Webpage: www.ijaret.org Volume 3, Issue VI, June 2015
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS…..

COMPONENT BASED SOFTWARE ENGINEERING USING UML DIAGRAMS

Palak Wadhwa[1], Manisha Gahlot[2]

[1,2]Department of Computer Science & Engineering
South Point Institute of Technology & Management
DCRUST, Murthal, India
[1]palakwadhwa02@gmail.com

Abstract*: The growing interest in CBSE is reflected in the number of workshops and conferences with CBSE tracks. CBSE refers to using software modules that were developed on a previous software project as part of a new software development project. CBSE is a worthwhile goal since it has shown to reduce software costs, and improve software quality as well as programmers productivity. Software components have played an important role in modern software and system development. The main contribution of software components is reuse which helps reduces development cost and time, and increase productivity. While libraries of Unified Modeling Language (UML) diagrams and source codes do exist, one of the challenges that still remain is to locate suitable designs and source codes, and adapt them to meet the specific requirements of the software designer. The class diagram contains valuable information about the structural description and contents of a class, i.e. class name, attributes, behavior, relationships, generalization etc. These attributes can be used for specification matching with the contents of the repository. The Use case diagram contains valuable information about the requirements specification of software. These include use cases and actors. If we search the repository on the basis of attributes, the search result would be better and thus giving higher precision, as compared to keyword based search. Moreover if we assign some numeric values or weights to different contents of a class, and arrange the search results in ascending or descending order, we would be able to find out the precision of the components in order of percentage match. Hence the role of user to find the best-fit component from the search results would be much easier. In this work, we develop a tool named as Component Based Search Engine, which assists the software designers in the retrieval of the designs as well as source codes.*

Keywords: *Component based Search, CBSE, Component based development.*

1. INTRODUCTION

CBSE addresses challenges and problems similar to those encountered elsewhere in software engineering. There is however one difference; CBSE specifically focuses on questions related to components and in that sense it distinguishes the process of “component development” from that of “system development with components”. There is a difference in requirements and business ideas in these two cases and different approaches are necessary [1]. Components are built to be used and reused in many applications, some possibly not yet existing, in some possibly unforeseen way. Marketing factors play an important role, as development costs must be recovered from future earnings, this being especially true for COTS. System development with components is focused on the identification of reusable entities and relations between them, beginning from the system requirements and from the availability of components already existing. A standard approach in a system development is a top down approach; the system design starts with specification of system architecture. The architecture defines the components of a software system as well as their interactions and can be used to analyze its quality attributes. The system design process typically consists of three phases, which might be passed in several iterations. The first phase includes functionality-based design (i.e. a design of the software architecture based on the functional requirements). Although software designers generally will not design a system without concern nonfunctional requirements, these are not explicitly addressed at this stage [2]. Functionality-based design consists of four steps: defining the boundaries and context of the system, identification of archetypes, decomposition of the system into its main components and, finally, the first validation of the architecture by describing a number of system instances). The second phase is the assessment of the quality attributes of software architecture in relation to the quality requirements. The third phase of the software architecture design process is concerned with transformation of design solutions to improve the quality attributes while preserving the domain functionality captured by the software architecture. These transformations result in a new version of the software architecture which in general improves one or some quality attributes while they affect others negatively [3]. The final result of this stage is a system software architecture which identifies components and interactions between them. Up to now the design model is not specific for component-based approach. In a "classical" approach the next step would be to implement the components identified by the design. In a component-based approach the main idea is re-use already existing components, i.e. to find the most suitable components. The implementation effort in system development will decrease but the effort required in dealing with components; locating them, selecting those most appropriate, testing them, etc. will increase [4].

2. COMPONENT BASED DEVELOPMENT

The types of components that a system is composed of influence the architecture of the system. In a similar way as the framework into which components are to be plugged influences architecture of the system and, the type of components selected, influences the system design process. Design freedom is limited to component selection and the way the selected components are integrated. This restricted freedom points out the importance of managing and controlling component integration. Component specifications in from of API do not normally provide enough information about how the component will behave when used in a given environment. This difficulty raises issues related to understanding and verifying both the functional and non-functional properties of the components so that the unexpected mismatches among components are avoided and overall system behavior can be accurately predicted. The verification of component properties is required in order for developers to have confidence that a system will behave as its architect predicted it would. In many cases a component property alone cannot be used to predict the system behavior, but clusters of components (assemblies) must be analyzed separately. The examples briefly mention above show that much of activity in the design and development phase belong to the component properties and composition issues [5]. This shows that a component-based design can have difficulties in using top-down approach. Rather a mix of a top-down and a bottom-up approach will occur. The growing complexity of software systems that play critical roles in our society has raised exceptionally challenging software development problems. Mission-critical software systems that must run on networks consisting of heterogeneous computing platforms, and must interact in complex ways with other systems, are especially difficult to develop. Examples include intelligent vehicles and transportation systems, air traffic control systems, and home healthcare systems that dynamically adapt to a home owner's assistive care needs. In addition to being functionally correct and usable, these systems must also be highly available and operate in a safe, secure, robust, and otherwise dependable manner [6]. Despite advances made in program development technologies, developing complex software using current technologies requires tremendous human effort. Software complexity is growing at a rate that is outpacing the rate of the introduction of necessary software development technologies. Significant insights into challenging software development problems are best obtained through extensive experimentation with proposed solutions. These insights enable the development of next generation technologies that better manage growing complexity. The current lack of shareable software development products significantly impairs our ability to perform the extensive experimentation needed to increase the rate at which new and effective software development technologies are produced.

Softwarereusereferstousingsoftwaremodulesthatwere developedonaprevious software project as part of a new software development project .Software reuse is a worthwhile goal since it has shown to reduce software costs, and improve software quality as well as programmers productivity [7]. Software components have played an important role in modern software and system development. The main contribution of software components is reuse which helps reduces development cost and time, and increase productivity. While libraries of Unified Modeling Language (UML) diagrams and source codes do exist, one of the challenges that still remain is to locate suitable designs and source codes, and adapt them to meet the specific requirements of the software designer.

3. TECHNIQUES OF COMPONENT RETRIEVAL

Traditional approaches to component retrieval are keyword-based; which resulted in the retrieval of many irrelevant components. A more promising approach is retrieval based on MDL format, where the contents of MDL file of the UML diagrams are matched to retrieve the components [8]. The UML models that are used for modeling are stored as MDL file format. These MDL file formats are semantically very information rich and contains lot of valuable information about the asset. The information can be structural as well as behavioral. The class diagram MDL file format contains valuable information about the structural description and contents of a class, i.e. class name, attributes, behavior, relationships, generalization etc. These attributes can be used for specification matching with the contents of the repository [9].The Use case diagram MDL file format contains valuable information about the requirements specification of software. These include use cases and actors. If we search the repository on the basis of attributes of MDL file descriptions, the search result would be better and thus giving higher precision, as compared to keyword based search. Moreover if we assign some numeric weights to different contents of a class, and arrange these architectures results in descending order, we would be able to find out the precision of the components in descending order of percentage match [10]. Hence the role of (re)user to find the best-fit component from the search results would be much easier. In this work, we described and implemented a tool named as Component Retrieval Search Engine, which assists the software designers in the retrieval of the designs as well as source codes.

4. METHODOLOGY

Step 1: To model UML Diagrams
To model UML diagrams some software for modeling is required. In this work, we can use software's such as Visual Paradigm or Rational Rose, which is the product of IBM Corporation. In Rational Rose/ Visual Paradigm all the diagrams of the UML can be modeled and stored in a single file of MDL file format. Hence various sample cases are taken and are modeled in Rational Rose.

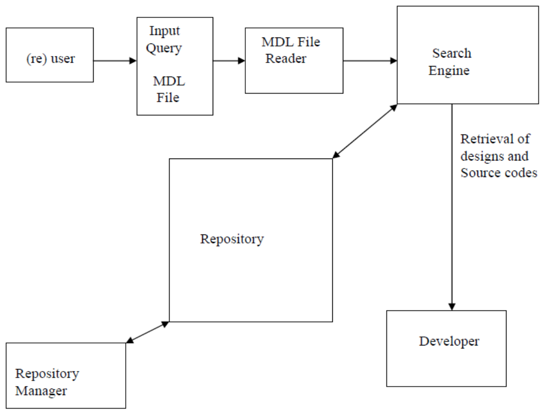


Figure 1: The Design of Component Retrieval Search Engine to handle Reuse Process.

Step 2: Storage of UML Diagrams and Java Code in Database
The repository should be indexed and classified according to the projects. Here, our repository includes two assets, namely source codes and designs. Designs are the diagrams, which are modeled in Rational Rose and stored in the Repository. Source codes consist of programs in java. Designs and Source Codes about a project are stored in the Repository.
Step 3: Development of Search Engine
The search engine should search upon the queries built to match the relevant components. The search input consists of MDL file. The output can be designs or source codes, as required by the user. The resultant should not only be most relevant matched items, but also some relevant matched items for browsing. Component Retrieval Search Engine is developed in C#. Net or VB.Net and MS Access. The repository can be changed or integrated with any other standard databases available like SQL Server or Oracle. For the reporting purposes the data reports of VB.Net are used. The input provided to search engine is modeled in Rational Rose to produce MDL file.
Step 4: Input to the Search Engine
sThe input provided to search engine is modeled in Rational Rose to produce MDL file. The search engine should search upon the queries built to match the relevant components. The search input consists of MDL file. The output can be designs or source codes, as required by the user. The resultant should not only be most relevant matched items, but also some relevant matched items for browsing.
Step 5: Retrieving the Results
The search engine should search upon the queries built to match the relevant components. The search input consists of MDL file. The output can be designs or source codes, as required by the user. The resultant should not only be most relevant matched items, but also some relevant matched items for browsing.

4.1 CASE 1MDLFILE–CLASS DIAGRAM SEARCH

Figure 2 shows the Main Interface of Component Retrieval Search Engine. Fig 3shows the Interface for providing search input, which is class diagram MDL File. Fig. 4 shows the interface for the overall search process. Figure5 shows the result set for search of Class Diagrams and Source Codes from MDL file.

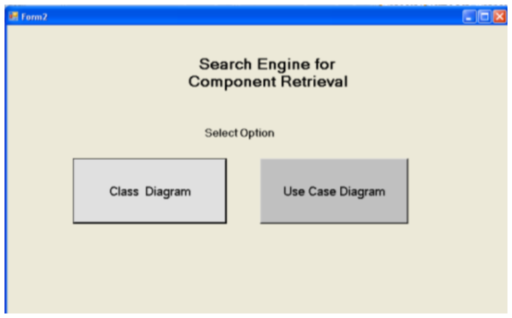


Figure 2: Main Interface of Component Retrieval Search Engine

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

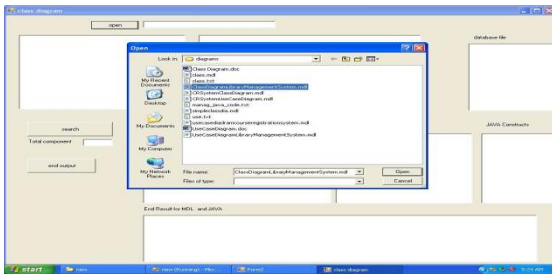


Figure 3: Interface for providing search input, which is class diagram MDL File

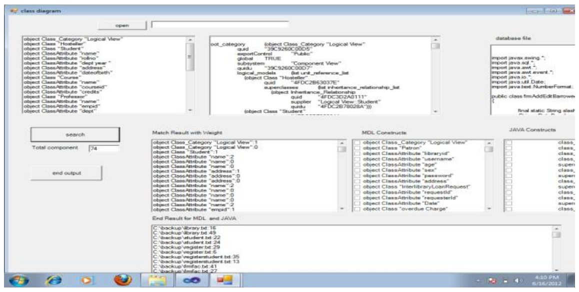
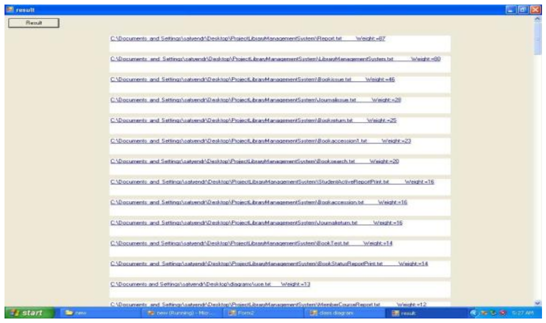


Figure 4: Interface for the overall search process



C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Report.txt Weight:=67
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\LibraryManagementSystem.txt Weight:=60
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Bookissue.txt Weight:=46
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Journalissue.txt Weight:=28
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Bookreturn.txt Weight:=25
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Bookaccession1.txt Weight:=23
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Booksearch.txt Weight:=20
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\StudentActiveReportPrint.txt Weight:=16
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Bookaccession.txt Weight:=16
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\Journalreturn.txt Weight:=16
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\BookTest.txt Weight:=14
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\BookStatusReportPrint.txt Weight:=14
C:\Documents and Settings\satyendr\Desktop\diagrams\use.txt Weight:=13
C:\Documents and Settings\satyendr\Desktop\ProjectLibraryManagementSystem\MemberCourseReport.txt Weight:=12

Figure 5: Result set for search of Class Diagrams and Source Codes from MDL file

4.2 CASE2 MDL FILE–USE CASE DIAGRAM SEARCH

The input provided is the MDL file. We model a similar type of one use case diagram of Library Management System in Rational Rose and store in MDL File. Now this MDL is set as input to the Search Engine.

4.2.1 Extracted Part of MDL File

(object Petal root_usecase_package (object Class_Category" UseCase View"
(objectClass "Library Staff" (object Class "Resource" (object Class "Patron"
(objectUseCase "Check InResource" (object Use Case "Manage Resource" (object Use Case "Check OutResource

4.2.2 Keywords extracted by MDL File Reader

Class: Library Staff, Resource, Patron
UseCase: Check InResource, Manage Resource, Check OutResource

As per the diagram modeled in MDL file these architectures results are shown in Fig 8.As we can see that the right components i.e. MDL files was selected and shown back along with their respective weights i.e. percentage match by the search engine.
Fig 6 shows the Main Interface of Component Retrieval Search Engine. Fig 7 shows the use case diagram interface for providing search input, which is use case diagram MDL File. Fig 8 shows the interface for the overall search process. Fig 9 shows the result set for search of Use Case Diagrams from MDL file.

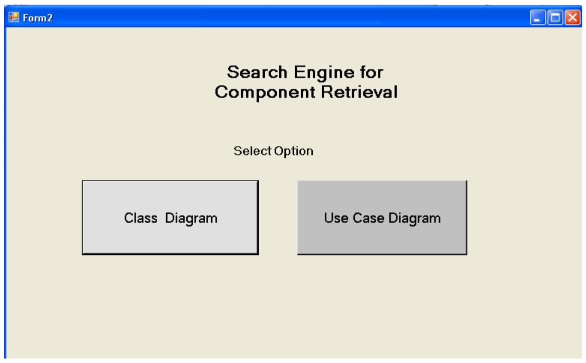


Figure 6: Main Interface of Component Retrieval Search Engine

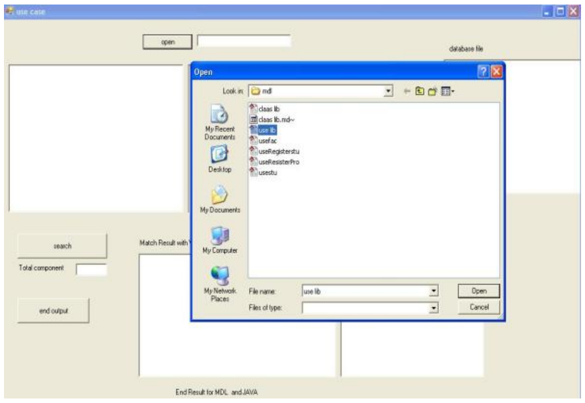


Figure 7: Interface for providing search input, which is use case diagram MDL File

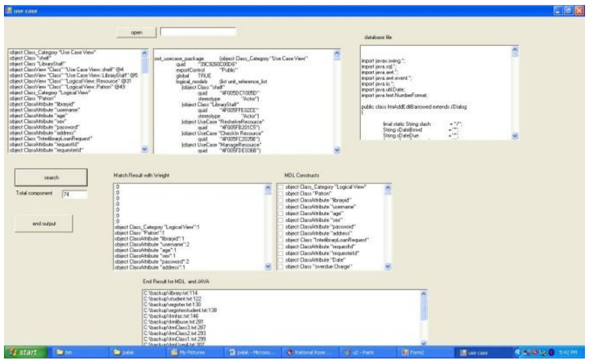
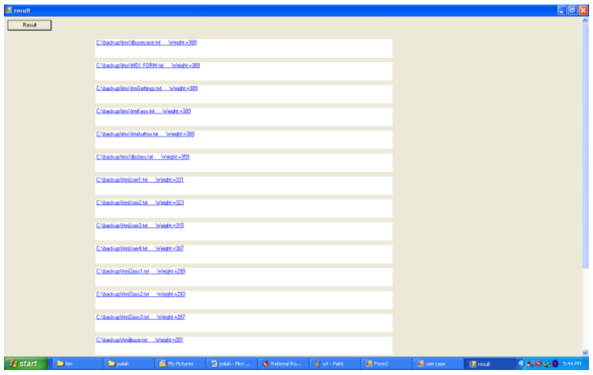


Figure 8: Interface for the overall search process

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..



```
result
Result
C:\backup\lms\libusecase.txt    Weight =389
C:\backup\lms\MDI FORM.txt    Weight =389
C:\backup\lms\lmSettings.txt    Weight =389
C:\backup\lms\lmKass.txt    Weight =389
C:\backup\lms\lmAuthor.txt    Weight =389
C:\backup\lms\libclass.txt    Weight =359
C:\backup\lmUser1.txt    Weight =331
C:\backup\lmUser2.txt    Weight =323
C:\backup\lmUser3.txt    Weight =315
C:\backup\lmUser4.txt    Weight =307
C:\backup\lmClass1.txt    Weight =299
C:\backup\lmClass2.txt    Weight =293
C:\backup\lmClass3.txt    Weight =287
C:\backup\lmlibuse.txt    Weight =281
```

Figure 9: Result set for search of Use Case Diagrams from MDL file

4.3 ANALYSISOFDATASET

Precision: Precision is defined as the number of relevant components retrieved divided by the total number of components retrieved.

$$\text{Precision}=\frac{\text{Number of relevant components retrieved}}{\text{Total number of components retrieved}}$$

Recall: Recall is defined as the number of relevant components retrieved divided by the total number of relevant components in the index.

$$\text{Recall}=\frac{\text{Number of relevant component retrieved}}{\text{Total number of relevant components in the index}}$$

Case1: MDL File – Class Diagram Search
Total components in the repository = 74
Total number of components retrieved = 68
Total number of relevant components retrieved = 48
Total number of relevant components in index = 70
Precision= 48 / 68 = 0.70
Recall= 48 / 70 = 0.68

Case2: MDL File – Use Case Diagram Search
Total components in the repository = 74
Total number of components retrieved = 19
Total number of relevant components retrieved = 17
Total number of relevant components in the index = 19
Precision= 17/19 = 0.89
Recall = 17/19= 0.89

Attaining the precision of 0.70 in Case 1 is considerably good which indicates that match is up to 70%.

Recall value of 0.68 indicates that we would have been able to retrieve 68% relevant components, in Case 1.

In Case 2, getting the recall value of 0.89 indicates that nearly no relevant component is missed.

Table I: Randomly Selected Components [11]

Component ID	**Component Name**	**Component Model**	**Component Type**
1254	Addition	Corba	dll
1253	Addition	Java bean	Source code
1257	Addition	Com+	dll

The following result is obtained by the Keyword-based Search on the basis of precision and recall as shown in below table II and compared with technique based on UML diagrams.

Table II: Result comparison of Keyword Based search and Proposed Technique (UML diagrams)

Comp ID /Comp Criteria	Corba	Java bean	Com+	**Class Diagram of Library Management**	**Use case Diagram of Library Management**
Precision	0.25	0.125	0.25	0.70	0.89
Recall	0.16	0.375	0.166	0.68	0.89

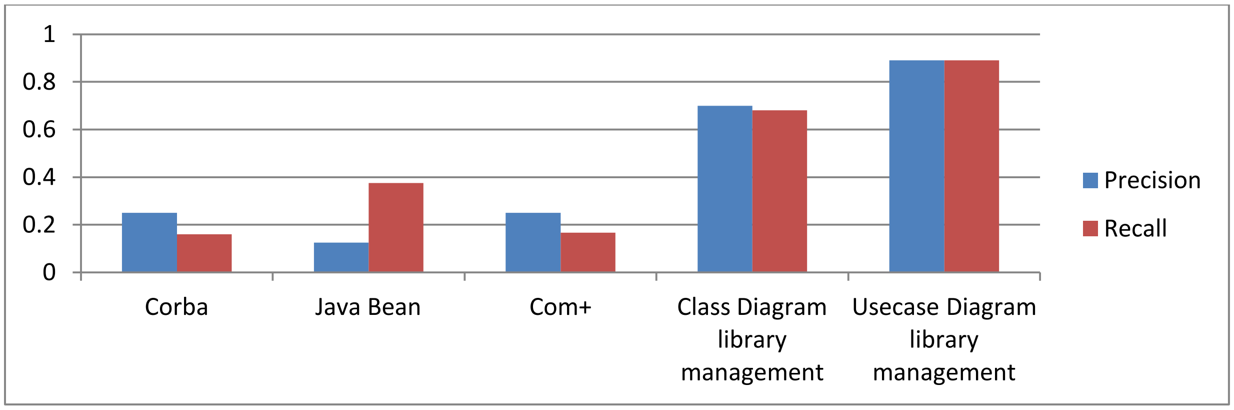


Figure 10: Result comparison of Keyword Based Search Technique and Proposed Technique (UML diagrams)

As we can see clearly from the table II as well as from figure 10, the proposed UML based component retrieval technique is much better than keyword based search technique.

5. CONCLUSION AND FUTURE SCOPE

Combining software reuse with UML is a new emerging trend in software development process. Combining these technologies helps the software development process by locating pre-existing components at the design time only, due to which the total effort of software development is decreased. To add to the tools for software reuse and UML, Component Retrieval Search Engine is proposed which helps for the diagram extraction and retrieval of the best-fit component from the repository. The diagrams of UML are modeled in Rational Rose and stored in MDL file format. These UML designs along with the source codes are stored in the Repository. After that a search engine is there to extract diagrams, as well as source codes from the repository according to requirements of the user, by search technique. The work presented can even produce more delicate reports depending upon the output required. UML and software reuse working together can be used to retrieve appropriate components. The search of diagrams as well as source codes from the repository can be made to search for compatible software/components available in the repository before designing and coding by providing the MDL file to the search engine. These arches can be made on class diagrams to search according to structural description of the software. The search can be made on use case diagrams to search according to requirements specification of the software. These architectures results are displayed in decreasing order of percentage match. Hence the role of (re)user to find the best-fit component from this architectures result is much easier. It has been observed that there is considerable improvement in precision and recall.

In future, this work can be improved search techniques to extract data from the repository can be improved further by including other UML diagrams. This work can be integrated with Rational Suite to facilitate component reuse.These can be integrated with some standard library or repository provided by organization.

REFERENCES

[1] Wenbin (William) Dai, Valeriy Vyatkin, "A Component-Based Design Pattern for Improving Reusability of Automation Programs", IEEE 2013.

[2] Ivica Crnkovic, "Component-based Software Engineering: Building Systems from Software Components", Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02) IEEE, 2002.

[3] Burton, B. A., Aragon ,R. W. , Bailey ,S .A., Koehler ,K .D., and Mayer ,L .A, "The Reusable software library", IEEE Software 4, 4, 25-33, 1987.

[4] Lionel C. Briand, YvanLabiche, Massimiliano Di Penta and Han (Daphne) Yan-Bondoc, "An Experimental Investigation of Formality in UML-Based Development", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 10, OCTOBER 2005.

[5] Robert A Andrews, Behan Webster, "A Component Oriented Software Engineering Approach to a Deeply Embedded Firmware based Control Platform", CCECE/CCGEI, Saskatoon, IEEE, May 2005.

[6] Patrıcia D. L. Machado, Jorge C. A. Figueiredo, Emerson F. A. Lima, Ana E. V. Barbosa, Helton S. Lima, "Component-Based Integration Testing from UML Interaction Diagrams", IEEE 2007.

[7] Marco Tulio Valente, Member, IEEE Computer Society, Virgilio Borges, Leonardo Passos, "A Semi-Automatic Approach for Extracting Software Product Lines", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 4, JULY/AUGUST 2012.

[8] Robert B. France, James M. Bieman, Sai Pradeep Mandalaparty, "Repository for Model Driven Development (ReMoDD)", IEEE 2012.

[9] Cristina Seceleanu and IvicaCrnkovic, "Component Models for Reasoning" ,IEEE 2013.

[10] Hafedh Mili, Fatma Mili and Ali Mili, "Reusing Software: Issues andresearch Directions,"IEEE Transactions on Software Engineering, Vol. 21, No 6, 1995

[11] Amandeep Bakshi, Seema Bawa, "Development of a Software Repository for the Precise Search and Exact Retrieval of the Components", International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 3, Issue 8, August 2013.