the data in an effective way.

Constraints:

- a) The Classification patterns do not provide the actual mechanism or process, by which, one can perform classification processes; rather, it depends on the mode in which one performs the classification. Classification could be manual or automated performed through a number of computer programs.
- b) Each entity and categories should be associated with some types, which can be used to identify or differentiate an entity or the category.
- c) AnyEntity should have at least one class variable or property, which is usable as the classification variable, which differs from the one category to the other.

- d) It is meaningless to classify one or two entities; therefore, as the number of entities grows, classification will become more useful and beneficial. To keep that in mind, the pattern has forced a constraint that the classification should have at least three entities.
- e) Classification should have at least one mechanism or logical procedure, which defines the categories based on the training data.
- f) The mechanism used in classification should use some definite rules and must follow the constraint attached to the class variable.

A.3.6 Solution

The existing classification techniques do not talk or deliberate about the generic concept; rather, the classification techniques always tag to the specific application logic. This pattern tries to overcome that problem, by introducing a generic way of defining the core knowledge of the classification, based entirely on EBT and BO, which can be further reusable by different applications, as the core knowledge as shown in Figure 38.

Figure 38 - Class diagram of the Classification pattern

Source: Adapted from Fayad's pattern archive

A.3.7 Pattern structure and participants

The participants of the Classification pattern are:

Classes:

- a) **Classification:** This class totally encapsulates the core enduring knowledge of the Classification process. It also defines the goal of the classification system and ties all other components together.

Patterns:

- a) **AnyCategory:** This class represents the category, generated from the classification process. Based on some common properties, it is also possible to classify entities into different categories. For example, the status of a user may depend on the factor of age, where young, middle age, and old are the categories, which have the age range associated with each of them.
- b) **AnyProperty:** This class defines the common properties among the entities targeted for the purpose of classification. In the same example of classifying users based on the age, age will be the main property, which all users share and dividable based on the range of age.
- c) **AnyMechanism:** AnyMechanism represents the method used to classify the data. AnyMechanism can also range from mathematical algorithms, such as binary tree to any mechanical procedure.
- d) **AnyEntity:** This class represents entity or objects, which actually need classification. This can be human, any living being or objects depending on the context of the problem.

- e) AnyParty: This class represents the any person or group, who request the classification.
- f) AnyType: Classification is not only just classifying the data; moreover, the process should validate the correctness of the property used for entities. To achieve those objectives, entities need group themselves, based only on their type. AnyType used in this context can provide the flexibility of defining any type, which can be used to group entities into a category.
- g) AnyCriteria: This class represents the criteria of dividing entities into categories.

CRC- Cards:

Classification (Classification handler)		
Responsibility	Collaboration	
Represents core enduring knowledge of Classification process.	Client	Server
	AnyCategory AnyEntity	defineCategories() identifyClassVariable() classifyEntity()
Attributes: name, type, category		

AnyMechanism (Mechanism descriptor)		
Responsibility	Collaboration	
Represents the generic interface for different mechanism used to generate the view.	Client	Server
	Classification AnyCategory AnyRule AnyConstraint	run() stop() specifyConstraints() followsRule()
Attributes: name, type, context, media		

AnyProperty (Property descriptor)		
Responsibility	Collaboration	
Represents the common properties shared by the items in the same category.	Client	Server
	AnyConstraint AnyEntity	describeProperty() readProperty() attachConstraints()
Attributes: name, value, range		

AnyCategory (Classification descriptor)		
Responsibility	Collaboration	
Represents the category classified that is by the classification process.	Client	Server
	Classification AnyType AnyConstraint	defineProperty() createCategory() addEntity()
Attributes: name, list of entities, state, type		

AnyEntity (Classification inducer)		
Responsibility	Collaboration	
Describes the objects participated in the classification process.	Client	Server
	Classification AnyType AnyProperty	defineClassVariable() isTypeOf() createEntity()
Attributes: id, name, property		

AnyCriteria (Classification inducer)		
Responsibility	Collaboration	
Describes the constraints that are used to categorize the data,	Client	Server
	Classification AnyCategory AnyProperty	specifyRules() validate() apply()
Attributes: name, type, userid, password		

AnyType (Category descriptor)		
Responsibility	Collaboration	
Represents types that are used by entity or category.	Client	Server
	Classification AnyCategory AnyEntity	provideIdentification() identify() differentiate()
Attributes: name, type, capability		

A.3.8 Consequences

As the classification pattern solely rely on the software stability model, a number of benefits of SSM, such as scalability, flexibility, stability and testability will inherit automatically. This section has discussed some benefits that are very specific to the classification patterns, and many more things uncovered by the pattern.

Benefits:

- a) Adaptability in different domains: The classification pattern is always independent of domains. Many different applications can really extend the core knowledge of classification to fit the specific needs. For example, while classifying the different types of cancers, a technique known as gene expression analysis is used. Molecular structures of the genes will form the basis for number of category classifications. On the other hand, the same pattern will categorize books in the library, specifically based on author or subject.
- b) Extensibility: The classification can evolve over a period; therefore, the classification pattern provides the flexibility to change the attributes or

mechanisms used in the classification process. New mechanism, constraints, or rules merge easily to the exiting process, without changing the core logic.

Limitations:

- a) No application specific logic: The classification is just a conceptual model; yet, all the components used in this pattern, has clear names and functionality, which helps to visualize the pattern very easily, but the pattern does not provide any application specific context that can help us to understand the applicability of the pattern in a specific context. In one way, this is a severe limitation, but on the other hand, the goal of this pattern is to provide core knowledge of classification independent of any domain or application.
- b) No validation: In many types of applications, the classification is applicable along with validation or verification of the class variables or the properties. The validation is not included in the classification model, as it is a separate process, which is not a part of the core classification. This limitation is very easy to overcome, by connecting classification with a stable analysis pattern like Validation.

A.4 Applicability with illustrated examples

We have presented two case studies to display the application of the Classification pattern in different domains: One is the classification of natural disaster like hurricanes and typhoons while the other, is the classification of terrorist and disruptive activities. For simplicity, the thesis only focuses on the usage of the Classification pattern in the

problem space; therefore, the design does not include the other classes or component that is involved in the specific application.

A.4.1 Case study 1: Hurricane classification

Hurricanes are very dangerous and disastrous natural hazards for a country like, USA, which has a large coastal population. Every year, millions of people suffer from these natural disasters. Thus, it is very important to identify the hurricanes into different categories based on their devastating power and later warn people accordingly.

The most popular classification of hurricane is Saffir-Simpson scale (SSS), which classifies the hurricanes into five different categories based on the factors of wind speed. According to Jason Senkbeil and Scott Sheridan, the SS scale, however, does not accurately classify these storms; therefore, they have developed a post landfall, Hurricane classification system, which uses six different variables to categorize hurricanes :- open water storm surge, rainfall, duration of hurricane force winds, maximum sustained winds, gust score, and minimum central pressure. This thesis also uses both of these models to display, how the classification pattern can be used to develop a storm classification system very easily and effectively, which will be stable over a period of time, and can be used as the core logic right behind the application. Figure 39 describes the class diagram and Figure 40 describes the sequence diagram for this case study.

Figure 39 - Diagram of hurricane classification system based on SS scale

Use case: Hurricane classification

Use case description: To provide better clarity, a use case along with the sequence diagram, shows how a new hurricane event is classifiable into a specific category specifically based on the wind speed of different categories.

Use case id: 1.0

Use case title: classify new hurricane based on the wind speed.

Actors	Roles
AnyParty	Scientist
AnyParty	News reporter

Class	Type	Attributes	Interfaces
Classification	EBT	title, description, type, mechanism	classify()
AnyParty	BO	name, responsibility	initiate()
AnyMechanism	BO	name, type, criteria, category	run()
AnyCriteria	BO	name, description, condition, result	readCriteria() applyCriteria()
AnyCategory	BO	name, description, type, property	assignCategory()
WindSpeedCriteria	IO	windSpeedRange, category	matchWindSpeed()
WindSpeedComparison	IO	type, leftValue, rightValue, result	compare()
WindSpeed	IO	speed, time	recordWindSpeed()
Hurricane	IO	name, type, category, WindSpeed	collectData() assignCategory()

Enduring Business Themes (EBT): Classification

Business Objects (BO): AnyCategory, AnyParty, AnyMechanism, AnyCriteria

Industrial Objects (IO): Hurricane, WindSpeedCriteria, WindSpeed,

WindSpeedComparison

Description of the use case:

- a) AnyParty<<BO>>, which can be a WeatherScientist<<IO>> or NewsReporter<<IO>>, collects information about the properties of the Hurricane<<IO>> such as WindSpeed<<IO>> for Classification<<EBT>> purpose.
- b) Hurricane<<IO>> creates an instance of WindSpeed<<IO>> recording the data from a particular period.
- c) AnyParty<<BO>> requests Classification<<EBT>> to classify the Hurricane<<IO>> based on WindSpeed<<IO>>.
- d) Classification<<EBT>> runs AnyMechanism<<BO>>.
- e) AnyMechanism<<BO>> read the criteria from AnyCriteria<<BO>>.
- f) AnyCriteria<<BO>> creates an instance of WindSpeedCriteria<<IO>> and load the criteria defined based on WindSpeed<<IO>>.
- g) AnyMechanism<<BO>> runs the mechanism to compare wind speed, which is WindSpeedComparison<<IO>>.
- h) WindSpeedComparison<<IO>> applies criteria from AnyCriteria<<BO>> and then read the WindSpeed<<IO>> for the Hurricane<<IO>>. Based on the category defined in the matched criteria, it creates a proper instance of the AnyCategory<<BO>>.
- i) Classification<<EBT>> assigns the category to the Hurricane<<BO>>.

Sequence diagram:

Figure 40 - Sequence diagram for hurricane classification

A.4.2 Case study 2: Classification of terrorism

At different times and in different places, a terrorist may strike ordinary people to spread the ugly tentacles of terrors. It is important to classify different terrorist activities, which

can be helpful to relate an event to a particular terrorist group or to analyze a pattern, which can help to avoid any future terrorist activities. Figure 41 describes the class diagram and Figure 42 describes the sequence diagram for this case study.

Figure 41 - Class diagram of a terrorist classification system

Use case: Terrorist classification system

Use case description: This use cases shows, how different characteristics classify a terrorist group called AlQaeda.

Use case id: 1.0

Use case title: define categories for terrorist activities.

Actors	Roles
AnyParty	UN(United Nation)
AnyParty	FBI

Class	Type	Attributes	Interfaces
Classification	EBT	title, description, type	Categorize()
AnyParty	BO	name, responsibility	initiate()
AnyCriteria	BO	name, description, condition, result	defineCriteria()
AnyCategory	BO	name, description, type, characteristics	describeCategory()
GroupCharacteristics	IO	issues, Target	declareCharacteristics ()
AlQaeda	IO	type,leftValue, rightValue, result	assignCharacteristics ()
Issues	IO	name, description, type	addIssues()
Target	IO	name, type, description	matchTarget()

Enduring Business Themes (EBT): Classification

Business Objects (BO): AnyCategory, AnyParty, AnyMechanism, AnyCriteria

Industrial Objects (IO): GroupCharacteristics, AlQaeda, WindSpeed, Issues, Target

Description of the use case:

- a) AnyParty<<BO>>, which can be UnitedNation<<IO>> or FBI<<IO>>, initiates the categorization, by a Classification<<EBT>>.
- b) Classification<<EBT>> creates instance of AnyCategory<<BO>>.
- c) AnyCategory<<BO>> creates instance for a category called AlQaeda <<IO>>.
- d) Classification<<EBT>> describes AnyCriteria<<BO>>.
- e) AnyParty<<BO>> describes properties of AlQaeda <<IO>> by defining GroupCharacteristics<<IO>>.
- f) GroupCharacteristics<<IO>> assigns the Issues<<IO>> and Targets<<IO>> to of AlQaeda <<IO>>.

Sequence diagram:

Figure 42 - Sequence diagram of terrorist classification system

A.5 Design & implementation issues

The main implementation issues related to the implementation of EBTs and BOs are described below.

A.5.1 Separation of concern

The software stability separates the pattern into three different layers as EBTs, BOs, and IOs, all tackled individually, with explicit dependencies among them. EBTs could be implemented as interfaces, BOs are partly interfaces and partly business logic, whereas, IOs are the concrete classes, which implement the application specific logic.

A.5.2 Hooks

Hooks are extension points on the BOs that are adaptable by IOs. It is possible to implement hooks in different ways based on the ability or capability provided by the language. In general, hooks also describe a structured and uniform method to allow the application developer to determine how to use the capabilities provided by the pattern.

This section provides the description on various hooks, attached with this pattern.

- a) AnyParty: The involved party in the visualization process is extendable, by attaching an IO. For example, in the hurricane classification problem, a weather expert can be involved to categorize the available data, and a reporter can use that classification knowledge to classify a new hurricane. A user can gather all these different roles of the actors from AnyParty, by using a number of hooks.
- b) AnyCriteria: This BO can have a hook attached to it, to provide different criteria that can influence the mechanism used in the classification process. For example, in a hurricane classification process, the wind speed and duration are the main criteria while defining the categories.

- c) AnyCategory: This can have hooks attached to it to present the category, as defined by the classification. The type of category depends on several external factors based on the criteria; it is easy to replace or modify IOs, without affecting the structure of the overall system.
- d) AnyMechanism: Mechanisms works on the data very closely, as a specific data format will not be a part of the pattern hooks. An innovative mechanism can handle the data, especially for the application.
- e) AnyEntity: By attaching an IO, one can extend the already classified entries. . For example, in the hurricane classification system storm is the entity, which is already classified. All these entities are specific to the given application; therefore, they act as the hooks to AnyEntity, which describes the common behavior of all the entities, who participate in the visualization process.
- f) AnyType: Type relates to this pattern in several ways; for example, entities can have type, and category based on the type. Different type specific to the domains attached as hooks to AnyType, provide the core function of AnyType in the pattern.

A.5.3 Business issues

The main objective of classification is to a build a stable model to classify the exiting data into different categories, which could be further used to categorize all unclassified data. The major issues faced by most business today are how to classify the data with a fair degree of accuracy, and how to impose criteria with the proper

methodology to classify the exiting entities. The unified data mining engine provides an efficient way to describe how criteria and mechanism should be linked close together to provide an accurate solution to this problem. The Classification patterns used by a business organization look forward in finding any pattern from the existing data, which can help them define the criteria for classification.

A.6 Tips and heuristics

Finding the EBTs and BOs does not come immediately. The designer needs to understand the field completely in a comprehensive manner, and then should run many iterations before finalizing the EBTs and BOs, whereas identifying IOs is actually very easy; anything which is tangible can be replaced, changed over time, are IOs. One need to use the concept of inheritance with immense caution, if possible, using aggregation will provide a dynamic relationship, rather than inheriting everything from the base class. The link between parent and a child is often weak in the case of aggregation, whereas it is a strong link in the case of inheritance. For this reason, aggregation becomes very handy, while defining the hook to AnyMechanism, AnyEntity, AnyType, and AnyCategory.

A.7 Conclusions

Classification is more of a generic way of classifying data into many different categories. The important thing about classification is that its core knowledge will never change, based on the available data and properties. Many different design techniques presented in the near past, model a classification problem, but they all try to model

classification purely based on the domain knowledge; hence, it makes the implementation quite difficult for different domains. Therefore, the present chapter has addressed the analysis pattern of the classification that is usable in many different applications. All the inherent benefits and advantages of SSM, such as scalability, flexibility, stability and testability has been described in considerable detail, as the classification is always based on SSM. Through the enclosed case studies 1 & 2, we have made an effort to relate an event, by analyzing a pattern that could help us to reduce the bad impact of hurricane in a hurricane country like the USA, or could help to relate an event to a particular terrorist group, to avoid any future terrorist activities.

This thesis also discusses the merits of many challenges and provides a host of relevant solutions to a number of problems and constraints. The existing classification techniques do not deal with the generic concept, and they only represent specific application logic. Therefore, this thesis has also tried to address this unique problem, by introducing a generic way of defining the core knowledge of classification, based on EBT and BO, used by different applications as the core knowledge. Several classification classes and patterns like, AnyCategory, AnyProperty, AnyMechanism, AnyEntity, AnyType, and AnyConstraint will provide enhanced information and details on numerous classification patterns. The present chapter also displays how the Classification pattern will classify any object into a well-defined group based on some common properties.

Appendix B: The Visualization stable analysis pattern

“Visual Sciences is not a household name, but everyone in the web analytics business knows who they are. They have vaulted onto the scene and won the respect of everyone in our industry with the dynamic segmentation, multi-channel and real-time data visualization capabilities of their flagship Web analytics solution, Visual Site.” - Jeff Lunsford

B.1 Abstract

The main objective of this paper is to extract and document the core knowledge of the visualization, as an analysis pattern that provides a recurring and viable solution for the problem of the visualization in any domain, and in any of its several applications. Nowadays, people use the concept of visualization in every other field of study as an effective communication medium for an easy understanding of the subject. The usage of visualization in different domains or in different applications motivates us to design an efficient pattern shared by the application, as the core concept. The pattern presented in this paper relies on SSM, which also helps us to design a stable analysis pattern based on the core concept. The paper also provides a detailed documentation of the proposed pattern, and demonstrates the benefits of the pattern by using it, as the core logic in several different applications throughout several case studies.

B.2 Introduction

Visualization is the process of presenting information in an easy to understand form. In other words, this is a way of communication by creating images, diagrams, chart, or animation. Visualization has been used in maps, scientific drawings, and engineering for well over a thousand years. Some famous examples are the map of China (1137 AD.), and the famous map of Napoleon's invasion of Russia in 1812, by Jacques Minard. Recently, relevant information is available to enhance the Computer Graphics, by using the power of visualization. The recent emphasis on visualization started back in 1987, with the special issue of Computer Graphics, on visualization in Scientific Computing. Since then, there have been several conferences and workshops, co-sponsored by the IEEE and ACM SIGGRAPH, entirely devoted to the general topic, and special areas in the field, for example volume visualization. There have also been numerous books written and research articles composed on visualization in the past several years.

Modern technology such as internet, automated data loggers, and electronic devices have created a remarkable data explosion recently, thanks to our own human visual system, which helps us to understand this huge amount of data within a short time, though presented correctly. It is very important to represent the data or knowledge in a form, which can be understood by our brain easily and quickly; to cite a simple example, one picture is worth thousands words. That is the reason why visualization is so important and critical in a modern society.

Although the visualization is a collection of techniques and tools used to produce the visual representation of any entity, we cannot judge the effect of visualization just based on the results of those tools, without taking into account the context, where it is used and how it influences the scene. We have considered both of these aspects of visualization in our pattern.

Visualization plays an important role in the fields such as pattern recognition, animation, observation, learning, education, authoring, medicine, process improvements, modeling. Learning is easy through visualization; one can understand any difficult complex subject, if there is a visual representation. Nowadays, using animation techniques, visualization can help to create a real life simulation of abstract things like the solar system or the mode in which a black hole forms in the universe.

This paper intends to isolate the core knowledge of visualization concepts from the application specific logic, by presenting it as a stable analysis pattern, to provide the reusable core for other applications sharing the same core concept. In order to achieve this goal, the Visualization pattern relies on the software stability model. SSM provides a stable and *reusable* core for multiple applications sharing the same core knowledge. SSM also helps to find the goals of visualization and express them, as the Enduring Business Theme (EBT) and capabilities into Business Objects (BO). Fayad's pattern documentation template [B15] is the basis for the pattern documentation.

B.3 Pattern documentation

B.3.1 Name: Visualization pattern

Visualization works as a process of communicating information from one end to other. This process involves presenting the information in an easy to understand manner. Visualization is not just drawing pictures or graph; it is presenting information or thoughts in an effective way, by using different tools and technologies, such as computer animation, and chart. The pattern not only focuses on integrating different tools or mechanisms to provide a form, it also connects the participating parties, which either produce the view, or simply try to understand the view. As this pattern models the core knowledge of visualization integrating different entities with different mechanisms to provide one or multiple views, hence the name “Visualization” is chosen, rather than animation or imaging, because those terms focus on the specific mechanisms used in the visualization.

Few similar terms that carry almost same meaning as visualization is, picturization, mental imaging etc. Though picturization is sometimes used in place of visualization, it is a specific type of visualization; visualization is not limited to pictures, sometimes animation techniques can also be used as a visualization process. The goal is to develop a pattern, which is usable for different domains or applications; therefore, the term visualization, as it is more generic than picturization or imaging.

B.3.3 Context

Visualization is a changing process. It started, when cave dweller started painting on the wall; nowadays, we see the use of visualization in all fields of science, engineering, business, arts, education, movies etc. The computer graphics opened a new frontier for visualization, by providing new techniques such as animation. The visualization is both image creation and understanding of images. The actors can be humans or computers, who can help in creating the view, as well as the understanding of the view. The Visualization pattern addresses this diversity, by providing a stable analysis pattern, for the use in different contexts and for different applications.

To make the idea clear, the visualization demonstrations are usable in the classroom for the benefits of the students. In an engineering document, visualization will specify the guidelines and or specification of any structure, tools, or mechanism. In a data mining application, visualization generates a chart, graph, or animation, which shows the pattern in the data or helps predicting future occurrence of any event. In a war, visualization will document the strategy and the plan. We are all familiar with use of visualization in medical sector, where doctor monitors patient's vital signs, by using an automated chart. In simple words, there is no field or sectors, where visualization is never used. The vast number of applications adds a fair amount of complexity to design a core pattern, which extend to serve all these different applications.

In most applications, visualization is the channel, by which user of the application always interacts with the application. Visualization works as a process of communicating information from one end to other. This involves presenting the information in such a

way that a user can perceive it quite easily. This pattern discusses the core knowledge behind the visualization concept, and how application specific logic forms the top of core pattern, depending on the requirements or specific domain knowledge related to the specific application.

B.3.4 Problem

Major problems with present visualization model are:

- a) **Productivity:** The list of applications is infinite. This makes the job tougher to come up with a single design pattern, which could include all. Visualization is not only creating images or chart; it also includes the process of understating them. The problem is how we can describe the core knowledge behind the visualization process, which is always domain independent and usable in any context of visualization. Visualization is always evolving, as and when technology advances in the future. In early days, manual procedures produced drawings, but nowadays computers can generate drawings from huge data set, with in a second.
- b) **Maintainability:** Any pattern on visualization should provide ways to provide the functionality needed for the specific application.
- c) **Understandability:** Visualization is an important tool in learning and education. One of the important goals of visualization is to provide a better way to understand things. To keep that in mind, the pattern should provide a way to test and implement understandability of the visualization process.

- d) **Simplicity:** If something is simple, it is easy to understand. Visualization is different from Art. In Art, different people imagine different things by looking at the same picture, but visualization should provide a unique simple view to its viewer. Simplicity is the key to many visualization processes.
- e) **Observational ability:** The Visualization pattern should provide an uncanny ability to observe things. Without anybody observing, the concept of visualization has no meaning. Therefore, the pattern should provide a way to track the observer and their behavior.
- f) **Recognition ability:** The Visualization pattern should have the recognition ability. If the observer cannot recognize the view, the main of purpose of presenting anything with the help of visualization is almost defeated.
- g) **Documentation and recording:** The pattern should provide good documentation of the concept that can help designer to understand the core concept of visualization.

Given the above issues, the problem is how we can develop a model based on the core knowledge of visualization that is reusable by different applications and in different domains, and which is extendable, as and when needed.

B.3.5 Constraints and challenges

The Visualization pattern should resolve the following constraints and challenges:

Challenges:

- a) Visualization process involves the presentation of information in a clear and lucid graphical format. This information can be anything like data, concepts, entities, or ideas. The pattern on visualization should be flexible enough to consider any entity.
- b) Visualization can also be a chart representing a trend in the data, a process diagram, an engineering drawing of a building, or an animation of the growth of human brain. There is no boundary about what it can be; therefore, it is very difficult to design a model, which could work in all these different applications or situations.
- c) Visualization can generate multiple views from the same set of data, depending on the needs and techniques used to generate the view. For example, driving direction is in a map or written in a text format, where the reader visualizes the image, while reading the direction.
- d) The data used to generate the view in the visualization process can be in different forms i.e. numeric, symbolic, spatial, temporal, relational, discrete, and continuous etc. The pattern should be generic enough to accommodate any data type.
- e) Visualization can be an interactive process. For example, when somebody is looking at the map in Google, they can zoom to a particular position or might want to see, for example, where are the nearest restaurants. The model should accommodate user's active interaction.

- f) The Visualization pattern is a visualization engine that covers all the needs and purposes of each application, and/or all the applications in same time, by providing visualization as a separate tool to use for any application.
- g) Visualization can be a single layer or multiple layers of different natures.
- h) It is very easy to generate many applications with visualization techniques.

Constraints:

- a) The Visualization pattern does not provide the actual mechanism to generate a view. The application should extend the capability of AnyMechanism pattern used in the Visualization pattern to implement the actual technique in generating the view.
- b) The Visualization pattern does not specify any specific object, or properties, which can be a view, rather it has generic classes, like AnyEntity or AnyRequirement that can be extended to provide such detail related to a particular application.
- c) The Visualization pattern does not consider how users perceive the view or how they interact with the view. Rather, it depends on the application developer to implement IOs, which can develop such functionality based on the application logic.
- d) Visualization should have at least one entity to work.
- e) Visualization should use at least one mechanism to generate the view.
- f) The entities should have some properties, which generate the view, and the view should have some properties to uniquely identification.

- g) Visualization should use media to produce the view.
- h) There should be at least two actors, who participate in the visualization process: one creates the view, while the other perceives the view. If there is no viewer, there is no need of visualization.

B.3.6 Solution

As a solution, this thesis has developed an analysis pattern based on stability design concept i.e. AnyMechanism, AnyPresentation, AnyEntity etc. Stability patterns are generic and stable in nature. They are neither tied or nor hemmed to any specific domain, therefore, the pattern, based on such component are independent of nature and can be used in any application. The core concept of visualization is to provide the view of group of entities, based on certain characteristics with the help of certain mechanism. Visualization is untied with any specific mechanism or any specific entities. The Figure 43 shows how the core concept acts as an analysis pattern.

Figure 43 - Class diagram of the Visualization pattern

Source: Adapted from Fayad's pattern archive

Pattern structure and participants

The participants of the Interaction pattern are:

Classes:

- a) Visualization: This class encapsulates the core enduring knowledge of visualization process. It generates the view of several different entities, with different properties using different mechanisms.

Patterns:

- a) AnyMechanism: AnyMechanism represents the mechanism used to generate the view. In the case of two-dimensional view, it can be the classes used to draw a chart or bar diagram.
- b) AnyPresentation: This class represents the image, chart, or animation generated from the visualization process. This is the connector of the two ends of visualization, where one end creates the view while the other end understands the view.
- c) AnyRequirement: This class defines different requirements provided by the user, to generate the presentation, and generation the view. For example, in the case of a map, road is represented as line, whereas any point of attraction is shown as small circular box with different color. Therefore, the color, dimension, are the requirement to show different entities in the context of two-dimensional map.
- d) AnyEntity: This class represents entity or object shown in the view. For example, in a class diagram classes, connectors are the entities.

- e) AnyActor: Actors are participants of communication process. As mentioned earlier, visualization is for communicating information or concept. Whoever participates in creating the view and whoever tries to understand the view are the actors. Actors can be human or computer.
- f) AnyMedia: This class also represents the media from where one can generate the view. In the case of online map, internet is the media. Similarly, for animation movie, Television can be the media.

CRC- Cards:

Fayad, Hamza, and Sánchez. 2003 have proposed the CRC Cards used here in this thesis. Each card names the class, the role of the class, the responsibility of the class, and the collaboration; each name has two distinctive sections, the client and the server. Clients are the classes, which use the services provided by the named class, and server lists the services provided by the named class. CRC-cads for the Visualization pattern are listed below.

Visualization (Visualization handler)		
Responsibility	Collaboration	
Represents the visualization process	Client	Server
	AnyMechanism AnyPresentation	status() registerMechanism() meetRequirement()
Attributes: id, description, map (mechanism, view)		

AnyMechanism (Mechanism descriptor)		
Responsibility	Collaboration	
Represents the generic interface for different mechanism used to generate the view.	Client	Server
	Visualization AnyPresentation AnyRequirement AnyMedia	run() stop() resume() pause() useMedia()
Attributes: name, type, context, media		

AnyRequirement (Property descriptor)		
Responsibility	Collaboration	
Represents the property of the entities used in the view.	Client	Server
	AnyMechanism AnyMedia AnyEntity	nameRequirement () listRequirement ()
Attributes: name, value, range		

AnyPresentation (Visualization descriptor)		
Responsibility	Collaboration	
Represents the view generated by mechanism perceived by the user.	Client	Server
	Visualization AnyMechanism AnyEntity AnyActor	isAnimated() listLayers() addEntity() zoom()
Attributes: name,entity_list,state		

AnyEntity (Visualization inducer)		
Responsibility	Collaboration	
Describes the objects participated in the view.	Client	Server
	AnyPresentation AnyRequirement	provideData() updateProperty() isHuman()
Attributes: id,name,property, isHuman		

AnyActor (Visualization inducer)		
Responsibility	Collaboration	
Describes the users of the view	Client	Server
	AnyPresentation AnyMedia	listPreferences() notifyActor()
Attributes: name, type, userid, password		

AnyMedia (Media)		
Responsibility	Collaboration	
Represents the media used in the visualization process.	Client	Server
	AnyMechansim AnyActor	getCapability() interact() connect()
Attributes: name, type, capability		

B.3.7 Consequences

This section discusses some benefits and limitations of the proposed visualization pattern.

Benefits:

- Adaptability in different domains: The Visualization pattern is analysis design pattern, based on stability design model. The core knowledge of the visualization pattern is defined, based on EBTs and BOs whereas IOs can be added based on the application logic. The pattern is not dependent on any specific business logic; therefore, any application is the core model.
- Support different type of object: A visualization pattern is applicable to different entities. In a map application, the entities can be road, hospital,

restaurants, whereas in a medical application heartbeat, blood pressure can be the involved object, and the pattern to provide the view of any entity.

- c) Support different properties: The properties of the participating entity can be different based on application logic. In a map application, the road can be expressed as a line and the point of interest can be pictured as rectangular box of different dimensions, where as in medical monitoring system, the heart beat of a patient can be represent as a two dimensional graph.
- d) Support different media: Visualization process can be through the internet web page or through a TV depending on the availability of media or application type. The Visualization pattern is independent of any specific media type. The media is AnyMedia pattern, which extended by IOs, implements the application specific media.

Limitations:

- a) No application specific logic: Visualization is a conceptual model; yet, all the components used in this pattern has clear names and functionality, which helps to visualize the pattern easily, but the pattern does not provide any application specific context, which can help to understand the applicability of the pattern in a specific context. One-way, this is limitation, but on the other hand, the pattern provides a common core, used in different applications, as the specification logic is united with the core knowledge.
- b) No quality monitoring: As one judges the quality of visualization, based on a specific context, it is unsuitable to consider it as the core knowledge of

visualization. This limitation offset by attaching other stable analysis patterns, such as monitoring or recoding.

B.4 Applicability with illustrated examples

We present two case studies to show the application of the Visualization pattern in different domains:

Location based online services and a patient monitoring system. For simplicity, the usage on the Visualization pattern in the problem space is considered, therefore, the design does not include other classes or component involved in the specific application.

B.4.1 Case study 1 - Geographic Information Systems (GIS)

With the growth of web technology, online GIS systems are becoming part of our daily life. Many different applications are reaping the benefit of online visualization of GIS, such as real estate applications, where viewer can see the location of the property, without visiting the place in person. There are applications, where parents can track their children with a location based GIS application. The following class diagram in Figure 44 shows a location based GIS application developed by using the Visualization pattern as described earlier.

Figure 44 - Using Visualization analysis pattern in GIS

Use case: Visualization in GIS

Use case description: To provide clarity, a use case along with a sequence diagram are presented, which shows how user can find good restaurant, within five-mile radius from the location of the user, using a real time location aware GIS application.

Use case id: 1.0

Use case title: Find restaurants

Actors	Roles
AnyActor	User
AnyActor	MeetingOrganizer

Class	Type	Attributes	Interfaces
Visualization	EBT	title, description, mechanism	observe() operates()
AnyActor	BO	name, responsibility	initiate()
AnyMechanism	BO	name, type, requirements, presentation	run()
AnyRequirement	BO	name, description, priority, type, value	readRequirement () validateRequirement ()
Map	IO	windSpeedRange, category	draw() showResturants()
ServiceLocator	IO	name, description, list of services,	loacteResturants()
Restaurant	IO	name, description, location, food	isCloseBy()
MobileDevice	IO	isConnected, capability, connection	connect() findResturants()
LocationFinder	IO	GPSconnection, location	detectPresentLocation ()

Enduring Business Themes (EBT): Visualization

Business Objects (BO): AnyActor, AnyRequirement, AnyMechanism, AnyPresentation

Industrial Objects (IO): Map, Restaurant, MobileDevice, LocationFinder,
PresentLocation

Description of the use case:

- a) AnyActor <<BO>>, which can be any User<<IO>> or MeetingOrganizer<<IO>>, initiate the request to list all nearby Restaurants <<IO>> in a Map <<IO>> for Visualization<<EBT>> purpose.
- b) MobileDevice <<IO>> sends a request to LocationFinder <<IO>> to detect the present location of User<<IO>>.
- c) LocationFinder <<IO>> updates the PresentLocation<<IO>> of the MobileDevice<<IO>>.
- d) MobileDevice <<IO>> ad the PresentLocation<<IO>> in AnyRequirement<<BO>>.
- e) AnyActor <<BO>> send the request to Visualization<<EBT>> to view the nearby Restaurants<<IO>>.
- f) Visualization<<EBT>> runs AnyMechanism <<BO>> to generate the Map<<IO>>.
- g) AnyMechasnism<<BO>> reads all requirements from AnyRequirement <<IO>>.
- h) AnyMechasnism<<BO>> sends a request to ServiceLocation<<IO>> to find Location<<IO>> of all Restaurants<<IO>> near to PresentLocation<<IO>>.

- i) AnyMechanism<<BO>> produces Map<<IO>> to show all nearby Restaurants<<IO>>.
- j) Visualization<<EBT>> shows the Map<<IO>> to the User<<IO>> with all nearby Restaurants<<IO>>.

Sequence diagram:

Figure 45 describes the sequence diagram for this use case.

Figure 45 - Sequence diagram for finding nearest hospital

B.4.2 Case study 2: Visual data mining application in health care

The development of interactive intelligent data mining applications is getting popular in health care industries. Visualization is one of the key and hot concepts used in such application, where the health condition of a patient can be monitored by using a simple graph or a three dimensional view of the human body. Moreover, an application can find pattern and generate alert in a visual form. Figure 46 describes the class diagram for this case study.

Figure 46 - Using Visualization analysis pattern in health care application

Use case: Visualization in health care

This use case describes how patient's heartbeat is monitored by the doctors or nurses in a hospital using a chart.

Use case id: 1.0

Use case title: Chart patient's heartbeat

Actors	Roles
AnyActor	Patient
AnyActor	Doctor
AnyActor	Nurse

Class	Type	Attributes	Interfaces
Visualization	EBT	title, description, mechanism	observe() operates()
AnyMechanism	BO	name, type, requirements, presentation	run()
AnyActor	BO	name, responsibility	initiate() monitor()
MonitoringDevice	IO	id,company,description, type,capability,patientId	readPatientHeartbeat()
Patient	IO	name, age, address, phone, vitalsToBeMonitored	recorsHeartbeat()
Graphics	IO	id, type	updateData()
Chart	IO	id, title, legends, data	draw() update()

Enduring Business Themes (EBT): Visualization

Business Objects (BO): AnyActor, AnyMechanism

Industrial Objects (IO): MonitoringDevice, Patient, Graphics, Chart

Description of the use case:

- a) AnyActor <<BO>>, which can be any Nurse<<IO>> or Doctor<<IO>>, initiate the request to monitor the Heartbeat<<IO>> of any Patient <<IO>> in a Chart <<IO>> for Visualization<<EBT>> purpose.
- b) Visualization<<EBT>> runs AnyMechanism<<BO>> to generate the Chart<<IO>>.
- c) AnyMechanism<<BO>> sends a request to MonitoringDevice<<IO>> to read the Heartbeat<<IO>> of the Patient<<IO>>.
- d) MonitoringDevice<<IO>> reads the Heartbeat<<IO>> of the Patient<<IO>>.
- e) AnyMechanism<<BO>> update the data in Graphics<<IO>> that generates the Chart<<IO>>.
- f) AnyMechanism<<BO>> notify Visualization<<EBT>> about the Chart<<IO>>.
- g) Visualization<<EBT>> presents the chart to AnyActor<<BO>>.

Sequence diagram:

Figure 47 describes the sequence diagram for this use case

Figure 47 - Sequence diagram for monitoring heartbeat of a patient

B.5 Implementation issues

The main implementation issues related to the implementation of EBT and BO are:

B.5.1 Separation of concerns

Software stability separates the pattern into three different layers, as EBTs, BOs, and IOs, which tackled on an individual basis with explicit dependencies, will solve the concerns of separation. EBTs are interfaces; BOs are partly interfaces, partly business logic, whereas IOs are the concrete classes, which implement the application specific logic.

B.5.2 Hooks

Hooks are extension points, on which the BOs are adaptable by IOs. Hooks are set up in several different ways, based on the capabilities provided by the language. In general, hooks describe a structured and uniform method to allow the application developer to determine how to use the capabilities provided by the pattern.

This section provides the description on various hooks that relate to this pattern.

- a) AnyActor: The actor that participates in the visualization process is extendable, by attaching an IO. For example, the actor, which generates a graph for the visualization process, can be machine, which plots data in a piece of paper such as seismograph, or it is an operator or societies, who view that chart. For a corporate presentation, presenter can be an actor and whoever attend the meeting can be an actor as well.

- b) AnyRequirement: This BO can have a hook attached to it to enforce different requirements that can be associated with the mechanism used in the visualization process. For example, in visualization process the requirement can change based on the expectation of the viewer and on the intention of the presenter, while the pattern remains stable.
- c) AnyPresentation: This can have hooks attached to it to present the view, generated by the visualization process. Presentation depends on several external factor and capability of the devices used in the visualization process. These IOs are possible for replacement or modifications, without affecting the overall structure of the system.
- d) AnyMechanism: Mechanism works on the data very closely as a specific data format that is not part of the pattern hooks, or mechanism can be provided to handle the data, especially for the application.
- e) AnyEntity: By attaching an IO, the entities are always extendable. For example, GIS system roads, hospital, restaurants are the entities, which are under review. All these entities are specific to application; therefore, defined as the hooks to AnyEntity that describes the common behavior of all entities, who participate in the visualization process.
- f) AnyMedia: The media, on which visualization develops, can be different based on the capability of the devices and also the communication protocol presents between the presenter and the viewer for example when viewer watches TV then TV is the media but when the read newspaper works as the

media. The functionality and capabilities of such different media are usable, as hooks to AnyMedia. AnyMedia provides the generic interfaces that are usable in the visualization process.

B.5.3 Business issues

Visualization is often used as a communication medium in today's business. Starting from a marketing advertisement to a business plan, it solely depends upon the quality of visualization used in the process. Visualization is not only a mere presentation and it should communicate the information in a proper and effective manner, which should be unambiguous and easy to understand for the targeted viewers. All these business issues have been taken care in this pattern, which allows different involved parties to specify requirements that can be used by the mechanisms, which are used in the visualization process.

B.6 Tips and heuristics

Finding the EBTs and BOs does not come immediately. Designer needs to understand the field completely, then should run lot of iterations before finalizing the EBTs and BOs whereas identifying IOs is very easy, anything, which is tangible, or can be replaced, and changes over time, are IOs.

Inheritance is a factor, which one should use with maximum caution; if possible, aggregation will provide a dynamic relationship, rather than inheriting everything from base class. The link between parent and child is weak in case of aggregation, whereas it

is a strong link in case of inheritance. For this reason, aggregation is usable, while defining the hook to AnyMechanism, AnyEntity, AnyPresentation, and AnyMedia.

B.7 Conclusions

As discussed, visualization is very important in analyzing patterns. Visual presentation makes any document easy to understand, and it works as an effective way of communicating information from one end to other. The domains, where this Visualization pattern used, are vast and advantages of using it are unlimited. This paper shows how the concept of visualization makes excellent patterns in health care applications and geographic information systems. Interactive data mining systems are becoming increasingly popular in health care and pharmaceutical industries. Visualization is the most important concept used in such application, as the health condition of a patient are monitored, by using a simple graph or a three dimensional view of the human body.

An analysis pattern for visualization has been well developed by using stable design concepts, like AnyMechanism, AnyPresentation, and AnyEntity etc. There are certain limitations addressed in the present chapter. In addition, such limitations could be handled very easily. One such limitation is that the patterns described do not provide any application specific context that can help to understand the applicability of the pattern in a specific context. However, the pattern provides a common core, which is usable in different application domains. Another limitation addressed is, no quality monitoring,

which is manageable by attaching other stable analysis patterns, such as monitoring or recoding with the application aspect. Therefore, the stable pattern developed in this paper for visualization, is very flexible, and it ties domain independent core knowledge together, which is applicable in any application, as the core engine.

Appendix C: The AnyPrecision stable design pattern

“It is the mark of an instructed mind to rest satisfied with the degree of precision which the nature of the subject admits and not to seek exactness when only an approximation of the truth is possible.” - Aristotle

C.1 Abstract

The AnyPrecision pattern essentially models the core knowledge of precision, in a domain independent way, to be usable in any application process, where a fair degree of precision is always important and critical for the measured data. One can use this pattern as the core logic, for the aspect of precision in any application, rather than implementing it as a part of the application. To achieve this goal, this thesis uses the concept of “Software Stability Model” (SSM), to identify and isolate the core knowledge of precision, from the application specific knowledge. There are several scenarios to show the applicability and reusability of this pattern.

C.2 Introduction

The usage and utilization of precision may vary, from very simple to very complex, based on the usage in many domains for many different purposes. Here are some samples of usages, utilization, and purposes of precision:

The use of precision is deeply rooted in our civilization: in other words, civilizations depend on a correlation process, where many things work together in combination to

achieve a single goal. To do that, each action for individual entity has to be very precise; otherwise, it will never work for a group. For example, to assemble a machine all individual parts have to be precise and up-to-date, otherwise it will not fit.

The degree of precision varies based on the need; for example, while building a house, bricks can have precision in millimeter range, whereas, in an electronic chip, the dimension of the transmitter has to be very precise and in micron levels. Precision is the exactness of data; if the measurement is consistent in respect of actual findings, then it is always precise.

In a mathematical term, it is representing by the digits, soon after the decimal point. Precision is very important and critical in measurement and design process, as it dictates the exactness of any information or any mechanism. For example, if the temperature of boiling water is measured a couple of times, and if the measurement gives the values of; 210.1F, 210.2F, 210.6F, 210.8F, 210.9F, 210.5F, and then it could be said that the distribution is displaying that the measurement is quite accurate and precise, because all the measured values are leaning towards the high range.

From engineering to commerce, and from medicine, to day-to-day life precision plays a very important role in our daily life. If the computer clock is not precise and accurate, then one cannot bid at right time on eBay. The reliability of the result entirely depends on the level of precision. If the precision level is high, then one could always expect better performances.

Sometimes accuracy and precision are interchangeable: But, accuracy, mostly tells us regarding the reliability of the results, whereas, precision relates with the perfection of the method used to obtain the result.

Precision can also be used to represent the abstract things, like expressing ideas and answering questions: There is no doubt that precision is very important in every field of life, and it could be associated with various domains; therefore, we need a stable model that could accommodate various situations, without making much changes in design. Precision uses the property of a measuring instrument. Its usage and importance is far beyond than just a simple property of the instrument. We can not simply judge the effect of precision, based on the instrument itself, without taking into account the context, where it is used and what is the expected level of precision from the involved parties, in a specific given application. This thesis has considered all these aspects of precision, while using this pattern.

The degree of precision varies widely based on the testing capability of the system, and the expected result. Different domains have their own limitations and testing requirements, for example, when we are measuring our height, we can use any ruler that can measure in inches; but, if we have to measure the dimension of a microprocessor chip, it has to be very accurate at the micron level. Any pattern in precision should enforce the required precision level based on the requirements of the domains.

This paper intends to isolate the core knowledge of precision concept from the application specific logic, and present it later, as a stable analysis pattern, to provide the reusable core for other applications sharing the same core concept. In order to achieve

this goal, the AnyPrecision pattern is designed based on the software stability model.

SSM provides a stable and *reusable* core for multiple applications, which are sharing the same core knowledge. Software stability describes the knowledge as design pattern named AnyPrecision. The pattern is documented using Fayad's pattern documentation template [B15].

C.3 Pattern documentation

C.3.1 Name: AnyPrecision stable design pattern

Precision is an accurate science or knowledge, which specifies the degree of exactness; it is unrelated to any specific application or domain. As the concept is generic in nature, the AnyPrecision described in this paper, is also very stable and generic, and not tied to any application specific logic. The patterns describe the core logic behind precision, and how different entities interact with each other to achieve the precision in the context, and where the patterns will be applied. Therefore, the name AnyPrecision is chosen as the name of the pattern, where the word "Any," always signifies any domain or context, where the pattern can be applied.

Precision is also about exactness. People often misunderstood accuracy with precision. However, these two terms have different meanings; accuracy is the degree of conformity with a standard. Accuracy is measurable in the quality of a result, whereas precision relates to the quality of the operation that provides the desired result.

C.3.3 Problem

Major problems with the present precision model are:

- a) Mathematical: Most of the present-day models deal only with the aspect of mathematical precision; for example, in a floating-point number, how many digits are there after the decimal sign? Precision also plays a major role in mathematics and statistics, but it is also possible to use in other fields of study as well. For example, precision is very important in text mining areas. The quality of any text-mining algorithm depends on how precisely it can narrow down to the result of a search. The existing model is tough to apply under such cases.
- b) Extensibility: Any pattern in precision should provide ways, by which they are extendable further to provide the functionality, needed for the specific domain.
- c) Accuracy: The accuracy of measurement depends on the specific requirements and needs of the application. The pattern should provide a capability to indicate the required precision level for a specific domain.
- d) Exactness: In other words, precision provides exactness in measurement. The degree of exactness varies based on the mechanism used in the process; the pattern should provide ways to monitor the exactness, provided by the mechanism used in the measurement process.
- e) Consistency: The pattern should provide an effective way to measure the consistency in the results, by validating precision levels needed for an

application, with the precision provided by the mechanism used in the measurement process.

Given the above issues, the real problem is how we can develop a model based on the core knowledge of precision, which are reusable by different applications in different domains and extended, as and when needed.

C.3.4 Context

Precision is mainly associated with the measurement, but sometimes related to abstract things as well, like expressing ideas precisely or providing precise answers to a question. In the context of measurement, precision relates to an engineering precision, which describes how precisely different objects put together to build a structure, like how to design different component to build a bridge. In data mining, precision describes the accuracy of the data and the analysis.

Precision is used everywhere, from a laser guided missile systems, to agriculture, and even in our daily life, it plays a very important and critical role. During day-to-day communication, if the words are precise and up to the point, then people can understand dialogues easily and effectively. However, every application needs a high level of precision, but there is no easy way to express that logic into a pattern, into an easily measured and monitored form. This thesis has developed a stable design pattern AnyPrecision, used in any application and in any given domain, where the aspect precision is too important and needs strict monitoring and measurement.

In measurement, one can use precision to describe the limitation of instrument; for example, different instruments have specific and precise jobs to do. Depending on the requirement of the jobs, the instrument is developed by keeping a predefined precision level. It is important to match that level with the requirement of the application, where the instrument is currently used. That is why precision is so important in the context of any measurement, because it provides the most important metric in any measurement.

Precision will find the usability of any mechanism; for example, in text mining, precision can verify, if the algorithm used to return search result is actually returning the cases, for which user is looking for. In this case, one can use precision to measure user satisfaction, which is quite different from measuring a length, but the underlying usage of precision is identical.

C.3.5 Constraints and challenges

The AnyPrecision pattern should resolve the following constraints and challenges:

Challenges:

- a) Precision relates to any object or entity; for example, in the case of manufacturing a chip, precision relates to the measurement of the electronic components, used to build the chip, whereas in a data mining application, the success depends on the data collected for the data mining process.
- b) Precision is often limited to the ability of the instrument or mechanism used in the measurement process. For example, a measurement ruler can hardly measure any length less than 1 millimeter, where sophisticated instrument can

measure distance as small as 1 micron.

- c) The goal of the any measurement is to specify the degree of precision in the data, which depends on the mechanism or instrument, used to do measurement job, therefore, different mechanisms will achieve different degrees of precision purely based on the needs.

Constraints:

- a) The AnyPrecision pattern does not specify the exact mechanism used in the measurement, or the degree of exactness that is required by the measurement process. It depends on the application logic to assess, which tools or mechanism can get the desired level of precision in the measurement.
- b) There should be at least one entity present, to be used in the measurement, by one or multiple actors.
- c) At least one mechanism should be present, which will have some indicators that relate to the precision.
- d) The actors involved in the measurement process should impose a zero level to many criteria, used by the measurement process.
- e) The metric should produce at least one model, which will be used the context of the precision.
- f) There should be at least one instance of the data, used by the measurement process.
- g) AnyModel should relate to at least one AnyData.

C.3.6 Solution

Very few attempts have been made in the past to design patterns based on precision, most of the existing design on precision are application specific, and they can only work in a particular set of data or can handle particular type of problems. This pattern has tried to overcome that problem, by separating the application specific logic from the core logic, and tried to implement the core logic based on EBT and BO, which are stable and generic, therefore, can be used in any application in any domain as shown in Figure 48.

Figure 48 - Class diagram of the AnyPrecision pattern

Source: Adapted from Fayad's pattern archive

Pattern structure and participants

The participants of the Interaction pattern are:

Classes:

- a) AnyPrecision: This class describes the core knowledge of this stable design pattern, and provides a way to define the precision needed for any application based on the mechanism used. It also provides capability to monitor the output of any algorithm or mechanism used in measurement.

Patterns:

- a) Measurement: This represents the EBT employed by AnyPrecision, as the goal of the pattern. Any actors can use Measurement to run several different mechanisms based on the context.
- b) AnyModel: This represents the model produced by the metric; it can represent accuracy, exactness, or consistency.
- c) AnyMetric: AnyMetric represents the method or procedures used to measure certain properties of any entity based on the defined criteria that is defined. AnyMetric can range from mathematical algorithms, such as binary tree to any mechanical procedure, such as using a ruler to measure the length of a table.
- d) AnyData: This represents the input data to the measurement process.
- e) AnyParty: This represents the actors, those who are involved in the measurement process. The actor can also be a verifier of the output, who can help to determine the degree of precision in the result data.

- f) **AnyCriteria**: This pattern helps to define a criterion that is associated with properties or mechanism. For example, different tools have a pronounced limitation, while measuring any property of an object, such as simple ruler will fail to measure length, smaller than millimeter.

CRC- Cards:

Measurement (Measurement handler)		
Responsibility	Collaboration	
Defines the process by registering entities measured, by using different mechanism.	Client	Server
	AnyParty AnyMetric AnyData	specifyMechanism() definePrecision() doMeasurement() measureEntity()
Attributes: name, type, description, metric, precision, data		

AnyMetric (Mechanism descriptor)		
Responsibility	Collaboration	
Represents the generic interface for different mechanism used to measure the entities.	Client	Server
	Measurement AnyCriteria AnyModel AnyPrecision	runMechansism() addProperties() registerConstraints()
Attributes: name, constraints, procedure, properties, precision		

AnyPrecision (Measurement descriptor)		
Responsibility	Collaboration	
Represents the precision that is defined by the measurement process.	Client	Server
	AnyModel AnyCriteria	definePrecision() validatePrecision() defineUnits()
Attributes: units, value, range		

AnyData (Data descriptor)		
Responsibility	Collaboration	
Describes the object being measured.	Client	Server
	Measurement AnyModel	measures() propertyToMeasured()
Attributes: id, name, property, model		

AnyCriteria (Constraint definer)		
Responsibility	Collaboration	
Describes the limitation of different mechanism or tools used in the measurement.	Client	Server
	AnyParty AnyMetric	imposes() modifyConstraint()
Attributes: type, rule, property		

AnyParty (Measurement inducer)		
Responsibility	Collaboration	
Represents the operator of the measurement process, who runs the measurement process or verify the precision in the output.	Client	Server
	Measurement AnyCriteria	initiate() monitorProcess() validateResult()
Attributes: name, role		

AnyModel (Precision indicator)		
Responsibility	Collaboration	
Represents the degree of precision in the output.	Client	Server
	AnyPrecision AnyMetric AnyData	indicatePrecison() registerIndicator()
Attributes: type, value, precision		

C.3.7 Consequences

The benefits of using the AnyPrecision pattern are:

Benefits:

- a) Scalable: The AnyPrecision pattern is a design pattern that based on stability design model. The core knowledge of precision has been defined based on EBTs and BOs, whereas, IOs can be added based on the application logic. The pattern is not dependent on any specific business logic; therefore, one can use it in any application as the core model.
- b) Stable: EBT and BO can help achieve a fair amount of stability. By definition, EBT and BO does not change over time, nor they change based on application requirements, and they are the core knowledge behind the precision concept.
- c) Support different type of objects: One can work out and introduce the AnyPrecision pattern to different entities. For example, it can be a simple table to a spacecraft depending on the application.
- d) Support different properties: The properties of the participating entity can be different based on application logic. In the case of precision navigation, velocity is the property, which requires a precision measurement.
- e) Flexible: One can work out and introduce the AnyPrecision pattern to different domains as discussed before. The AnyPrecision pattern is also a precision navigation tool for a submarine or can describe a process to measure a table using a ruler.

- f) Adaptable: Using IOs, one can apply the same pattern in different domains.

The patterns do not dictate the IOs; rather it specifies how to lay out the core knowledge, and how application logic fits into it very easily.

Limitations:

- a) No implementation logic: AnyPrecision is a conceptual model; yet, all the components used in this pattern have clear names and functionality, which helps to visualize the pattern easily, but the pattern does not provide any actual implementation of the pattern a specific context. One-way, this is a severe limitation, but on the other hand, the pattern provides a common core, used in different applications, as the core knowledge is untied with the specific application logic.
- b) No monitoring of result: As the quality of precision is applicable on a specific context, it is yet to be considered as the core knowledge of precision pattern. This limitation is override by attaching other stable analysis pattern, such as monitoring or recoding with the AnyPrecision pattern.

C.4 Applicability with illustrated examples

We hereby present two case studies to show the application of the AnyPrecision pattern in different domains:

One is the use of precision measurement in navigation, and the other, is the use of precision to verify the output of a text mining applications. For the sake of simplicity, the use case only focuses on the usage on the AnyPrecision pattern in the problem space;

therefore, the design does not include other classes or component involved in the specific application.

C.4.1 Case study 1: Precision navigation

Precision navigation is very vital for providing accurate coordinate information for any moving object, ranging from missile to satellite. With the help of new and modern technology, we can precisely locate a moving object or can predict the trajectory of movement, which later helps us to send space missions to Mars. To provide such a high degree of precision, the application needs precise development, all the while considering the limitation of instruments that requires proper design and monitoring of each component used in the system. In addition, one needs to monitor any measurement to enforce the demanded precision. To model such an application, one needs to possess the deep knowledge of the topic of precision and methods that can ensure a fair degree of precision. The next section describes a precise navigational application based entirely on the AnyPrecision pattern. Figure 49 describes the class diagram for this case study.

.

Figure 49 - Class diagram of a precision navigation system for a submarine

Use case: Precision navigation system

Use case description: To provide clarity, use case, along with the sequence diagram are presented, which shows how precision will assist in guiding a submarine.

Use case id: 1.0

Use case title: Guide submarine

Actors	Roles
AnyActor	SubmarineDriver
AnyActor	RemoteOperator

Class	Type	Attributes	Interfaces
Measurement	EBT	Title, description, metric	request()
AnyParty	BO	name, responsibility	initiate()
AnyCriteria	BO	name, description, type, value	validate () impose ()
AnyModel	BO	precision, type, description	produces()
AnyPrecision	BO	name, description, list of services,	loacteResturants()
PrecisionAccelerometer	IO	name, description, precision, mechanism	run()
Location	IO	Position, time, precision	relatedTo()
DragFreeControlCriteria	IO	Name, description,	create()

Enduring Business Themes (EBT): Measurement

Business Objects (BO): AnyParty, AnyCriteria, AnyModel, AnyPrecision

Industrial Objects (IO): PrecisionAccelerometer, Location, DragFreeControlCriteria

Description of the use case:

- a) AnyParty <<BO>>, which can be any SubmarineDriver <<IO>> or RemoteOperator<<IO>>, defines AnyCriteria<<BO>> such as DragFreeControlCriteria<<IO>>Measurement<<EBT>> of the Velocity<<IO>> of the Submarine<<IO>>.
- b) AnyParty <<BO>> sends a request to Measurement <<EBT>> to determine the location of the Submarine<<IO>> using AnyPrecision<<BO>>.
- c) Measurement <<EBT>> reads the criteria from DragFreeControlCriteria<<IO>>, then issues a run request to PrecisionAccelerometer<<IO>>.
- d) PrecisionAccelerometer<<IO>> produces the model with the AnyPrecision<<BO>> defined for that metric.
- e) AnyModel<<BO>> relates the AnyPrecision<<BO>> with the determined Location<<IO>> at that point of time.
- f) PrecisionAccelerometer<<IO>> returns the Location<<IO>> with the AnyPrecision<<IO>>.

Sequence diagram:

Figure 50 describes the sequence diagram for this use case.

Figure 50 - Sequence diagram of a precision navigation system for a submarine

C.4.2 Case study 2: Text mining

Precision is a term used in text mining, as the measurement of system performance.

Precision denotes the ratio of spots correctly identified by the user, as on topic, out of all the spots reported as being on topic. As correct identification of spots based on user queries is always desirable in any text-mining algorithm, it is also very important and

critical to measure the degree of precision, and monitor the performance of the text mining applications. The case study shows how any text mining application can use the AnyPrecision pattern to monitor and evaluate the performance of its algorithm. For the sake of simplicity, this case study focuses only on the area of precision in text mining application, by omitting other details on some specific purposes. The class diagram for this case study is described in Figure 51.

Figure 51 - Class diagram of a text mining application

Use case: Text mining

Use case description: To provide clarity, a use case is presented along with the sequence diagram is presented, which shows how precision can be used measure the usefulness of a search algorithm used in a text mining application.

Use case id: 1.0

Use case title: Calculate precision in a text search algorithm

Actors	Roles
AnyActor	Website
AnyActor	Disambiguator

Class	Type	Attributes	Interfaces
Measurement	EBT	title, description, metric	request()
AnyParty	BO	name, responsibility	initiate()
AnyCriteria	BO	name, description, type, value	validate () impose ()
AnyModel	BO	precision, type, description	produces()
AnyPrecision	BO	name, description, list of services,	loacteResturants()
PrecisionAccelerometer	IO	name, description, precision, mechanism	run()
Location	IO	Position, time, precision	relatedTo()
DragFreeControlCriteria	IO	Name, description,	create()

Enduring Business Themes (EBT): Measurement

Business Objects (BO): AnyParty, AnyCriteria, AnyModel, AnyPrecision

Industrial Objects (IO): PrecisionAccelerometer, Location, DragFreeControlCriteria

Description of the use case:

- a) AnyParty <<BO>>, which can be any SubmarineDriver <<IO>> or RemoteOperator<<IO>>, defines AnyCriteria<<BO>> such as DragFreeControlCriteria<<IO>>Measurement<<EBT>> of the Velocity<<IO>> of the Submarine<<IO>>.
- b) AnyParty <<BO>> sends a request to Measurement <<EBT>> to determine the location of the Submarine<<IO>> using AnyPrecision<<BO>>.
- c) Measurement <<EBT>> reads the criteria from DragFreeControlCriteria<<IO>> then issues a run request to PrecisionAccelerometer<<IO>>.
- d) PrecisionAccelerometer<<IO>> produces the model with the AnyPrecision<<BO>> defined for that metric.
- e) AnyModel<<BO>> relates the AnyPrecision<<BO>> with the determined Location<<IO>> at that point of time.
- f) PrecisionAccelerometer<<IO>> returns the Location<<IO>> with the AnyPrecision<<IO>>.

Sequence diagram:

Figure 52 describes the sequence diagram for this use case.

Figure 52 - Sequence diagram of a precision navigation system for a submarine

C.5 Design & implementation issues

The main implementation issues related to the implementation of EBT and BO are:

C.5.1 Separation of concerns

Software stability separates the pattern into three different layers as EBTs, BOs, and IOs, all tackled individually, with explicit dependencies among them. EBTs are

implemented as interfaces, BOs are partly interfaces partly business logic, whereas IOs are the concrete classes, which implement the application specific logic.

C.5.2 Hooks

Hooks are extension points on the BOs, which are adaptable by IOs. Hooks are usable in several different ways, based on the capability provided by the language. In general, hooks describe a structured and uniform method to allow the application developer to determine, how to use the capabilities provided by the pattern.

This section provides the description on various hooks that attach to this pattern.

- a) AnyParty: The involved parties in the measurement process are extendable by attaching the proper IO with this BO. For example, carpenters use simple measure tool to measure the dimension of the furniture, whereas automated robotic machine measure the dimension of an electronic chip. All these different roles of the actors can be extended from AnyParty by using hooks.
- b) AnyCriteria: This BO can have a hook attached to it, to provide different criteria that can influence the mechanism, used in the measurement process to keep track the precision involved in measurement.
- c) AnyPrecision: The level of precision varies from application to application. AnyPrecision provides hook, which can be used by the application to set the value and type of the precision as needed.

C.5.3 Business issues

The main purpose of precision is to build a stable model to verify and enforce the requirement of the precision in the measurement. The major issue faced by most businesses of the day regarding precision is how to enforce the precision needed for a specific application, as they differ depending the use of the measurement data, and based on the tools used in the measurement process. The AnyPrecision pattern tries to solve that issue by defining the relation between precision and measurement, with the requirement specific to any application.

C.6 Tips and heuristics

Finding the EBTs and BOs does not come immediately or quickly. Designer needs to understand the field completely in a thorough manner, and then should run lot of iterations before finalizing the EBTs and BOs, whereas identifying IOs is very easy, anything, which is tangible, can be replaced, changes over time are, IOs.

Inheritance needs caution while using, while using aggregation provides a dynamic relationships rather than inheriting everything from base class. The link between parent and child is weak in case of aggregation, whereas it is a strong link in case of inheritance.

C.7 Conclusion

This chapter intends to present the AnyPrecision pattern that models the core knowledge of precision in a domain independent way. This chapter also shows that it can be used in any application, where, for the factor of precision is also important for the measured data, it also details all the problems, challenges, and solution of AnyPrecision model. Two separate case studies show the application of the AnyPrecision pattern under various domains. The first case study talks about of precision measurement in the navigation. The precision navigation mainly talks and deliberates about, how with the help of this new technology, we can precisely locate a moving object or can predict the trajectory of movement.

For an easy understanding, we have also provided the class diagram of precision navigation. The other case study for AnyPrecision, presented in this chapter, is the Text mining. This case study shows how we can achieve the desired precision in any text-mining algorithm. It also explains the importance of measuring the precision and monitoring the performance of the text mining application. The sequence diagram facilitates better understanding. To develop AnyPrecision model, we have used the concept of “Software Stability Model” (SSM). Further, SSM identifies and isolates the core knowledge of precision from the application specific knowledge.

Appendix D: Implementation & code

This appendix provides the implementation details for the Classification pattern implemented as a part of the thesis. Figure 53 represents the class diagram for the “Hurricane Application.”

Figure 53 - Class diagram of a hurricane classification system

Code for Classification pattern

```
package com.udme.patterns.Classification;
```

```
/**  
 * <p>Title: AnyCategory</p>  
 *  
 * <p>Description: Describes AnyCategory</p>  
 *  
 * <p>Copyright: Copyright (c) 2007</p>  
 * @author Somenath Das  
 * @version 1.0  
 */
```

```
public class AnyCategory  
{
```

```
    private String mCode = "";  
    private String mName = "";  
    private String mDescription = "";
```

```
    public AnyCategory(String code, String name, String description)  
    {  
        mName = name;  
        mCode = code;  
        mDescription = description;  
    }
```

```
    public String getCategoryCode()  
    {  
        return mCode;  
    }
```

```
    public void setCategoryCode(String str)  
    {  
        mCode = str;  
    }
```

```
    public String getName()  
    {
```

```

        return mName;
    }

    public void setName(String str)
    {
        mName = str;
    }

    public String getDescription()
    {
        return mDescription;
    }

    public void setDescription(String str)
    {
        mDescription = str;
    }
}

package com.udme.patterns.Classification;

/**
 * <p>Title: AnyCriteria</p>
 *
 * <p>Description: Describes AnyCriteria</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

public class AnyCriteria
{
    private AnyCategory mCategory = null;

    public AnyCriteria()
    {

    }
}

```

```

    public String getName()
    {
        return "";
    }

    public void attachCategory(AnyCategory category)
    {
        mCategory = category;
    }

    public AnyCategory getCategoryForCriteria()
    {
        return mCategory;
    }
}

```

```

package com.udme.patterns.Classification;

```

```

/**
 * <p>Title: AnyEntity</p>
 *
 * <p>Description: Describes AnyEntity</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

```

```

public class AnyEntity
{

    private String mName = "";
    private String mId = "";

    public void setName(String str)
    {
        mName = str;
    }

    public void setId(String str)
    {

```

```

        mId = str;
    }

    public String getName()
    {
        return mName;
    }

    public String getId()
    {
        return mId;
    }

    public void initiate()
    {

    }

    public void addCriteria(AnyCriteria criteria)
    {

    }

}

```

```

package com.udme.patterns.Classification;

```

```

/**
 * <p>Title: AnyMechanism</p>
 *
 * <p>Description: Describes AnyMechanism</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

```

```

import java.util.Vector;

```

```

public class AnyMechanism

```

```

{

private Vector mCriteriaList = new Vector();
private AnyEntity mEntity;
private AnyMechanismInterface mMechanismImpl;


public AnyMechanism()
{

}

public void loadCriteria()
{
    mMechanismImpl.loadCriteria();
}

public void registerEntity(AnyEntity entity)
{
    mEntity = entity;
}

public void registerMechanismImpl(AnyMechanismInterface mechanism)
{
    mMechanismImpl = mechanism;
}

public Vector getCriteriaList()
{
    return mCriteriaList;
}

public AnyEntity getEntity()
{
    return mEntity;
}

public AnyCategory runs()
{
    return mMechanismImpl.runs();
}

```



```
}
```

```
package com.udme.patterns.Classification;
```

```
/**
```

```
 * <p>Title: AnyMechanismInterface</p>
```

```
 *
```

```
 * <p>Description: Describes AnyMechanismInterface</p>
```

```
 *
```

```
 * <p>Copyright: Copyright (c) 2007</p>
```

```
 * @author Somenath Das
```

```
 * @version 1.0
```

```
 */
```

```
public interface AnyMechanismInterface
```

```
{
```

```
    public void loadCriteria();
```

```
    public AnyCategory runs();
```

```
}
```

```
package com.udme.patterns.Classification;
```

```
import com.udme.IOs.WindSpeedComparison;
```

```
/**
```

```
 * <p>Title: AnyParty</p>
```

```
 *
```

```
 * <p>Description: Describes AnyParty</p>
```

```
 *
```

```
 * <p>Copyright: Copyright (c) 2007</p>
```

```
 * @author Somenath Das
```

```
 * @version 1.0
```

```
 */
```

```

public class AnyParty
{
    private static String name = "";
    public AnyParty()
    {}

    public void setName(String Name)
    {
        name = Name;
    }

    public String getName()
    {
        return name;
    }

    public AnyCategory classify(AnyMechanism mechansim, AnyEntity entity)
    {
        Classification classification = new Classification();
        classification.registerMechanism(mechansim);
        return classification.requestClassification(entity);
    }

    public void addCriteria(AnyCategory category, AnyCriteria criteria)
    {
        criteria.attachCategory(category);
    }
}

package com.udme.patterns.Classification;

/**
 * <p>Title: AnyType</p>
 *
 * <p>Description: Describes AnyType</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0

```

```

*/

public class AnyType
{
    private static String mName = "Default";
    private static int mId = 0;

    public AnyType()
    {
    }

    public void setId(int id)
    {
        mId = id;
    }

    public void setName(String name)
    {
        mName = name;
    }

    public String getName()
    {
        return mName;
    }

    public int getId()
    {
        return mId;
    }
}

package com.udme.patterns.Classification;

/**
 * <p>Title: Classification</p>
 *
 * <p>Description: Describes Classification</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das

```

```

* @version 1.0
*/
class Classification
{

    private AnyMechanism mMechansim;

    public Classification()
    {
    }

    public void registerMechanism(AnyMechanism mechansim)
    {
        mMechansim = mechansim;
    }

    public AnyCategory requestClassification(AnyEntity entity)
    {
        mMechansim.loadCriteria();
        mMechansim.registerEntity(entity);
        return mMechansim.runs();
    }
}

```

Code for IOs

```

package com.udme.IOs;

/**
 * <p>Title: CriteriaList</p>
 *
 * <p>Description: Describes CriteriaList</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

import java.io.*;
import java.util.Vector;
import com.sd.util.DirTree;

```

```

import javax.swing.JOptionPane;

public class CriteriaList
{
    private Vector mCriteriaList = new Vector();
    public CriteriaList()
    {
    }

    public void classify()
    {

    }

    public void load()
    {
        try
        {
            File file = new File(DirTree.CONF_PATH);
            File[] fileList = file.listFiles();

            if (fileList == null)
            {
                return;
            }
            for (int k = 0; k < fileList.length; k++)
            {
                HurricaneCategory data = new HurricaneCategory();
                data.readData(fileList[k].getPath());
                data.printData();

                long minSpeed = data.getMinimumWindSpeed();
                long maxSpeed = data.getMaximumWindSpeed();
                System.out.println("minSpeed = " + minSpeed);
                System.out.println("maxSpeed = " + maxSpeed);

                WindSpeedCriteria criteria = new WindSpeedCriteria(minSpeed,maxSpeed);
                criteria.attachCategory(data.getCategory());
                mCriteriaList.add(criteria);
            }
        }
    }
}

```

```

    }

    catch (Exception ex)
    {
        JOptionPane.showMessageDialog(null,
            "Error reading directory " +
            DirTree.CONF_PATH,
            DirTree.APP_NAME,
            JOptionPane.WARNING_MESSAGE);
    }
}

public static void main(String[] args)
{
    CriteriaList list = new CriteriaList();
    list.load();
}

}

package com.udme.IOs;

/**
 * <p>Title: Hurricane</p>
 *
 * <p>Description: Describes Hurricane</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

import com.udme.patterns.Classification.AnyEntity;
import com.udme.patterns.Classification.AnyCategory;
import com.udme.patterns.Classification.AnyType;

import com.sd.util.CustomData;

```

```

import com.sd.util.GlobalData;

public class Hurricane extends CustomData
{
    long mSpeed = 0;
    String mPlace = "";
    String mStartDate = "";
    AnyCategory mCategory = new AnyCategory("", "Not Categorized", "");
    AnyEntity mEntity = new AnyEntity();

    public long getWindSpeed()
    {
        return mSpeed;
    }

    public void setWindSpeed(long speed)
    {
        mSpeed = speed;
    }

    public AnyEntity getEntity()
    {
        return mEntity;
    }

    public String getPlace()
    {
        return mPlace;
    }

    public void setPlace(String place)
    {
        mPlace = place;
    }

    public String getName()
    {
        return mEntity.getName();
    }

    public void setName(String name)
    {

```

```

        mEntity.setName(name);
    }

    public String getId()
    {
        return mEntity.getId();
    }

    public void setId(String id)
    {
        mEntity.setId(id);
    }

    public String getStartDate()
    {
        return mStartDate;
    }

    public void setStartDate(String date)
    {
        mStartDate = date;
    }

    public AnyCategory getCategory()
    {
        return mCategory;
    }

    public void setCategory(AnyCategory category)
    {
        mCategory = category;
    }

    /**
     * This method prints the values of the attributes to System.out
     */
    public void printData()
    {
        GlobalData.log(toString());
    }

```



```

public String toString()
{
    String outputStr = new String(
        "\nHurricane Name: " + mEntity.getName() +
        "\nId      : " + mEntity.getId() +
        "\nCategory   : " + mCategory.getName() +
        "\nWindSpeed   : " + mSpeed +
        "\nStartDate   : " + mStartDate +
        "\nPlace      : " + mPlace);
    return outputStr;
}

public void copy(CustomData data)
{
    Hurricane hurricane = (Hurricane) data;

    mSpeed = hurricane.getWindSpeed();
    mEntity.setId(hurricane.getEntity().getId());
    mEntity.setName(hurricane.getEntity().getName());
    mPlace = hurricane.getPlace();
    mStartDate = hurricane.getStartDate();
    mCategory.setCategoryCode(hurricane.getCategory().getCategoryCode());
    mCategory.setDescription(hurricane.getCategory().getDescription());
    mCategory.setName(hurricane.getCategory().getName());

}

}

```

```

package com.udme.IOs;

```

```

/**
 * <p>Title: HurricaneCategory</p>
 *
 * <p>Description: Describes HurricaneCategory</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

```

```

import com.sd.util.GlobalData;
import com.sd.util.CustomData;
import com.udme.patterns.Classification.AnyCategory;

public class HurricaneCategory extends CustomData
{
    // Attributes

    private AnyCategory mCategory = new AnyCategory("", "", "");
    WindSpeedCriteria mWindSpeedCriteria = new WindSpeedCriteria(0, 0);

    // Methods

    /**
     * This method prints the values of the attributes to System.out
     */
    public void printData()
    {
        GlobalData.log("Category Code: " +mCategory.getCategoryCode());
        GlobalData.log("Name      : " +mCategory.getName());
        GlobalData.log("Description : " +mCategory.getDescription());
        GlobalData.log("MinimumWindSpeed : " +
            mWindSpeedCriteria.getMinimumWindSpeed());
        GlobalData.log("MaximumWindSpeed : " +
            mWindSpeedCriteria.getMaximumWindSpeed());
    }

    public String toString()
    {
        String outputStr = new String("\n\nCategory Code: " +
            mCategory.getCategoryCode() +
            "\nName      : " + mCategory.getName() +
            "\nDescription   : " +
            mCategory.getDescription() +
            "\nMinimumWindSpeed : " +
            mWindSpeedCriteria.getMinimumWindSpeed() +
            "\nMaximumWindSpeed : " +
            mWindSpeedCriteria.getMaximumWindSpeed());

        return outputStr;
    }

    public String getCategoryCode()
    {

```

```

    return mCategory.getCategoryCode();
}

public void setCategoryCode(String str)
{
    mCategory.setCategoryCode(str);
}

public String getName()
{
    return mCategory.getName();
}

public void setName(String str)
{
    mCategory.setName(str);
}

public String getDescription()
{
    return mCategory.getDescription();
}

public void setDescription(String str)
{
    mCategory.setDescription(str);
}

public long getMinimumWindSpeed()
{
    return mWindSpeedCriteria.getMinimumWindSpeed();
}

public void setMinimumWindSpeed(long data)
{
    mWindSpeedCriteria.setMinimumWindSpeed(data);
}

public long getMaximumWindSpeed()
{
    return mWindSpeedCriteria.getMaximumWindSpeed();
}

public void setMaximumWindSpeed(long data)

```

```

    {
        mWindSpeedCriteria.setMaximumWindSpeed(data);
    }

    public AnyCategory getCategory()
    {
        return mCategory;
    }

    public WindSpeedCriteria getWindSpeedCriteria()
    {
        return mWindSpeedCriteria;
    }

    public void setWindSpeedCriteria(WindSpeedCriteria criteria)
    {
        mWindSpeedCriteria = criteria;
    }

    public void copy(CustomData data)
    {
        HurricaneCategory HurricaneCategory = (HurricaneCategory) data;
        mCategory.setCategoryCode(HurricaneCategory.getCategoryCode());
        mCategory.setName(HurricaneCategory.getName());
        mCategory.setDescription(HurricaneCategory.getDescription());

        setMinimumWindSpeed(HurricaneCategory.getMinimumWindSpeed());
        setMaximumWindSpeed(HurricaneCategory.getMaximumWindSpeed());
    }
}

package com.udme.IOs;

/**
 * <p>Title: Operator</p>
 *
 * <p>Description: Describes Operator</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0

```

```

*/

import com.udme.patterns.Classification.AnyMechanism;
import com.udme.patterns.Classification.AnyParty;
import com.udme.patterns.Classification.AnyCategory;

public class Operator extends AnyParty
{
    public Operator()
    {
    }

    public void doHurricaneClassificationBasedOnWindSpeed(Hurricane hurricane)
    {

        AnyMechanism mechansim = new AnyMechanism();
        WindSpeedComparison algorithm = new WindSpeedComparison(mechansim);
        algorithm.setHurrinaceInstance(hurricane);
        mechansim.registerMechanismImpl(algorithm);
        AnyCategory category = classify(mechansim, hurricane.getEntity());

        if (category != null)
        {
            hurricane.setCategory(category);
            System.out.println("category " + category.getName());
        }
        else
        {
            hurricane.getCategory().setName("No category found");
            System.out.println("No category found");
        }

    }

    public WindSpeedCriteria defineWindSpeedCriteria(AnyCategory category, long
minSpeed, long maxSpeed)
    {
        WindSpeedCriteria criteria = new WindSpeedCriteria(minSpeed, maxSpeed);
        addCriteria(category, criteria);
        return criteria;
    }
}

```

```

public static void main(String[] args)
{

    Hurricane hurricane = new Hurricane();
    Operator opertor = new Operator();
    opertor.doHurricaneClassificationBasedOnWindSpeed(hurricane);

}

}

```

```

package com.udme.IOs;

```

```

/**
 * <p>Title: WindSpeed</p>
 *
 * <p>Description: Describes WindSpeed</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

```

```

public class WindSpeed {
    public WindSpeed() {
    }
}

```

```

package com.udme.IOs;

```

```

/**
 * <p>Title: WindSpeedComparison</p>
 *
 * <p>Description: Describes WindSpeedComparison</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0

```

```

*/

import com.udme.patterns.Classification.AnyMechanism;
import com.udme.patterns.Classification.AnyMechanismInterface;
import com.udme.patterns.Classification.AnyEntity;
import com.udme.patterns.Classification.AnyCategory;

import java.util.Vector;
import java.io.File;
import javax.swing.JOptionPane;
import com.sd.util.DirTree;

public class WindSpeedComparison implements AnyMechanismInterface
{
    private AnyMechanism mMechanism;
    Hurricane mHurricane;

    public WindSpeedComparison(AnyMechanism mechanism)
    {
        mMechanism = mechanism;
    }

    public void loadCriteria()
    {
        load();
    }

    public void setHurricaneInstance(Hurricane hurricane)
    {
        mHurricane = hurricane;
    }

    public AnyCategory runs()
    {
        Vector criteriaList = mMechanism.getCriteriaList();
        for (int i = 0; i < criteriaList.size(); ++i)
        {
            WindSpeedCriteria criteria = (WindSpeedCriteria) criteriaList.get(i);
            if (criteria.matchCriteria(mHurricane.getWindSpeed()))
            {
                return criteria.getCategoryForCriteria();
            }
        }
    }
}

```

```

    }

    return null;
}

private void load()
{
    try
    {
        File file = new File(DirTree.CONF_PATH);
        File[] fileList = file.listFiles();

        if (fileList == null)
        {
            return;
        }
        for (int k = 0; k < fileList.length; k++)
        {
            HurricaneCategory data = new HurricaneCategory();
            data.readData(fileList[k].getPath());
            data.printData();

            long minSpeed = data.getMinimumWindSpeed();
            long maxSpeed = data.getMaximumWindSpeed();
            System.out.println("minSpeed = " + minSpeed);
            System.out.println("maxSpeed = " + maxSpeed);

            WindSpeedCriteria criteria = new WindSpeedCriteria(minSpeed,
                                                                maxSpeed);
            criteria.attachCategory(data.getCategory());
            mMechanism.getCriteriaList().add(criteria);

        }
    }

    catch (Exception ex)
    {
        JOptionPane.showMessageDialog(null,
            "Error reading directory " +
            DirTree.CONF_PATH,
            DirTree.APP_NAME,
            JOptionPane.WARNING_MESSAGE);
    }
}

```



```

    }
}
}

```

```

package com.udme.IOs;

```

```

/**
 * <p>Title: WindSpeedCriteria</p>
 *
 * <p>Description: Describes WindSpeedCriteria</p>
 *
 * <p>Copyright: Copyright (c) 2007</p>
 * @author Somenath Das
 * @version 1.0
 */

```

```

import com.udme.patterns.Classification.AnyCriteria;
import com.udme.patterns.Classification.AnyEntity;

```

```

public class WindSpeedCriteria extends AnyCriteria
{

```

```

    private long mMinimumWindSpeed;
    private long mMaximumWindSpeed;

```

```

    public String getName()
    {
        return "WindSpeedCriteria";
    }

```

```

    public WindSpeedCriteria(long min, long max)
    {
        mMinimumWindSpeed = min;
        mMaximumWindSpeed = max;
    }

```

```

    public long getMinimumWindSpeed()
    {
        return mMinimumWindSpeed;
    }

```

```

public void setMinimumWindSpeed(long data)
{
    mMinimumWindSpeed = data;
}

public long getMaximumWindSpeed()
{
    return mMaximumWindSpeed;
}

public void setMaximumWindSpeed(long data)
{
    mMaximumWindSpeed = data;
}

public boolean matchCriteria(long speed)
{
    if (speed >= mMinimumWindSpeed && speed <= mMaximumWindSpeed)
    {
        return true;
    }
    else
    {
        return false;
    }
}
}

```

Appendix E: Demo & instructions

E.1 Setup requirements

Installing the java development kit (JDK)

The application needs JDK 1.5 or later versions. The latest version can be downloaded from the Sun website <http://java.sun.com/j2se/1.5/download.html>. After the installation, the environment variable PATH should include the JAVA_HOME/bin directory.

Running the application

To run the application double clicks the run.bat file under the directory
ClassificationPattern\Code\Demos\HurricaneClassificationApp\run.

Demo instruction and screenshots

The main screen of the application has two buttons as shown in Figure 54.

Figure 54 - Main application frame

To define the category press the “Define Category” button which open the category window where any new category can be added or modified as shown in Figure 55.

Figure 55 - Category edit menu

To enter the data related to hurricane press the “Classify Hurricane” button in Figure 54, which opens a new window to add or modify data related to any hurricane as shown in Figure 56.

Figure 56 - Hurricane data frame

Initially the CategoryName in the hurricane screen shows as “Not Categorized” to classify the hurricane press the “Classify” button, which classify the data, based on the categories defined as shown in Figure 57.

Figure 57 - Hurricane screen after the data is classified

May 14, 2007

Dr. Mohamed Fayad
Professor of Computer Engineering
Computer Engineering Dept.,
College of Engineering
San José State University

Dear Sir:

I am completing a masters dissertation at San Jose State University entitled "Unified Data Mining Engine as a System of Patterns" I would like your permission to reprint in my dissertation excerpts from the following:

Fayad's pattern archive which contains several pattern documents, essays and articles based on Software Stability Modeling.

The excerpts to be reproduced are: See the attached copy

The requested permission extends to any future revisions and editions of my dissertation, including non-exclusive world rights in all languages, and to the prospective publication of my dissertation by UMI. These rights will in no way restrict republication of the material in any other form by you or by others authorized by you. Your signing of this letter will also confirm that you own the copyright to the above-described material.

If these arrangements meet with your approval, please sign this letter where indicated below and return it to me in the enclosed return envelope. Thank you very much.

Sincerely,

Somenath Das

PERMISSION GRANTED FOR THE
USE REQUESTED ABOVE:

Dr. Mohamed Fayad

Date: 6/3/07

Class Diagram for:

- 1. Discovery Analysis Pattern**
- 2. Classification Analysis Pattern**
- 3. Visualization Analysis Pattern**
- 4. AnyInteraction Design Pattern**
- 5. AnyPrecision Design Pattern**
- 6. AnyModel Design Pattern**
- 7. AnyCriteria Design Pattern**
- 8. AnyData Design Pattern**