International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN : 2278-800X, www.ijerd.com
Volume 5, Issue 8 (January 2013), PP. 01-08

Unified Pattern Repository (UPR) for Design pattern Selection

S S Suresh[1], Dr.Kameswara Rao[2], Dr.Sagar S. Jambhorkar[3]
International Institute of Information Technology, Pune, India
Associate professor, K.L.University, Vaddeswaram, Vijayawada
National Defence Academy, Pune-23 (MS) India.

Abstract:- Design pattern selection is a difficult task. Evaluating the text based problem statement and finding the pattern is a challenging task. Generally, improper text evaluation leads to incorrect pattern selection. The current paper describes a text support model (XML) for design pattern organization called Unified Pattern Repository (UPR), related work of pattern repositories, pattern indexing process, and pattern retrieval.

Keywords:- Design pattern, Inverted Index, Lucene index, JAXB, Document and pattern retrieval, and XML

I. INTRODUCTION

A pattern is a common solution to a recurring problem in a software design [1]. Frequently,software developers use design patterns for development. Basically, the concept of design pattern evolved from software design principles (e.g. Coupling, Cohesion etc.) [9]. The foundation for patterns laid by Christopher Alexander [1][2]. His work had inspired GoF (E.Gamma, R.Helm, Ralpha Johnson and J.Vlissides) for developing software-related patterns. GoF described 23 design patterns and published them in UML notations (Unified Modelling Language). The GoF described design pattern as "Descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context". In the fall of 1994, the Eric Gamma et al., developed pattern catalog containing Creational, Structural and Behavioural patterns [12]. The catalog was part of his Ph.D thesis [12]. The Patterns of Software Architecture [POSA] defined patterns into several levels of abstraction, from architecture to design patterns to low-level idioms. The POSA had grouped patterns based on criterias such as interaction and adoptable systems, organization of work, communication and access control [17]. For decade, pattern have been remaining as software engineering problem-solving discipline that emerged from the object-oriented community [21]. Patterns have roots in many disciplines, including literate programming, and most notably in Alexander's work on urban planning and building architecture (Alexander, 1977).
This approach was named as "Pattern-Oriented Software Architecture". The definitions of these three level patterns as per POSA are as follows:

Architectural pattern: An architectural pattern describes a fundamental structural schema for software systems. It provides a set of predefined subsystems specifies their responsibilities, includes rules and guidelines for organizing the relationships between them [17].
Design pattern: A design pattern describes commonly-recurring structure of communicating components that solve general design problem in a particular context [17].
Idiom: An idiom is a low-level pattern, specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language [17].

Design pattern selection remains a challenging task. Alexandrian states that a pattern description mainly contains three parts (PCS): problem statement, context and solution ('P' represents problem statement, 'C' represents context and 'S' represents solution). The PCS represents a template for pattern description. Problem statement describes a statement about the problem that a pattern solves. Context describes a situation in which the problem occurs. Solution describes solution or technique for solving the problem. Every design pattern describes PCS. Developer must understand the logical connection between P, C, and S. Evaluating the problem statement (P) is important task for pattern selection. Problem statement (p) consists of collection of statements. Each statement is a collection of words. A word can be a keyword or non-keyword. Technically, extracting keyword from the problem statement and then identifying the pattern makes the pattern selection easy. The proposed model is designed based on the keywords.

The remaining section is organized as follows:
Section II describes block diagram of pattern repository, section III describes UPR description and organization, section IV describes related work, and section V describes Indexing and retrieval, followed by conclusion.

Pattern Repository

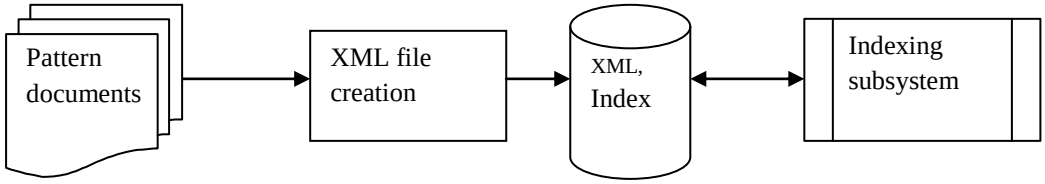

Fig. 1.1 UPR block diagram

The fig 1.1 shows the block diagram of the UPR under consideration. A designer documents the description of a design pattern using one of the design pattern description templates. There are at least six most widely used templates namely Alexandrian form, Pattern catalog form, POSA [Pattern Oriented Software Architecture] form, Portland form, Coplin form, and P of FF form. As part of the study, a Unified Document Repository (UPR) for Design patterns is proposed to store different types of patterns under single schema. Which facilitates information retrieval (IR) based pattern retrieval easier. A design pattern collected from external source is transformed into unified pattern description template. The pattern description in Unified design pattern description template are stored as XML files using text editor like notepad. The standard components of Java technology namely data binding, un-marshal and Lucene are used for creating indexes of XML files. The selection process includes the Lucene IR process for evaluating the problem text.

Further, Indexing subsystem extracts text from xml files and index the text and stores in the central repository. The index process uses Lucene java technology for creating the index. The Lucene IR process analyzes the problem text and generates terms. Subsequently, the terms are added to the index. The index subsystem automatically calculates term statistics like term frequency (tf), document frequency (df) and inverse document frequency (idf) and stores in index files [15].

II. UNIFIED PATTERN REPOSITORY

Design Patterns are organized in XML format. XML is the defacto standard for developing World Wide Web (WWW) applications.XML has wide variety of uses . Important uses are; knowledge representation, and data encoding & transfer.. However, different techniques are available for representing the knowledge. Rule-based approach is the popular one [38] [8][18][11]. In the current work, the pattern text is organized in a structured way to facilitate better information retrieval [34]. To improve the retrieval performance, key words are included in XML document. Patterns are classified based on various factors. For example, GRASP (General Responsibility Assignment Software Patterns) describes principles for assigning responsibilities to classes or objects [11]. GRASP emphasis on object responsibilities. Architects identify object responsibilities (responsibility represents operation of an object) in requirement analysis. Further, requirements are documented using use cases (A use case represents steps required for achieving a function/task). The assignment of responsibilities to objects involves design principles. The GRASP design principles are: Low-coupling, high-cohesion, polymorphism, protected variance, controller, creator, pure fabrication and information expert [11]. Developers should be aware of these principles before choosing a pattern in GRASP category.

The authors Gamma et al [12] Classifies patterns based on ‘Purpose’ and 'Scope’. The purpose describes what a pattern does. Three categories of purpose are defined. They are Creational, Structural and Behavioural. A creational pattern describes the process of object creation. Structural pattern describes the composition of classes or objects. Behavioural Patterns describes object interactions and their responsibilities. Developers should know about what each pattern does before selecting a design pattern. The scope specifies whether a pattern applies primarily to classes or to objects. There are two categories of scope: Class and Object. GoF (Gang-of-Four) authors had created a template form for documenting the design patterns [10]. In literature different templates existed [5]. Each template has granularity, purpose, advantage and disadvantage. Due to wide variety of pattern templates, pattern identification becoming difficult. No template form is standardized. The table 2.1 shows popular template forms and their elements.

Table 2.1 Pattern Template Forms

Alexandrian	**GoF**	**POSA**	**Portland**	**Coplin form**	**P of FF form**
Title, Prologue, **Problem statement,** Discussion, Solution, Diagram, Epilogue.	**Intent,** Motivation, **Applicability,** Structure, **Participants, Collaborations Consequences,** Implementation Sample code **Known Uses, and Related Patterns.**	Summary Example **Context, Problem, Solution,** Structure, Dynamics, Implementation, Example resolved, **Variants, known uses, Consequences,** and see also.	**problem, Context and forces.**	**Problem, Context, Forces, and Solution**.	How it works, **When to use it,** and one or more examples.

The table 2.1 includes the Alexandrian form, Pattern catalog form, POSA [Pattern Oriented Software Architecture] form, Portland form, Coplin form, and P of FF form [5]. The purpose of each template form is as follows:

The Alexandrian form proposed by Alexander in the year 1979. The purpose of the Alexandrian form is to guide users to generate solutions for the problems [11]. The second form is pattern catalog form proposed by Gang-of-Four (GoF). The purpose of this form is to help users to create solutions to problems but less focused on when to apply the pattern [5]. The pattern catalog form express structure and dynamics of the pattern [5]. It has many elements. However, it gives detailed understanding about the pattern from problem statement to related patterns. To understand and apply GoF patterns, developers should have knowledge in object oriented programming (preferably C++). The third form is POSA. It is a structured form. POSA form is a lengthy form. It has less focus on applicability's, forces, programme code, and structure of a pattern.The fourth form is Portland form. The Portland form is descriptive and short. It has not focused on forces, applicability's, and consequences, when to use a pattern, structure, and implementation [5] The fifth one is coupling form. It is also called Canonical form. It describes patterns in a short form [5]. It has less focus on applicability's, consequences, when to use a pattern, structure, and implementation P off FF form. The P of FF form is a narrative form with few sections [5]. From these template forms, it is observed that many elements create difficulty for pattern selection. According to authors view some elements are important. Based on the pattern selection criteria's, a rank is given to each element. Lower the rank higher the selection criteria. The table 2.2 gives elements and their ranks in ascending order.

Element	**Rank**
Problem statement	1
Context	2
Force	3
Solution	4
Applicability's	5
Consequences	6
Known uses	7
Structure	8
Related pattern	9
Implementation	10
Examples	11
Prologue	12
Epilogue	13

Table 2.2 Pattern Element Ranks

The elements having ranks 1-7 are important for pattern selection. Based on the first 1-7 ranks, a repository is designed and is named as Unified Pattern Document Repository (UPR). The UPR facilitates search operation for retrieving suitable patterns for the problem.

UPR Description:

The repository contains two types of elements namely standard elements and user defined elements. Standard elements includes: name of the pattern, type of a pattern, intent, problem statement, forces, consequence, known uses, variations of the pattern, version number of the pattern, functional domain, author of a pattern, related pattern, applicability. The UPR does not focus on structure of a pattern and implementation of a pattern. Because, the goal of this repository is to provide suitable pattern to the user problem or scenario when problem statement is given in text format. Brief description about each of the element are given the table 2.3.

Table 2.3 UPR Standard Elements

Element	Description
Name	The name signifies name of a pattern. Name is useful to know the scope and purpose of a pattern.
Type	The type signifies the category in which a pattern belongs (e.g algorithm design pattern, execution pattern, computational pattern etc).
Intent	The intent signifies pattern intention and tells what a pattern does.
Problem statement	The problem statement signifies the situation in which a pattern can be applied
Force	A scenario consisting of a problem and a context in which this pattern can be used
Applicability	Applicability signifies situations in which a pattern is usable
Consequences	Consequence signifies tradeoffs and side effects of pattern.
Known use	A known use signifies real usages of the pattern
Author	The Author signifies the author of a pattern.
Related Pattern	. Related pattern signifies other patterns that have relationship
Definite Keyword	keyword signifies a word which makes a system easy for retrieval of a pattern
Functional domain	It explains the list of functional domains in which patterns are already used.

The user defined elements includes definite keyword, structural questions and generic questions. Brief description about each of the element are given in the following table 2.4.

Table 2.4 UPR user defined elements

Definite Keyword	Keyword signifies most important words through which the search system retrieves pattern documents quickly without performing text parsing.
Generic Question	Generic Questions are the questions related to a pattern. The questions reflect charectertics of a pattern. Questions can be single choice, multiple choices; fill in the blank and true/false type. A question requires answer and a question may have dependant question. Each question is given alphanumeric code (Ex: CQ01).
Structural question	Structural question indicates a question related to a structure of a pattern. The structural question emphasis on structure of a pattern. It improves pattern identification and strength the pattern selection.
Answer type	There are two types of answer type included. There are Boolean or Text.

2.2 Conceptual Model

The conceptual model represents different elements and their relationships [6]. The reason is, most of the patterns are organized around these templates. Hence, this model not only supports GoF patterns but can also support other category of patterns (e.g anti patterns, GRASP patterns etc.). The XML repository of this work and its organization of pattern elements is shown in the fig 3.1 using Unified Modelling Language notations (UML) [9].

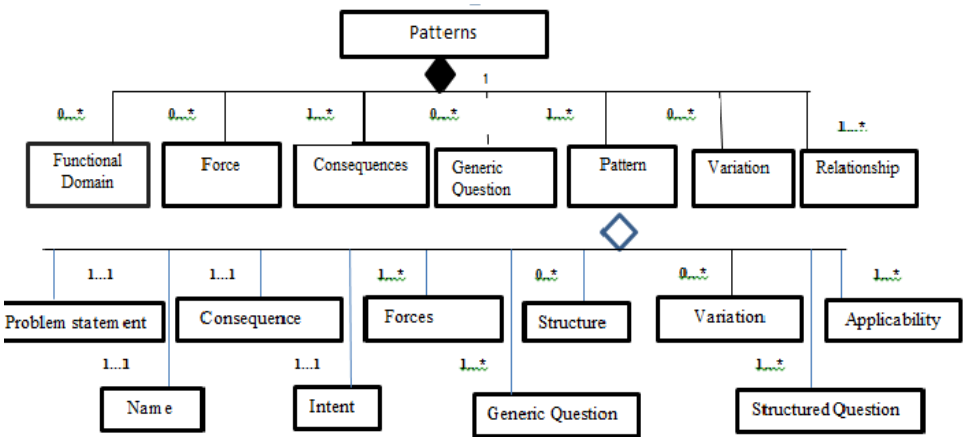

Fig 3.1 Conceptual diagram

The pattern repository is implemented in XML format. The repository is an aggregation of 7 complex elements. Each element is part of pattern representations. The description about each of the 7 elements is as follows:

a) **<Functional Domain>**: The functional domain indicates the functional areas where a pattern has already been used. Each functional domain can be identified with a name and summary about that domain.
b) **<Forces>** Every pattern has to balance certain forces. Each force is described with a summary description and set of keywords. The keywords are indexed using Lucene's search API and stored in Index database. The pattern search operation access the index database for performing search operation.
c) **<Consequences>:** Consequences are the effects on the problem after applying the pattern. There can be more than one consequence for a pattern. Each consequence has summary description and keywords for indexing consequence type.
d) **<Generic Question>**: Each pattern contains pattern related questions, which should be answered by the user while selecting a pattern. There should be at least one generic question for every pattern.
e) **<Pattern>:** Each pattern data is organized in the form a template. In the pattern community, there are two popular authors, Christopher Alexander and Gang-of-four. The structured template that is followed in this work contains elements stated by both the authors. To enhance the pattern search, 4 additional elements (Authors, Definite keyword, Generic question and Structure question) have been added. Finally, the elements in the template are as follows: Name , Intent, Problem statement, Force, Consequences, Known uses, Structure, Variation, Version, Reference, Functional domain, Author, Related pattern, Applicability, Definite keyword, Generic question and Structure question.
f) **<Variation>:** Every pattern contains variation. It contains two elements, name and summary.
g) **<Related pattern>**: Every pattern has relationship with other patterns. Related type and its relationship is mentioned in this element. This is useful for expanding the search operation.
h) **<Structural question>** this element contains a question related to the structure of the pattern.

2.3 Advantages of the UPR

a) Accommodates various categories of (e.g., GRASP, Anti patterns etc) patterns to store.
b) New elements can be easily added and they are extendable.
c) It supports interoperability and platform independence.
d) Web services can be included without altering the underlying schema.
e) Indexing and retrieval becomes easy.

III. RELATED WORK

There are many online catalog's and blogs available describing various categories of patterns in the internet [60]. However, the following repositories have been observed.

- **Net Objectives Design pattern repository**: This repository is sponsored by Net Objectives. It says, "They are on to training, coaching, and consulting on software design, agile methodologies, test-driven development, lean software process, and scrum" [18]. The repository is free for users for study purpose. The site maintains design pattern information in the form of HTML files. The site has provision for discussion group. The site author provides answers for user queries.
- **"Portland pattern repository (PPR):** "Portland Pattern Repository was set up for documentation of design patterns. The PPR is a repository for computer programming design patterns. It was accompanied

by a companion website, WikiWikiWeb, which was the World's first wiki. The repository has an emphasis on Extreme Programming. It is hosted by Cunningham & Cunningham (C2) of Portland, Oregon at http://c2.com/ppr/" [19].

- **KONIGI, design pattern repositories**: "This is a list of resources for design pattern research. The list contains books and web sites with formal design pattern libraries that include categorized patterns with problem statement and description of solution" [20].
- **The Hillside Group** is important resource for sharing design patterns information. The Hillside Group promotes the use of patterns and pattern languages to record, analyze, and improve software and its development. It maintains pattern catalogs, and sponsors a variety of activities to achieve mission-organizing workshops, conferences, and publications for discussing, recording, and documenting successful software practices [21].

The current work is no way comparable to these catalogs and blogs.

IV. PATTERN INDEX PROCESS

In the whole work, index is an important component for pattern retrieval. Without index component it would be difficult to retrieve the patterns based on problem statement. Index is a data structure which contains pattern records and their addresses. For text based retrieval, inverted-index is one of the best choices. The current work uses Lucene's inbuilt index features. Pattern retrieval is depended on the index. The process of indexing is described in two parts: part-I explains pre-process steps for indexing. Part-II explains index structure and retrieval process.

Part-I: Pre-process for index

Acquire content is the first step of the Index process. Without suitable content, Index will be useless. Raw pattern text is collected from freely available material [15]. Mislleneous data is removed from it manually. The processed text is stored in the XML file. XML file must confirm to UPR schema format. A single large xml file is created to store the entire 23 design pattern. Suitable xml content must be extracted. Extraction requires Parsing. Parsing involves several overheads. For example, DOM (Document Object Model) follows tree based approach [15]. The result of parsing an xml file is represented as an object and stores in primary memory. It is memory intensive [22]. The developer must know the structure of xml file to access the content from it. Hence, to reduce the parsing overhead, JAXB (Java XML binding) technology is used. JAXB does not require parsing. It requires data binding. So, the overhead of parsing is eliminated. Using JAXB technology an xml file converts into collection of java objects using un-marshalling process [22]. Further each java object wraps into a Lucene document object. A Lucene document object contains xml fields that are to be indexed and its corresponding values. At present the UPR contains 23 patterns. For each pattern, the indexable elements are: problem statement, intent, applicability, and forces. The following code snippet shows how each pattern is added to Lucene document object.

(1) **For I =1 to n** {
(2) Document.problemstament=pattern (i).xml. Problemstatement;
(3) Document.intent= pattern (i).xml. intent;
(4) Document.appilcability=pattern (i).xml. applicability;
(5) Document.forces= pattern (i).xml. forces;
(6) Document.patternname=Pattern(i).patternname';
(7) Document.Path_of the Pattern=Pattern (i).path'; }

GoF describes 23 design patterns. Hence, 23 documents are created and subsequently added to the segment. When a query (called problem statement) occurs the system process the query and searches for the pattern against Lucene document objects.

Part-II: Index structure: Lucene facilitates search based on inverse-index model. The inverse-index files and their relations are shown in the fig 3. Terms are stored in a file called .fnm file. For each .fnm file, a corresponding .tis file is maintained. This .tis file contains fields and values and document frequencies. A document frequency signifies, how many documents contain the given term. For each term in the .tis file, corresponding term frequencies are stored in the .trf file. For each document in the .trf file, the position of the term in the document is stored in the .prx file. The files, .fnm, .tis, .trm and .prx are index files, which are created and maintained in operating system [15].

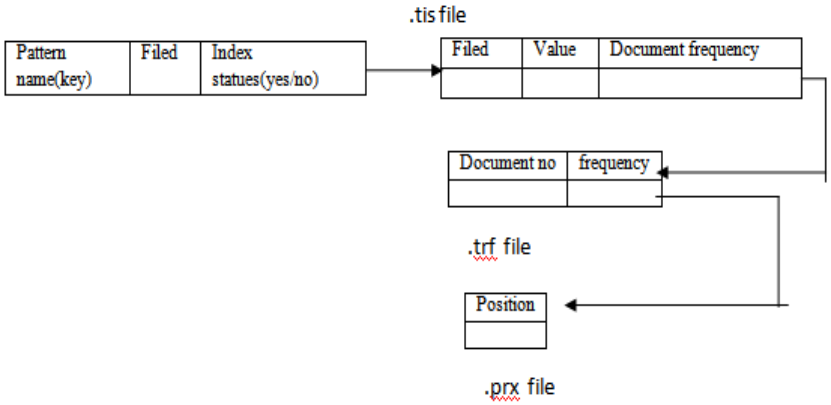

Fig. 4.1 Lucene Inverse-Index structures

The term frequency and document frequency are used for calculating the document matching score [15][9]. Document score is a floating point number greater than 0.0 Higher the score, greater the relevance. An Index is created for each segment. Lucene's Index contains the following information:

- **Field names**. This contains the set of field names used in the index.
- **Stored Field values.** This contains, for each document, a list of attribute-value pairs, where the attributes are field names. These are used to store auxiliary information about the document, such as its title, url, or an identifier to access a database. The set of stored fields are what is returned for each hit when searching. This is keyed by document number.
- **Term dictionary**. A dictionary containing all of the terms used in all of the indexed fields of all of the documents. The dictionary also contains the number of documents which contain the term and pointers to the term's frequency and proximity data.
- **Term Frequency data**. For each term in the dictionary, the numbers of all the documents that contain that term, and the frequency of the term in that document if omit Tf is false.
- **Term Proximity data**. For each term in the dictionary, the positions that the term occurs in each document. Note that this will not exist if all fields in all documents set omit Tf to true.
- **Normalization factors**. For each field in each document, a value is stored that is multiplied into the score for hits on that field.
- **Term Vectors**. For each field in each document, the term vector (sometimes called document vector) may be stored. A term vector consists of term text and term frequency. To add Term Vectors to your index see the Field constructors.
- **Deleted documents**. An optional file indicating which documents are deleted.

V. CONCLUSIONS

In the current work, the system has some limitations; collecting external data and subsequently preparing the xml files is manual task. Manual operations delay the development and may enforce errors. Automation of the xml file creation process may reduce development time. The system requires keywords for effective information retrieval. Improper identification of keywords leads to inaccurate results. Experienced users can identify keywords. Automatic identification of keywords from user queries will improve the search performance. In future the repository will be extended with more pattern categories.

ACKNOWLEDGEMENTS

I, thank Chakradhar Yadavalli and Surya Phani kiran for their valuable support and suggestions for creating this repository.

BIBLIOGRAPHY AND REFERENCES

[1]. Albit-Amiot, H.P.Cointe, Y.G. Guehenuue, and N.Jussien, 2001, November. Instantiating and detecting design patterns: Putting bits and pieces ogether. In 6 th annual International Conference on Automated software Engineering (ASE), 166-173.

[2]. A. Sengupta, S. Mohan, R. Doshi. XER - Extensible Entity Relationship Modelling. In Proceedings of the XML 2003 Conference, p. 140-154. Philadelphia, USA, December 2003.

[3]. Bernadette Farias Lóscio, Ana Carolina Salgado, Luciano do Rêgo Galvão. "Conceptual Modelling of XML Schemas", Proceedings of the fifth ACM international Work shop on Web information and data management (WIDM03), 102-105, ACM Press, 2003

[4]. David C. Kung, Hitesh Bhambhani, Riken Shah and Gaurav Pancholi, an Expert System for Suggesting Design Patterns— A Methodology and a Prototype, 2005.

[5]. Dirk Reihle and Heinz Zullighoven, "Understanding and using patterns in Software Development", Theory and practices of Object systems, 2, 1996

[6]. D.W.Embley, S.W.Liddle,R.Al-Kamha. Enterprise Modeling with Conceptual XML. In Proceedings of the 23rd International Conference on Conceptual Modeling(ER 2004), p 150- 165. Shanghai, China, November 2004.

[7]. G. Psaila. ERX: A Conceptual Model for XML Documents. In Proceedings of the 2000ACM Symposium on Applied Computing, p. 898-903. Como, Italy, March 2000.

[8]. S.S.Suresh, Dr.M.M.Naidu," Design Pattern Recommendation system, (Methodology, Data model, Algorithms) Proceedings of International Conference on Computational Technique and Artificial Intelligence (ICCTAI'2011), Planetary Scientific Research Center, Oct 7-8 Pattaya, Thailand. ISBNNO:

[9]. S.S.Suresh, Dr.M.M.Naidu," An XML based Knowledge-Driven Decision Support System For Design Pattern Selection**,** International Journal of Research in Engineering and Technology (IJRET), Vol 1. December, 2011. ISSNNO: 2277-4378,

[10]. Tichy W. F., "A Catalogue of General-Purpose Design Patterns", In Proceedings of Technology of Object-Oriented Languages and Systems (TOOLS 23), IEEE Computer Society, 1998. pp. 330-339.

[11]. Craig Larman, "Applying UML and Patterns," Prentice Hall, 1998.ISBNN0:

[12]. Gamma E., et al. "Design Patterns: Elements of Reusable Object-Oriente Software," Addison-Wesley, 1995.ISBNNO

[13]. Jeffery A. Hoffer et al,"Modern Database Management", Sixth edition, Pearson Education, Low price Edition, ISBNNO: 81-7808-804-5.

[14]. L. Rising, "The pattern Almanac 2000", Addition Wesley publications, 2000, .ISBNNO: 0201379678.

[15]. Erik Hatcher and Otis Gospodnetić, "Lucene in Action", Manning 2004 publications, Second edition, ISBN: 1932394281

[16]. Fowler, Martin (2002). Patterns of Enterprise Application Architecture. Addison-Wesley. ISBN 978-0321127426.

[17]. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Summerland and Michael Stal. Pattern-Oriented Software Architecture: A System of Patterns, Volume1, Wiley publications, 2007.ISBNNO:

[18]. Scott Brain,"Net Objectives Design Pattern repository", [online] available at http://www.netobjectivestest.com/PatternRepository/index.php?title=Main_Page,last modified 27 June 2012.

[19]. Cunningham & Cunningham,"Port land Pattern repository", [online] available at http://c2.com/ppr last accessed 19 12 2012.

[20]. Michael Angeles KONIGI, "Design pattern repositories", [online] available at http://konigi.com/wiki/design- pattern- repositories, last edited 17 November 2011.

[21]. "The Hillside Group, [online] available at http://hillside.net

[22]. Daniel Steinberg, Data binding with JAXB, mapping XML to Java Objects, and vice-versa, 20 May, 2033, IBM Corporation, 2003.