Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/102434, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Algorithmic Approach for Efficient Retrieval of Component Repositories in Component based Software Engineering

R. K. Bawa[1] and Iqbaldeep Kaur[2*]

[1]Department of Computer Science, Punjabi University, Patiala - 147002, Punjab, India; rajesh_k_bawa@yahoo.com
[2]Department of Computer Science and Engineering, Punjabi University, Patiala – 147002, Punjab, India; eriqbaldeepkaur@gmail.com

Abstract

Objectives: Software development is largely being transformed from traditional model based approach to plug and play component based approach which is required for enhancing reusability of the system for robust development. **Methods/ Statistical Analysis:** Tremendous techniques and models have been used and adopted for component qualification and adaptation using selection, construction, analysis and evaluation algorithms. The major task has always been to find an optimum centric approach to trade off the advantages of customized development and development using reusable components. **Findings:** CBSE environment and development by using various components as object, detail description and tools with platforms have been presented. The classification algorithms showed robustness of component, based on various extracted module. The ontology based algorithm is one of the robust techniques in the retrieval process. Futuristic implementation will emerge with this segregation cum classification scenario in a single unit that is to refine component presentation, management and retrieval. **Applications/Improvements:** Initially IBM Company adopted the concept of CBSE by using the System Object Model (SOM). The functions, libraries and UNIX pipes are some other examples of such approaches. The scope of the improvement, as far as findings and further work is concerned, lies with the selection, classification and retrieval scheme for component repositories. This has been improved with different representations of meta data of the component such as domain, interface, services and other semantic information.

Keywords: COM, Component based, CORBA, Component Life Cycle, EJP, Software Component Models, Software Engineering

1. Introduction

In Software engineering, the significance of reuse is inevitable. In the early days of computing, the programmers adopted the idea of reusability with the use of software libraries, functions, UNIX pipes. Today, every developer tends to build their complex software system within a small period. Such concepts can only be realized through the organized approach of 'reusability' with 'Plug and Play' methodology using Components off the Shelf (COTS). This reusable components-based approach has emerged from the object oriented approach to provide efficient 'reuse'. A component is an independent executable unit that has already been tested and deployed. These generic components are then selected from a repository to be assembled together to form a new software.

In CBSE development model, each component is designed in a generic way to extend as much reuse as possible. These components are assembled to build various complex software in limited time and cost for different application domains. CBSE performs the two concurrent engineering activities:

P_{le} product-line engineering, C_{bd} component-based software development. F Features, A Architecture ,C Compatibility, S Sequence, O_s Organization structure, L Lifetime, Q_u Qualities, U_I User Interface, A_i added incre-

*Author for correspondence

mentally, U_{II} usually identical, B_C backward compatibility, S_S sequentially, O_{TR} One team for each release structured, S_Q Same qualities, S_{TS} stays the same.

Table 1. CBSE concurrent engineering activities

Sr. No	Activities	Area	Multiple Release	Product Line
1	P_{le}	F	A_D	usually are not a subset of another product
		A	U_{II}	instance of PL has its specific variations
		C	B_C	no compatibility
		S	S_S	Independently
		O_s	O_{TR}	structured organization
		L	old release → new release	indefinite lifetime
		Q_u	S_Q	each product has different qualities
		U_I	S_{TS}	each product → different user interface
2	C_{bd}	Areas		
		Domain Engineering Management Application analysis		

P_{le} product-line engineering, C_{bd} component-based software development. F Features, A Architecture ,C Compatibility, S Sequence, O_s Organization structure, L Lifetime, Q_u Qualities, U_I User Interface, A_i added incrementally, U_{II} usually identical, B_C backward compatibility, S_S sequentially, O_{TR} One team for each release structured, S_Q Same qualities, S_{TS} stays the same

CBSE approach works well with the integration of the component models for developing the large and complex software systems. In addition, CBSE reassures the use of a predictable architectural strategy for enhancing the resultant software quality.

In Table 1 the area, the activities related to Product line engineering and component-based development have been described. Software reuse is a pivotal to Assembly line development. In the production of new software system[1], the domain knowledge is reused through the process of Assembly line development. It explores application engineering to precisely determine the functional, data and behavioral components that are contenders for reuse. These precise components are accumulated in reuse libraries.

The different phases are covered in Figure 1 for developing the software application by components.

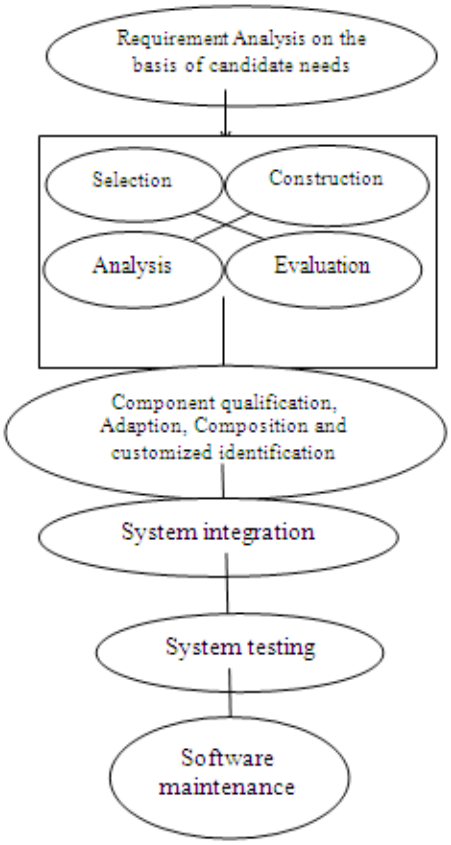

Figure 1. Component development lifecycle.

Component-based Development (CBD) analyses the customer requirements for developing an appropriate system by selecting the possible components for reuse. CBSE deals with necessary parameters in quality, cost, maintainability and throughput of the system. The development of the application is done with CBSE by using the data exchange model which includes storage structure and an object model.

The object model[2] observes one or more component standards that describe the method in which an application can access the reusable objects. The component standard models commonly followed are:

- Microsoft COM, DCOM (Component Object Model, Distributed COM).

- Object Request Broker Architecture (CORBA) by OMG.
- Oracle's Enterprise Java Beans (EJB).

Component models specify the standards and provide the interface between the component and the application software[3] where the components are used to build the integrated application software by using the list of component models. Some other notable component model architectures in addition to the ones mentioned earlier are NET, KOALA for consumer electronics, Fractal by ObjectWeb, SOFA by ObjectWeb and System Object Model (SOM) by IBM.

The phases of component development include: System requirements, design the components, develop the component and test the software[4] with structural and behavioral testing. The major advantages of Component-based software development lies in effective and easy management of complex applications, reduced development time, increased productivity, improved quality, increased reliability, reduced development and ownership cost[5]. However, unsatisfied requirements, lack of effective software metric, quality of components are some of the challenges being faced in component based software approach.

Many organizations and researchers have contributed to Component-based software engineering by establishing the base of component models for software development at industry level. CBSE has also been supported with advanced technologies such as ActiveX, Web, JavaBeans. These component models provide the platform services and horizontal services at the middleware between the components and the developing application. The platform services of component models are used to provide addressing, exception handling, component communication and other middle ware characterization[6]. The Horizontal services are the services independent of the software which are used by miscellaneous components and provide the concurrency, security, persistence, Transaction Management, Resource Management and Component Management. The three most popular component models COM/DCOM, EJP and CORBA cater to distinct level of services for developing the application.

1.1 COM: Model for Component Units

COM is model architecture from Microsoft Windows family that outlines direct interaction between the components and clients. It is a specification of standard set to create components. COM allows any programming language having pointer to function to be used as source code for application. COM Component is made of representative objects with identity and states. However, the source code is not accessible to the user. COM components like exe files or DLL can be linked dynamically anytime when needed. These are completely reusable as well. The binary standard in COM enables automatic software upgradation and versioning. The communication can be done through interfaces of the component like Packaging API and Active Directory. The distributed version of COM allows creating distributed applications over heterogeneous networks with a complete transparency of physical topology to the users. This means the objects may be lying at the remote computer but will work as if located on the local system.

1.2 Enterprise Java Beans

The server-side development framework Enterprise Java Beans is used for modular construction of highly robust and scalable distributed systems. A classical EJB system consists of EJB server, EJB client, EJB container, Java bean and some extension services like JDBC JMI and A Java Bean is a reusable component built to inter-operate across all Java based platforms. There are beans viz. time based such as session, object based such as entity and token based such as message driven. Time based beans such as Session are maintained till the time the calling client is alive. These can be either stateful of stateless. Entity beans implement various business units like rules, logistics etc. The EJB applications are implemented on JEE compliant Application Servers such as JBOSS, Web Logic etc.

1.3 Common Object Request Broker Architecture

CORBA or Request Broker based Architecture for Common Object is a middleware architecture and infrastructure that uses IIOP protocol to enable systems to interoperate. It is an application framework originated by the group that manages objects that is OMG that provides protocols and standards for integrating application development. It is a suite of protocols that offers reusable, interoperable and portable features specifications of object based software components in a distributed environment. CORBA is largely used in OODS since it gives

a reliable and consistent distributed programming and environment which is run-time in nature over diverse, heterogeneous and distributed technologies.

These Component Models provide conformance and standardization to the developing system with the usage of component based software engineering.

2. Background and Motivation

The idea of developing the application system using components first came into existence at North Atlantic Treaty Organization conference on software engineering titled as Mass Produced Software Components in 1968 by Douglas MacIlroy in Germany. In early 1990s, the IBM Company adopted the concept of CBSE by using the System Object Model (SOM). Then the CBSE concept was extended by the Microsoft Company for developing their software using the Object Linking and Embedding (OLE) and Component Object Model (COM). As of now in 2016, Component based development has formed the basis of software development in one or the other way in many of the companies. The available literature suggests the prominent research involving component development activities.

In[7] author has proposed a draft for achieving the advanced standardization life cycle model for component based system with application packages. Therefore they propose amalgamation of component based life cycles in order to retain a scheme functioning and defensible.

In[8] author has surveyed the current approach of component based software advance technology and determined the advantages and disadvantages while developing the software system. There has been considered the Quality Assurance issues for CBS and a model for QA has also been proposed to cover the requirement analysis phase, development phase, Certification phase, designing and testing phase in the component based software system.

In[9] the author has presented an overview of existing component models in use. Also a few successful instances of CBD are shown.

In[10] the author has analyzed the current component models and classified them into a taxonomy methodology with its key characteristics. These models have also been evaluated. Also reusable components and operators of composition which perform better systematic composition have also been used in models. It is concluded the robust model must have the key features of oops such as encapsulation and compositionality.

In[11] the author has figured that the practice of Component based development outperform when one compare it with some s/w developments those are traditional. The process model such as object oriented has been emerged as a novel, unique process model for development based on component. The main role of the repository in component development has been illustrated in detail. From the results one can concluded that CBD technique is much better for the cost effective parameter along with time saving and productive for software development.

In[12] the author presented an experience with corporation in the field of CBSE. It has been experimented with a number of industries/organizations having big projects with number of small projects as partner under the research wing. Each of this entity of cooperation has its own pro and cons.

In[13] the author has introduced the CBSE for the purpose of reusability for definition and deployment of independent component units into complex software structures. The interfaces of a component specifies its functions, explicit dependencies and the interfaces and its processes whereas the component model consists of a framework with standards and rules for COTS selection for compliance by component assembling team. A detailed evaluation criterion has been specified for refinement of application requirements. The evaluation process tends to be positively affected by the definition of detailed evaluation criteria driven by user requirements.

In[14] the author has evaluated CBSE approach towards software development by deploying a Personal Information System. A service calculation software component has been developed for multiple system nodes within PIS. Such hybrid and heterogeneous software system require inter-operability to perform precise and accurate calculation. A clear feasibility study of the whole set up of CBD leads to better reusability, tractability and maintenance.

In[15] author has focused on the task of a precise selection and retrieval of component given a query for certain software development. A thorough review of various tools and algorithms for exact component retrieval from available well-structured component repositories has been done.

In[16] the authors have presented component retrieval approaches that incorporate not just the syntactic infor-

mation but also the semantic and ontology data[17] as well. The meta-data model that takes into account the faceted taxonomy and repository along with a matching algorithm results in all possible relevant interrelated components.

Table 2. (Algorithm 1) Classification of components in CBSE

Algorithm_Classification_of_Components

Begin:
Step 1. Extraction Algorithm
{
Step 2. Input the Program Filename called F_{Name}
// F_{Name} or any other variable user choice
Step 3. While S = GetLine(FName) <> null
//Read all entries till the end of file
[Repeat Steps 4 to 11]
Step 4. if Contains(S, "c")
{
//c is class
Step 5. M++
//M= Module
Step 6. E_M = Extract Module(S)
// Extract all module for given file
Step 7. M = $T_m(E_M)$
// T_m =Total Modules
Step 8. M_e = Get M_e (M)
// M_e =Methods
Step 9. TM_e = TM_e + length(M_e)
// TM_e=Total
Methods
Step 10. A = Get A (M)
Step 11. T_a = TotalAttributes + length(A)
// T_a=Total Attributes, A= Attributes
}
Return (M, TM_e, TotalAttributes,)
}
End

In[18] the author has suggested that selection and identity of object as component for business through grouping method such as clustering. This proposed technique is handful for choosing components those are reusable with various sections such as domains also in the selection of newly configured component as reconfigured entities using the CBO measure that eliminate bonding between the entities such as objects for increased productivity in the organization.

In[19] the author has presented the representation of software component in the form of a suffix tree. A suffix tree data structure generates and represents components for effective modeling and specification. This further aids in effective component search from within the component repository. A multilevel suffix tree is built that is pruned at each level over the given keywords. Hence, edge based modeling under single symbol retrieval is done. The system performance and efficiency is based entirely on the pattern specification done for a defined requirement analysis.

A probable sequence of steps for the proposed efficient search is given as in Algorithm 1 in Table 2. First, the file name of the software repository is given in which the required component is to be searched. Repository selection is followed by the retrieval of components that are classified on the basis of type (for example a class, function or parameter). Then, an effective representation on components is done using suffix tree. This is followed by applying the keywords for the required component using keyword based search. Finally, the search results are presented to the user. It may indicate that if the necessary entity such as component exists in repository or exist somewhere else. The process of searching will repeat for all file entity for the full project.

In[20] the author has worked on explicit model of reliability and probabilistic approach for prediction entities such as components. In this, the reliability schema models are by design converted into Markov models by reliability prediction tool. The evaluation of the given approach using case studies in terms of reliability prediction and sensitivity analysis has proven the effective and correct design decisions making.

In[21] author describes the model system theory for software engineering design process. The overall design process in software development has been bifurcated into two concepts. The first one is decision making concept and the other one is a decision-under-risk process based on values. The model system concept has been considered as a set of interrelated homogeneous software components used for definition, development, and delivery of a software system. The model system concept as practiced in industry and academia enables representation of the manifold aspects of the project like stakeholders, domain areas and elements related to decision making process.

A Recent Article Published in January 2016 gives an insight into the future innovations in the field of component based development.

A meta-data repository based on the equation of facet and term space has been established with an abstract knowledge of its domain which formed coherent retrieval in the domain. Such an approach has shown to give

promising results in terms of precision and recall parameters. A comparison of results through general faceted retrieval and the meta-data method are shown as in Table 3 - Software engineering[22]. Future Software development may prevent car collisions, help fight human trafficking and predict genocide, enable electronic prescriptions and health records that save thousands of lives and is going to change the way doctors perform life-saving surgeries. All these embedded real time software development and implementation would be largely component based software engineering.

3. CBSE Retrieval Technique

Most component retrieval methods currently available for the users could be defined from three different perspectives each considering some aspect of the component retrieval method as presented in Figure 2. These three perspectives are: Component representation[23], component query specification that is focused on the requirements of the user and component retrieval process.

Component based Software Engineering deals with the challenge of precise component selection out of a large number of reusable components that are individually developed tested and stored in different structured repositories. For an efficient and precise retrieval of component, the conventional information mining methods are not enough. So, the domain engineering is done and the semantics and specification data is also incorporated in the retrieval process.

In[24] the author provides a solution based on ontology-based component repository to overcome the limitations of the traditional method. Using the data mining retrieval methods, the search engines for the users query processing are based mainly on literal matching of keywords to retrieve software components specification. But in this the conceptual meaning of the keywords and their synonyms are not applied. So, the author has used formal axioms to represent detailed information on concepts and their relationships as well as restrictions related to properties and concept values. The semantic-based approach makes the component retrieval more efficient and precise. Semantic information of a component may include the description logics in ontology languages with ontological components descriptions. Therefore, an ontological-based retrieval model based on semantic data of components proves worthwhile.

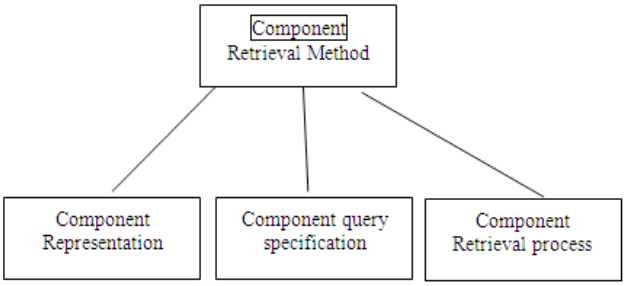

Figure 2. Component retrieval method classification.

Reusability refers to putting into use already tested and deployed software units so that the effort of working from scratch every time for the same task can be saved.

Almost every output from SDLC phases like the SRS, source code, design documents and other manuals. It is important to note that there is a tradeoff between the usability and reusability of software components. Successful reuse in component based software engineering paradigm requires having a large variety of good quality generic entities as components, well-structured ontology and proper classification and retrieval procedures. Effective software reuse implies having access to suitable and relevant components by users. The search and retrieval mechanism must have a high–level query formulation system that can generate queries on the basis of component semantic and domain information.

Table 3. The Experiment results of software components retrieval.

The Method of Retrieval	Components in Repository	Component Retrieved	Relevant Component Retrieved	Relevant Components in the index	Precision (%)	Recall (%)
T_{fr}	400	380	320	350	84	91
M_{DLF}	400	372	323	348	86	92
M_{RBR}	400	375	340	344	90	97
$M_{dr(Proposed\ Method)}$	400	392	375	380	96	98

T_{fr} =Traditional faceted retrieval, M_{DLF} = MDL File based retrieval, M_{RBR} Metadata repository based retrieval, M_{dr} Metadata repository and ontology based component retrieval

Accurate and precise component retrieval is followed by requirement customization or the component adaptation with the help of glue code if the need arises. Classifying software components supports quick retrieval by organizing the components into structures on the basis on domain, keywords, facet, specification etc. Different component classification and retrieval techniques have been used so far for reusable component repositories namely keyword based, Faceted, Enumerated, Attribute Value, Specification/signature based, Boolean, probabilistic, vector model odds, tf-idf and HTML based classifications. A matching algorithm is used to retrieve the best relevant component given a user query. The component modules communicate using common interfaces. Therefore, the overall architecture of proposed system has three major phases as it is shown in Algorithm 2 in Table 3.

Table 3. (Algorithm 2) Ontology based component retrieval method

Algorithm: Ontology_Component_Retrieval

Begin:
 	
Step 1: C_Q,T
// C_Q = Component based developer's query
//T= tokenization
Step 2: For (F=1,F<L_r,F++)
{
Step 3: Extract features and stored in feature vector
Create F_v
// Feature vector F_v where F_v€ C_Q,T
Step 4: Perform S_S
// Technique with RDF KB Database by distributed component repositories through ontology construction.
//S_S= semantic similarity
Step 5: Perform S_o using ontological indexing
// S_o Component Ontological Searching and retrieving
Step 6: Perform Ordering
//Sequencing and order & display of retrieval components.
Step 7: Perform R_{oc}
// R_{oc} Retrieved Ordered Components.
}
End

3.1 Keyword based Retrieval Technique

The retrieval system with free text classification or uncontrolled vocabulary generally applies the simplest form of search called a keyword based search. The user need to input certain words called keywords in the query to be looked for in the database or repository. Matching is done word by word and then a list of relevant components with a ranking is returned to the user. Keyword based search being the simplest search technique is widely used by web search engines and component recommenders in combination with other techniques. Most popular retrieval websites support keyword-based findings as initial technique for searching to find documents and web resources such as web sites. Free-text classification firstly analyses the number of occurrences of words or terms, called term frequency. These frequencies are used to derive relevant keywords through associated positional/statistical properties. This weighting is called automatic indexing. Indexing and searching are the two major processes in keyword search. The indexing process works to create a list of search items from all the available components in different repositories. This list is then utilized while the search process takes place. Keyword based search allows users for free text queries which also can lead to challenges like:

- Identify keywords to best explain the user need in a query.
- Possible ways in which a user may search.

3.2 Enumerated based Retrieval Technique

They said classification is a one-dimensional classification which implies the collection of mutually-exclusive objects in the classes. The DD (Dewey Decimal) model used to segregates or classify books in a library is a typical example of such classification. Each subject area like Science, arts, literature etc, has a unique classifying code. Being as further code (Sub) of this it behaves as specialist area within the main class as subject. These codes can further be sub coded to many levels. This classification does not work well with the reusable component repositories since its one dimension mutual exclusion feature does not permit choice based classification of entities as components to multiple place reusable software components. However it provides subtle support for best retrieval of components.

3.3 Attribute Value Retrieval Technique

The said classification scheme uses a set of parameters to classify various entities as components. In the library coding example, consider a book with attributes like title

of the book, publication house, number of pages, year, edition and an ISBN number which is unique and categorized code in the Dewey Decimal system[7,25]. Hence, the query can be framed based on such attributes to search a particular component. The challenges in component attribute classification can be:

- Variable attribute dimensions. The attributes being used to classify a component can vary from one repository to another.
- Dynamic attributes. All possible variations of parameters may not be known in advance at the time of classification.

3.4 Faceted Retrieval Technique

Facet based classification was proposed by author[9]. It depends on the facets, each of which represents a feature or a component descriptor. Experts carefully extract these facets that carry information about the functionality and implementation process related to the component. In this, the users are given choice in an exploratory or guided way to select features available for search. This helps the users to accomplish their search goals in a quick and efficient manner. Although this technique requires effort and labor for detailed representation of component descriptor, the retrieval of component has already been proved with the said method from repositories. The faceted scheme proposed by Prieto-Diaz used six facets.

- The functional facets can be: Function, procedures, Objects, routines.
- The environmental facets may include: Operating System, Network parameters, lines of Code. Each facet[25] is assigned a value at the time of classification of the components.

It is an enhanced version of the pre-enumerated vocabulary method that also takes account of domain knowledge as well[23] when designing the facets. In[26,27] authors have emphasized Facet retrieval to be one of the most far and wide approach that has been used in the retrieval techniques. Each facet represents the essential characters of components. Facets are evolved and formed by identification and then collection of certain relevant terms called vocabulary of the field. This organized way of classifying terms and hence the components as per the domain improves the retrieval process equally. In addition to this, such a cataloging arrangement also adds up for filtering and evolution of typical terminology and lexis for components and the associated attributes. Thus, the whole scheme is composed of a number of facets which further are made up of interrelated terms and words.

Although this method may seem as the most appropriate of the earlier mentioned methods, it has some limitations when it comes to forming the facets. The facets with a very complex design make the classification of components even a more difficult task since all components classified into different categories is challenging for users to understand. On the other hand, this method also performs low on too few or very simple facets as then a large number of components fall in same classification. This will defeat the purpose of the retrieval algorithm if the users will have to select further manually. One of the major problems with this method is that in most cases sufficient number of components do not come out as an exact match due to widespread dissimilarities between given components descriptions and component requirements that are later defined by the user.

3.5 Query and Browsing based Retrieval Technique

Querying forms an essential activity while retrieval takes place. The actual requirements are framed into a question or structured 'query' that is inputted to the retrieval system. The query is match with the available components in the repository and the relevant set of components is returned. In certain cases, query formulation becomes an arduous task due to unclear requirements and dearth of component domain terms. In such cases, browsing based retrieval may resolve the problem. Browsing refers to exploration and navigation of interlinked links. It enables users to traverse and then decide about the relevance of the information of components. The researchers claim that in component based development approach with ambiguous and vague requirements; the assemblers and component selectors majorly count on the browsing based retrieval to be acquainted with the existing components and the repositories.

3.6 TF-IDF

Term-frequency and Inverse Document frequency is a statistical numerical value that is used to assign weights to terms in a component. It helps to rank them in the order of relevance and importance with respect to a given query. This can be done at expression, document or entire component level so as to get precise results. TF-IDF

computation forms an essential part in keyword based retrieval.

3.7 Component Retrieval based on MDL Format

The unified modeling language with object representation can be utilized for component retrieval process to enable effective software reuse. This merger can help identify already available matching components during fabrication prior to the actual development. Therefore, it paves a natural way to reduced development effort. To add to the tools for software reuse and UML, one such Component Retrieval Search Engine has been proposed by[28]. This diagrammatic illustration eases the selection and extraction of the exactly as per Table 4 suitable component out of a 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 with s/w re-use working together can be used to fetch 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.
- The search can be made on class diagrams to find in accordance with structural description of s/w.
- The search can be made on use case diagrams to find in accordance with requirements specification of s/w.
- The search results are shown in descending way of % age match. So the role of (re) user to find the best-fit entity as component from the found result is much easier.

It has been observed that there is a considerable improvement in precision and recall.

3.8 XML Retrieving Method

Another approved method when it comes to retrieving components in cases when components are presented as[16] documents is XML retrieving method. It is guided by the idea to return only the most matched components (as entity) that answer the query specification. This method is further classified into 2 sub-tasks that are based on Content Only (CO) which is the simpler version where the user specifies queries as simple text and the search engine returns the most relevant XML of entities as components that answer the concepts (query based). The other one is the content/Structure topics (CAS) where the user can define the desired component to be returned by XPath by limiting concepts (query based) to particular XML tags. The primary version of this method has some extensions like a document pivot that scales scores of components by the score of their running entities as article that make significant improvement to the result. Another extension is to apply Automatic Query Refinement (AQR) algorithms on top of XML component ranking that gives excellent results in the CO track. Since there are many possible combinations of AQR parameters and their variants, the combination (best) that is to provide the best results is not found yet.

Table 4. (Algorithm 3) Reusability of the component through retrieval based on MDL format

Algorithm Reusability_ Component_ Retrieval Based on MDL Format

Step 1: Initialize User database U_d and $I_{q\text{-}MDL}$
//Where $I_{q\text{-}MDL}$ is query MDL file
Step 2: Read U_d and $I_{q\text{-}MDL}$
Step 3: For any S_e
//Where S_e Search engine
(S_e =1, S_e<< S_{el} , S_e++)
//Where S_{el} is l^{th} search engine
{
Perform Retrieval designs and source code through repository and main developer
}
Call Repository Manager.

3.9 Signature Matching Method

The said technique implies the signatures to recognize and hence to illustrate entities such as components and suitable queries and so with this to achieve a specification of the interfaces of entities for example no and associate types of i/p and o/p variables. The said method is highly recommended while searching for components that are meant to be implemented by using strongly typed programming. The problem with this method is its lack of ability for searching con-text information.

3.10 Behavior based Retrieval Method

In this case the method is focused on the executable components special characteristics. With this method queries are processed by a unique group as a set of i/p units as samples and their desired outputs when components take the form of executable codes. The retrieving begins with selecting samples that in the next step are used to execute the components. After the components are executed, those that give the appropriate output are retrieved. The problem with this component retrieval method is that it has low efficiency because it has long execution time and it is designed only for executable software components.

3.11 Knowledge based Retrieval Technique

It is the method that implies various type of reasoning which associate and relate the unique entity as query with previous one, alternatively that relate with the query of the components. The said methodology of retrieval is entirely based on making of decision on the basis of the knowledge-base present for the application domain. These methodologies maintain a database that consists of semantic analysis of NLP specifications, lexical, syntactic of s/w entities such as components. It manages the whole process by storing and matching the semantic information about the application domain (under test) and using natural language. This technique implies the use of a set of text which is verified by checking whether it is zero or partially/fully filled. In case the text set is not null then one can select the text for analysis by finding the topic which is found from the data base (dictionary) and the library of topic feature aggregation formula. Certain rules are applied on the methodologies to cater the need of finding the topic result in final o/p.

3.12 Hypertext based Retrieval

The hypertext based method uses the non-linear strategy for the design of information to the nodes links of the networks. The technique is based on the interconnection of the nodes where every tuple of the node is presented by link and each node is integral unit of textual information. The tuple is represented by parent node as starting point and destination node as ending point. The link may be one sided or multi directional. The information which is stored can be accessed by traversing along the links. Further links may be termed as pointer.

3.13 Semantic based Component Retrieval Technique

The semantic based retrieval methodology fetches the entities[27] that is the component on the concrete foundation of description about their domain (components). This approach can fetch the component entities on the features where features may be the entries in the vector set. These components are fetched from the database of repositories according to the matching scenario where as the meaning of every entry that is meaning of the component is mentioned in the repository. The procedure makes decision and without having a rigid and total knowledge of description the complete analysis has been performed the analysis is based on the information base that is stored using the semantics.

3.14 Genetic Algorithm based Approach

The genetic based information retrieval method is based on the natural selection and natural genetics in a bio system. The Darwin theory of the survival of the fittest is adopted. The algorithm is non- deterministic in nature which means it is used in situations where optimization necessity is high and requirement of model evolving system is must. The said algorithm gives the best and accurate findings in the form of result to the situations by historical breeding the population of sample under test. The steps selection and reproduction crossover and mutation are the integral parameters in the algorithm.

- SELECTION and REPRODUCTION: For initialization of the population and random select a sample which further fits the best
- CROSSOVER: From the cluster of fit samples, crossing find it place between the two samples (fit). The form of information exchange in bits even in string format is computational.
- MUTATION: The process immediate after the information exchange, there is the changing of the bits scenario. This implies the complete bit 1 is converted to 0 and vice versa.

3.15 Formal Specification Technique

Formal specifications used mathematical notations to describe the properties of computer system without ruling the way in which these properties are achieved. These

describe the functionality[29] done by system rather than actual description that is how it has been done. These abstractions make this technique apt for developing a computer system because they allow question about which the system do confidentially.

3.16 Structural Classification Technique (Informational Specification Techniques)

The authors[26,30] suggested a technique called as structural classification for the retrieval of component that incorporates features of structure extraction. The crux of the methodology lies in getting the structural information rather than their function. For the set of two different sets of criteria that one can use to select test components in a s/w library as functional criteria that looks for the function with similar properties based on functionality as query and structural criteria that look for the similar structure as best possible answer cum solution to the query. Functional criterion is good for black box reuse and structural criteria are best adapted by white box reuse. Signing and specification are the two methods in the structural classification. The formula, logic are the basic building blocks for any said classification. The component retrieval method uses proof theorem strategy to correlate the query with the entities (as components) in library specifications. Classes as object are defined through many set of feature signature. However the signature matching approach assumes the pre information (knowledge) of the permitted set of the signatures in the system. There are many flops of the structural classification

It is only aimed at white box reuse, so it only searches for approximate retrieval of the components.

Till date these methods are mainly in laboratory use.

Proof theorem that is required to match in library is very complex.

Signature matching technique is only beneficial if partial or full information or knowledge of the signature is pre-defined or known.

3.17 Behavioral Retrieval Technique (Informational Specification Techniques)

The behavioral retrieval technique is used to retrieve the component from repository. This focuses mainly on the behavior of the component. Behavioral retrieval is performed by exploiting s/w component's ability to execute. The codes are performed through components and answers are recorded. Fetching of the information is achieved by selecting the entity which is very much close proximity of the prerecorded set of desired results. The method is called 'behavioral matching' technique. Normally the said technique excites each component with i/p data stream in a fashion to fetch the entities that present the required behavior. Some flops of the technique are:

Random selection of the test samples to excite all operations in s/w library with a signature that best correlate the required one. More or less it is similar to black box testing.

3.18 Specification based Approach (Informational Specification Techniques)

Specification based approach[30] is useful for the enterprises where the reusability of the component is a major area of concern. The relation in the said methodology performs in 2 levels: Operation level and component level for business. In operation level the i/p is data type of business where as o/p data types and the related taxonomy of operations calculate similarity b/w operations that are business related. In the other type that is the component level, 5 attributes are matched b/w components. The similarity degree is the key parameter for the reusability. At last a query based process such as SQL is used for the retrieval from the repositories.

In the comparison with the other approaches, the said approach has the following features. The model is both static and dynamic in nature for getting structural information.

Further it is a multi-folded technique where the semantic information plays a crucial role for information retrieval from the repository.

3.19 Intelligent Classification Scheme

Intelligent classification scheme is also used to retrieve the components. This method for the segregation and fetching of components is divided[28] into two phases. In the first phase the classification, encoding and discovery are the key parameters. The second phase involves fetching of component from the reusable repository.

The Genetic Algorithm is used for the evolution of small no of different classifiers. This classification scenario portions the components into the small sets. The process results into the fast fetching of the right components according to the choice of the user. In the coding

the components are entirely encoded into binary format. The set of classifier which are non-heterogeneous emerge through the bit string process from the Genetic Algorithms. The component is then searched by the user in the fetching phase.

The user sets the attributes through the vector set of the desired component and sets the limit of threshold to become a matched component. First the user will enter the desired characteristics of a component which he wants to retrieve from the reuse repository, through an interface. Second the user will set the matching threshold value (for less threshold, a large group of relevant components will be retrieved and vice versa). For instance, for value of 33% minimum, 33% features are matched to those of the features of the components.

Fetching of information through retrieval methods based on feature characteristics need the user to place the list of some of the parameters of the entities as component sought. Informal methods inculcate the properties of keyword based search, sometimes multi-attribute search with natural language interfaces. Major algorithms can be classified and named as the follows:

- Natural language interface.
- Formal specification.
- Case based retrieval.
- External Classification.
- Encoding component description.
- Optimization approach.
- Full text indexing.
- Conversational case based reasoning.
- Rough fuzzy approach.
- Genetic Algorithm.
- Intelligent classification scheme.
- Specification based approach.
- Free text and faceted classification.
- Behavioral and structural classification.
- Attribute value.

4. Design of Component based Retrieval System

Retrieval of Components[27] from repositories can be identified through following techniques.

- Keyword based.
- Faceted.
- Specification/signature.
- Probabilistic.
- Vector model odds, TF-IDF.
- HTML based.

5. Conclusion

We report the CBSE environment and development by using various object, architecture and language platform. The classification algorithms showed how one can classify the component based on various extracted module. The ontology based algorithm shows the robustness of the retrieval techniques. The major task of repository and hence component management may be achieved by using MDL format based algorithm for reusability criterion. Futuristic implementation will emerge with this segregation cum classification scenario that will be to refine the scenario for component presentation based on multimedia and tired platform. Further crucial target of the repository scenario is to establish attributes to support logic generation. Indeed many methods of normalization and optimizing system for time saving with fast processing keeping efficiency at high must be explored in manner to keep the robustness of effective design tool for the repository fetching and management.

6. References

1. Lotfi R, Dastjerdi AG. The use of empirical software engineering to solve the software crisis. Indian Journal of Science and Technology. 2016 Jul; 9(27):1–4.
2. Shahabuddin SM, Prasanth Y. Integration testing prior to unit testing: A paradigm shift in object oriented software testing of agile software engineering. Indian Journal of Science and Technology. 2016 May; 9(20):1–10.
3. Wankhede HS, Kiwelekar A. Qualitative assessment of software engineering examination questions with bloom's taxonomy. Indian Journal of Science and Technology. 2016 Feb; 9(6):1–7.
4. Jain P. Towards the adoption of modern software development approach: Component based software engineering. Indian Journal of Science and Technology. 2016 Aug; 9(32):1–5.
5. Mohankumar M, Kumar MA. Green based software development life cycle model for software engineering. Indian Journal of Science and Technology. 2016 Aug; 9(32):1–8.
6. Su S, Tang H. Construct stereoscopic teaching system of software engineering course based on CDIO. Indian Journal of Science and Technology. 2012 Dec; 5(12):1–4.

7. Benneth C, Lars J. Component-based software development life cycles. Institution for Informations Technology: Karlstad University; 1999. p. 1–17.
8. Xia C, Lyu MR, Wong KR, Ko R. Component-based software engineering: Technologies, development frameworks and quality assurance schemes. Proceeding of IEEE 7th Asia Pacific Software Engineering Conference APSEC-2000; 2000. p. 372–9.
9. Ivica C, Magnus L. Component-based software engineering-new paradigm of software development. Invited talk and report, MIPRO; 2001. p. 523–4.
10. Kung KL, Zheng W. Software component models. Proceeding of Institute of Electrical and Electronics Engineering Transactions on Software Engineering. 2007; 33(10):709–24.
11. Qureshi MRJ, Hussain SA. A reusable software component-based development process model. Advances in Engineering Software. ScienceDirect. 2008; 39(2):88–94.
12. Zhang X, Zheng L, Sun C. The research of the component-based software engineering. IEEE Sixth International Conference on Information Technology: New Generations, ITNG'09; 2009. p. 1590–1.
13. Basha N, Mohan C. A strategy to identify components using clustering approach for component reusability. Natarajan Meghanathan, et al. editors. 2012; p. 397–406.
14. Chaitanya PG, Ramesh KV. Feasibility study on component based software architecture for large scale software systems. International Journal of Computer Science and Information Technologies. 2011; 2(3):968–72.
15. Amandeep K, Seema B. A survey for effective search and retrieval of components from software repositories. International Journal of Engineering Research and Technology. 2013; 2(4):1–9.
16. Gupta S, Ashok K. Reusable software component retrieval system. International Journal of Application or Innovation in Engineering and Management. 2013; 2(1):187–94.
17. Bhatia MPS, Kumar A, Beniwal R. Ontologies for software engineering: Past, present and future. Indian Journal of Science and Technology. 2016 Mar; 9(9):1–16.
18. Basha NMJ, Mohan C. A strategy to identify components using clustering approach for component reusability. 2012. p. 397–406.
19. Kanwaljeet S, Trilok G. A novel technique for components retrieval from repositories. I COMPUSOFT. An International Journal of Advanced Computer Technology. 2014; 3(6):912–20.
20. Stoica AJ, Pelckmans K, Rowe W. System components of a general theory of software engineering. Science of Computer Programming. ScienceDirect. 2015; 101:42–65.
21. Pham TT, Defago X, Huynh QT. Reliability prediction for component-based software systems: Dealing with concurrent and propagating errors. ScienceDirect. Science of Computer Programming. Special Issue: Selected Papers from the 12th International Conference on Quality Software (QSIC 2012). 2015; 97(4):426–57.
22. Forrest S, Anita C, Jeromy C, Rafael P, Dongmei Z. The future of software engineering. IEEE Software IEEE Computer Society. 2016; 33(1):32–5.
23. Bisera D, Anastas M. Component-based development and component retrieving techniques. V. Trajkovik, A. Mishev, Editors. ICT Innovations 2013 Web Proceedings; 2013. p. 142–51.
24. Nedhal AS, Intisar AS, Ahmed HA. Sematic-based retrieving model of reuse software component. International Journal of Computer Science and Network Security. 2010; 7(10):154–61.
25. Vaneet K, Shivani G. Facets of software component repository. International Journal on Computer Science and Engineering. 2011; 3(6):2473–6.
26. Richard Z, Bo Y. Keyword and image-based retrieval of mathematical expressions. In IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics; 2011. p. 78740I.
27. Shweta Y, Kamaljeet KM. Design of rank based reusable component retrieval algorithm. International Journal of Advanced Research in Computer Science and Software Engineering. 2013; 3(11):840–57.
28. Deepak K. Reusability: Component retrieval based on MDL Format. International Journal of Computer Applications. 2013; 65(19):1–7.
29. Nishi G, Kailash B. Design of component retrieval algorithm using enhanced rating based genetic. JECAS. 2014; 6(3):9–11
30. Aman JN. Component retrieval technique - A systematic review. International Journal of Scientific and Engineering Research. 2014; 1(5):1699–706.