SOFTWARE COMPONENT SELECTION AND COMPOSITION

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

ANSHUMAN SEKHAR DASH

2020IS04

Under the supervision of

Prof. D.K. Yadav

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD,
PRAYAGRAJ – 211004, INDIA

JUNE, 2022

I declare that the thesis work titled "**Software Component Identification and Composition**" submitted to the Computer Science and Engineering Department,Motilal Nehru National Institute of Technology Allahabad,Prayagraj for the award of the **Master of Technology** degree in **Computer Science and Engineering**, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

June 2022

Prayagraj (Anshuman Sekhar Dash)

CERTIFICATE

Certified that the work contained in the thesis titled "Software Component Selection and Composition", by Anshuman Sekhar Dash, Registration Number 2020IS04 has been carried out under my supervision and this work has not been submitted elsewhere for a degree

Date: dd.mm.yyyy

(Prof. D.K.Yadav)
Computer Science and Engineering Dept.
MNNIT Allahabad, Prayagraj

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep sense of gratitude to all who helped me directly or indirectly for my thesis work. Firstly, I would like to thank my supervisor, **Prof. D.K. Yadav**, for his advice and encouragement behind the successful completion of this dissertation. The confidence shown in me by him was the most significant source of inspiration for me. He provided me ample opportunities to explore myself. Besides my advisor, I would like to thank the rest of my thesis committee member: **comitee member names**, for their insightful comments and encouragement, but also for the hard question which incite me to widen my research from various perspectives. I wish to express my sincere gratitude to **Prof. Rama Shanker Verma**, Director, MNNIT Allahabad, Prayagraj, and **Prof. D.S. Kushwaha**, Head, Computer Science and Engineering Department, for providing me with all the facilities required for the completion of this thesis work.

Anshuman Sekhar Dash

Abstract

Reusing of software is found to have escalate the software engineers work output and also the craftsmanship of the final software product. Component based software engineering (CBSE) is a unique area in software engineering wherein software components i.e a chunk of software that contains a set of related functions and data are searched for and pieced together to form a software system.This method promotes re usability and efficiency in development of software systems. CBSE's purpose is to compose software applications using plug and play software components on the framework.There are several strategies for software component identification in a system among which most widely used is object oriented based clustering.We have proposed an approach which is based on clustering and formal methods to solve the problem of software component selection for a software system.The approach is based on describing a software component using various attributes like Domain, Type, Visibility and Language then using clustering algorithms that groups the components in an hierarchy.When components are arranged in an hierarchical manner it provides a means to store ,browse and retrieve reusable software components.Using the help of hierarchical tree generated software components are identified and selected software components based on the requirements from user, extracting the keywords by the removal of stop words and punctuation.The keywords are searched using the hierarchy and then string based searching is used to compare functionality of the subset of components and requirements keywords.The comparison is done using a similarity score which is used to select the software component.Composition of the software components selected is done to show proof of correctness of our work.

Motivation

Throughout last decade years wide spread use of software systems in many aspects of life and economy has placed new demands on the software industry for faster development of and software and also to revamp the quality of software and optimizing time take in developing of software.One method of achieving this is to promote software re-usability software re-usability means reusing of software previously developed which fulfills or nearly fulfills the software requirement As is the case with many software system , no two software systems are developed the same way.Every software system differs in one or more requirements and hence software re-usability will be limited unless a software system can be divided into software components each of which can be independently used in software development This is the reason for looking more into component based software engineering, various techniques used to identify software components in a system and then composition of software systems using components. Efficient and closest selection of software components from a software components repository for a given set of software system requirements is also a challenge in component based software engineering for which many different approaches are constructed.

Contents

Candidate's Declaration i

Certificate ii

Acknowledgement iii

Abstract iv

Motivation v

Contents vii

List of Tables viii

List of Figures ix

List of Symbols, Abbreviations x

1 INTRODUCTION xi

1.1 Component Based Software Engineering xi

1.1.1 Stakeholders in Component based software Development xi

1.2 Software Component Selection xii

1.3 Different Approaches for Component Selection 1

1.4 Software Composition 1

2 LITERATURE REVIEW 2

2.1 Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries 2

2.2 A QoS ONTOLOGY - BASED COMPONENT SELECTION 2

2.3 Evolutionary Algorithms for the Component Selection Problem 3

2.4 Using Formal Methods to Construct a Software Component Library 3

2.5 Design an Algorithm for Software Development in Cbse Environment using Feed Forward Neural Network 4

2.6 Software Component Recommendation Using Collaborative Filtering . . . 4

2.7 Classification and Retrieval of Reusable Components using semantic features 5

2.8 Vector Space Based on Hierarchical Weighting: A Component Ranking Approach to Component Retrieval 6

2.9 COTSRE: a COmponenTs Selection method based on Requirements Engineering 6

2.10 A Fuzzy-Based Approach for the Multilevel Component Selection Problem 7

2.11 Improving Retrieval Effectiveness using Ant Colony Optimization 8

2.12 A Hybrid Technique for Searching a Reusable Component from Software Libraries 8
2.13 New Algorithm for Component Selection to Develop Component-Based Software with X Model 9
2.14 Intelligent Component Selection 10
2.15 A Novel Approach of Components Retrieval in Large-scale Component Repositories 11
2.16 Clustering software components for program restructuring and component reuse using hybrid XNOR similarity function 12
2.17 A Model Software Reuse Repository with an Intelligent Classification and Retrieval Technique 13
2.18 A PERSONALIZED FACET-WEIGHT BASED RANKING METHOD FOR SERVICE COMPONENT RETRIEVAL 14
2.19 A π-Calculus Based Approach for Software Composition 15
2.20 A calculus for reasoning about software composition 15

3 METHODOLOGY **17**

4 RESULTS and DISCUSSION **18**

5 CONCLUSION AND FUTURE SCOPE **19**

A Appendix Title **20**

List of Tables

List of Figures

1.1 Stakeholders in Component based software development xii

List of Symbols

r	Radius, m
α	Angle of thesis in degrees
β	Flight path in degrees

Chapter 1

INTRODUCTION

1.1 Component Based Software Engineering

From the last 10 years use of computer software has entered into every part of economy which has in turn generated new demand from the software industry to develop reliable and cost effective software quickly and efficiently. Modern software systems are becoming more and more large scale which further increases the development cost , decreases the productivity and increases the cost of maintenance. In the 90's a understanding was developed among software engineers that a software system need not be developed all the way from scratch and instead the software system can be developed by piecing together small pieces of software from previously developed software system and assembling them to make a final product.This idea was further developed into a field in software engineering named component based software engineering(CBSE).

Software Component is a modular, portable, replaceable, and reusable set of well-defined functionality that encapsulates its implementation.In the context of CBSE a component will be defined as A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.By the procedure of CBSE, integration and up gradation of system in future is possible just by the deployment of some components in place of disturbing the whole system.

Component-based software development is very well known and widely used as a methodology, which helps greatly in the re usability of software and reduces the cost of development significantly.

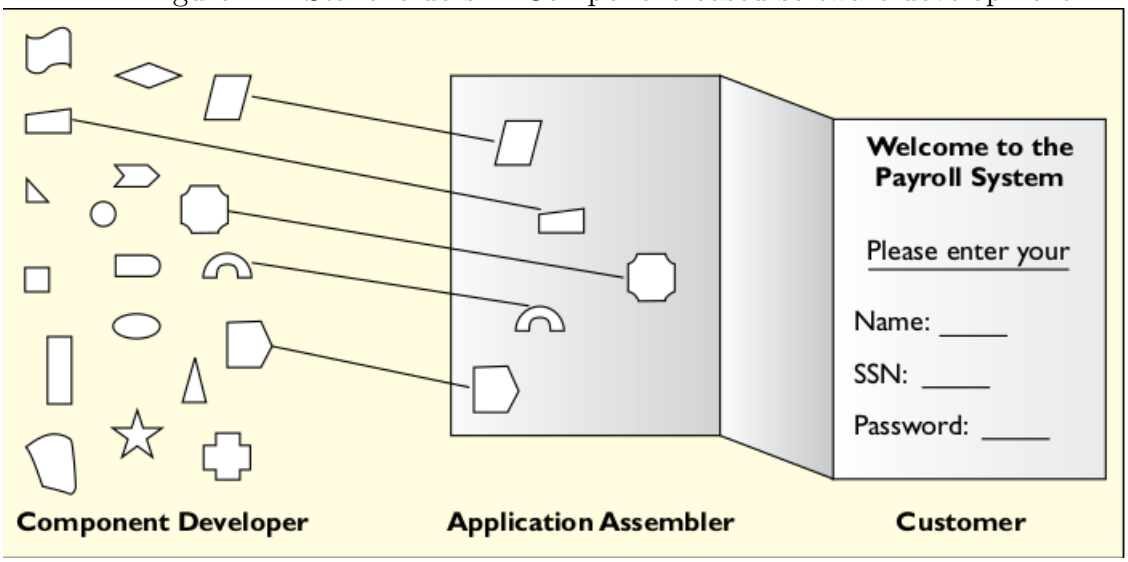
The primary role of CBSE is to direct making of software system as stitching of parts or components , development of the individual components as reusable entities and maintenance of the system developed with timely replacement of components.

1.1.1 Stakeholders in Component based software Development

- **Component Developers**:-These are engineers that can be small freelance developers or an entire organisation whose job is to develop and catalogue the software components such that any further system can be developed using the software components they develop

- **Application Assemblers**:-Application Assemblers locate appropriate software components and according to software requirement assemble them into an integrated system.

Figure 1.1: Stakeholders in Component based software development



- **System Developers**:-Developers are responsible for creating the software systems.They take decisions on which components to use based on the business domain and system requirements and availability of the components.They have to make a conscious decision whether to develop the requirement form scratch or use an existing software component which may be having more generalised functions.
- **Customers**:-They use the systems that are developed using component based software development. Their interest lies on faster delivery of system, cheap and reliable to use.

Important problems in Component based software development is identifying software components that can be reused and selecting of components for the development of a software system.Software component identification and software component selection are one of the primary research problems in CBSE.

1.2 Software Component Selection

Once it is decided that Component based software engineering is to be used, The programmer/developer prepares a set of requirements for the system.For fulfilling the set of requirements the developer needs to browse through software component repository.With the growing usage of component based software engineering the component repositories contains hundreds and thousands of components which are always growing day by day and hence selecting a particular component from these vast repository of components is a challenging task and some solution needs to be proposed for it. The structure of component selection problem is that of optimization problem.Hence many optimization methods have been used in component selection.

1.3 Different Approaches for Component Selection

Throughout the years there have been many approaches formulated by researchers for efficient selection of software components from a components repository.Some are based on Evolutionary algorithms based approach such as ant colony optimization and some are based on neural network based approach. Keyword based technique uses domain based dictionaries that helps in minimizing the dimensions of search space.The keywords are extracted from a query processing language and based on the frequency of matching components are arranged in descending order. Signature Based technique uses component signature for finding a particular component.Query matching is done by doing matching with query signature and component signature.Both exact matching and relaxed matching cases are considered and the results are shown by ordering from exact to relaxed matched.A faceted classification is a method of classification used to arrange information into a hierarchical order.A word is placed in the specified sense of the language and is defined by a particular angle of view (called facet) representing the critical feature of a software component in the facet classification The software component is defined using different facet and the organized term space is generated by common and special relationships. Behavioral matching uses component's behaviour.It uses the executability of software components.The responses of the software component while in execution is recorded.Retrieval is done by selecting those components whose responses are nearest to a predetermined set of desired responses.This mostly uses input data for matching. Semantic matching uses semantic information that are usually enclosed in lightweight ontology.The ontology matching operator works to classify nodes of two graphs that are similar to each other.The component is obtained based on relevance, which is if the query and the component's recorded aspects are same, then the component is referred to as the candidate component.

1.4 Software Composition

After getting the resultant set of software components from software component selection that needs to be used to develop the software system the next task is to stitch together the components in an appropriate manner. Correct and efficient reuse of software components not only depends on proper identification of software components from a system but also on the ways to merge the components. The other problem in CBSE is that of Component composition.Presently object oriented programming languages provide adhoc mechanisms for composing software using components. By providing a proper semantic foundation definition we will be able to cleanly integrate all the requirements of the software system into one system The AdHoc mechanism that is used widely in software composition is cumbersome and tedious and is prone to errors which in turns leads to wastage of time and resources. Software composition problem answers the questions like how are software components plugged into an application, In which way are the software components configured and composed.

Chapter 2

LITERATURE REVIEW

2.1 Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries

In this approach a software components document which may include anything from source code to software requirement document is used as an input to clustering algorithm. The clustering algorithm removes the useless stop words and stemming words from the document and finds and generates a set of unique words and its count in each document.This set of unique words is then used is calculating similarity measure.The similarity is calculated according to rule that if the word is present in both the documents the function returns 1,if present in either files but not both returns 0 and if not present in both the files then the value can be neglected.

The similarity matrix between two components is generated by adding the similarity value calculated between each pair of components. The cell with highest value that is the two most similar components is picked up and joined to form a cluster.The similarity matrix is then reduced and again the cell with highest value is picked to generate another cluster.This step is repeated until no further combination can be done.

2.2 A QoS ONTOLOGY - BASED COMPONENT SELECTION

In this approach only QoS or Quality of Service based characteristics which are a part of non functional characteristics are used for component selection.This approach contains two parts, first part is Qos matching process which used the concept of Quality of Service to get a subset of components according to the developer's request.In the second part a ranking process is performent on the subset of components that uses the Qos values to select the best component among the matched components.

Every component is represented using QoS Characteristics instance.There are two kinds of QoS Characteristics Service Characteristics which are high level characteristics of the component and second is Resource characteristics that shows the resources related to OS,network CPU rate etc.All characteristics contain a min Value,max Value, Process Measurement and direction.

For QoS matching a Request is taken from developer as input.The algorithms loops through all the QoS Profiles of components and values are given according to the following

rules.

1. Plugin is the most accurate match , the value is given is 2
2. Exact is the next most accurate match the value given is 1
3. If nothing matched then it is Fail

After Matching is done multiple components have matching values for which ranking is performed to get the best match for the requirement given by the developer. CRank is calculated for all the matched components and then ordered decreasingly to give out the final result.

2.3 Evolutionary Algorithms for the Component Selection Problem

Selection of a set of components from a given set of Requirements is a case of optimization problem.For optimization type of problems Evolutionary algorithms such as genetic algorithm are known to be effective.

The set of requirements are each given a fractional weight known as weighting coefficient.All these requirements are then added together to form a cost function.

The component selection problem can be defined as multi objective optimization problem having two objectives which is to minimize the cost of components and the number of components.Components are represented as chromosomes and using a initial population, mutation probability, crossover probability and number of runs which are fed to genetic algorithm component selection is performed.

Better results were found to be obtained when using smaller population sizes and smaller number of individuals.

2.4 Using Formal Methods to Construct a Software Component Library

In this Paper the author uses predicate logic to describe each individual software components.Each method is described using an interface,type declarations ,a pre condition and a post condition. After getting all the predicate logic based specification of all the software components , subsumption test algorithm is used to generate lower level hierarchy.Subsumption test algorithm uses generality principle and merge components to form a more general component using recursive comparison.

After obtaining the generalized components from subsumption based algorithm a traditional clustering algorithm is used to group similar components.The paper uses agglomerative hierarchical clustering approach since it generates a hierarchy and searching for components in hierarchy is more faster.

The similarity of two components is calculated using the number of equivalence classes between the two software components in consideration.In Agglomerative hierarchial clustering using the similarity matrix between all the components with each other the two most similar components are merges and form a new component for which again similarity values are calculated for every other component.The above steps is repeated until finally

only one component that is cluster is left. This type of clustering approach makes it easier and faster to search for components according to given requirement.

2.5 Design an Algorithm for Software Development in Cbse Environment using Feed Forward Neural Network

This approach uses Multi layer feed forward neural network to store components according to their respective repositories for accurate component identification and retrieval of components.

The algorithm takes a source document that can be a SRS document, software documentation .The requirements are then taken from user which is then divided into functional and non functional requirements.Using these requirements K means clustering algorithm is performed with word count as criteria to divide the document into clusters.These word counts are used to assign weights to keywords. Using these weights back propagation algorithm is performed which is a neural network algorithm. At the end repositories are created using the keywords.

In the results part the author compared accuracy,recall,precision values of the above method with integrated classification scheme.

This algorithm is found to have better decision making approach and best optimal component will be selected.

2.6 Software Component Recommendation Using Collaborative Filtering

This approach is little different and uses users browsing history to recommend software components.The system automatically recommends potentially beneficial software components from users based on users browsing history. Collaborative filtering is a technique to recommend items to a target user by supplying it reputation of other users who have the same interests with the target user.Collaborative filtering is known enhance content based filtering tools providing filtering of items whose content is hard to analyze.It recommends based on quality and taste.

The steps involved are

1. Gather Ratings for Components:-Store the browsing history of each user ratings for components on a database. The rating is done implicitly.Rating 1 is given if users browses the source code.When browsing history is deleted the ratings are cleared.

2. Compute Correlation:-Collaborative Filtering requires correlation between users for the purpose of speculating users similar to a user to whom the system recommend items.similarity between sessions is 1 if common component are browsed in sessions otherwise similarity is 0.Correlation is calculated between two users as similarity of ratings for components that is either voted either or neither of the two users.

3. Compute Recommendation score:-Collaborative filtering computes a recommendation score based on ratings made by users whose behaviour is similar to a reciever of recommedation.Recommedation score is an estimation value for the rating the

receiver will vote.estimated score is a weighted average of scores made by other users

4. Recommend components:-The system at last then recommends components in descending order of the scores.

2.7 Classification and Retrieval of Reusable Components using semantic features

The methodology used in this paper allows the environment to give out list of potential reuse candidates when given a problem specification.With formal verification in mind, formal specifications are used to describe problem requirements and potential solutions.A formal software specification describes the function of a piece of software free from most implementation details.For reusuability of a software component it requires modeling the desired component and also creating a library of existing solutions.for this formal specifications can be used.In the formal view a the end goal of reuse is to find an existing software component that satisfies a particular problem specification.If a matching component does not exists then a similar component is taken and adapted to satisfy the problem.

The paper in its demonstartion uses VHDL that is hardware description language.A component is specified by using the below seven clauses:-

1. requires:-mentions the input conditions for the component
2. ensures:-mentions the output conditions for the component
3. constrained by:-mentions the performance constraints on an entity
4. modifies:-mentions what the component may alter
5. based on:-mentions the data types of VHSL used
6. state:-mentions the collection of variables that represents the internal state of the component
7. includes:-indicates the shared language files needed

To efficiently compute semantic similarity, the problem and component specifications are classified by a set of (attribute, value) pairs, called features.

a component having the feature Select(intseq, int) indicates that the component has an input of type int that is a member of an input of the composite type intseq.

Feature sets are generated and stored for all of the components in the reuse database. Two components that contain a feature with the same attribute name are said to share the named feature. If the shared feature has equivalent values in both feature sets, it is a matching feature.

A shared feature indicates that some aspect of the functions performed by two components are analogous, but in terms of different domain sorts. A matching feature means that the components perform analogous functions over identical sorts. This attests to a stronger degree of functional similarity than a shared feature. The similarity of two specifications based upon their feature sets is the product of their shared feature ratio and their matching feature ratio.

Similarity is used to find a small subset of reuse candidates from a large reuse library. The candidates then undergo verification to see if they satisfy the problem specification.

2.8 Vector Space Based on Hierarchical Weighting: A Component Ranking Approach to Component Retrieval

The method used introduces a novel method of weighting keywords that takes account of where within the structure of a component the keyword is found. The author in this paper proposes a new ranking algorithm that exploits component structure by combining text information ranking techniques with the hierarchical structure analysis of components.A hierarchy weighting (HW) scheme is developed to define the relative importance of keywords found at different levels in the structural hierarchy of components.

The similarity of components and queries is defined using vector space representation of the set of index keywords.

The paper only uses Java components since a very large number of java components are available and java permits the exploration of structure of such components.

index keywords can be obtained by searching for legitimate words within the collection of names used by the programmer

in addition to extracting the keywords, it is desirable to associate a weight with a each keyword, in a each component, that reflects how much evidence it provides about component functionality

The weight given to keywords is such that if the keyword is found in jar file name it is higher than weight of keyword in class names which is higher than weight of keywords in function names.The weight is equal to the ratio of frequency of the occurrences of given keyword with the number of entities if the hierarchy level of the component.

The first, termed HW, is simply to give each component a similarity score formed as the sum of the hierarchical weights of all words appearing in the query.A more sophisticated alternative, termed VS-HW, is to adopt a vector space representation. The results demonstrate the consistent superiority of the hierarchical weighting algorithms. The vector space algorithm performed slightly better than the simple summed weight method

2.9 COTSRE: a COmponenTs Selection method based on Requirements Engineering

Requirements engineering includes elicitation, analysis and negotiation, documentation and maintenance of the requirements established for Information systems.

In order to select the most suitable component, we must take into account the rest of related requirements, especially, those non-functional requirements that impose some type of restriction in the final application (programming language, operating system, memory, etc.).

A selection matrix is made for each requirement which will contain a column for each one of the related requirements to it, and a row with each of the identified components.

A requirement can be fulfilled by a component to a certain degree. Depending on the requirement and the component features, it is possible to score the fulfillment between requirement and component. In this case, the entries of the selection matrix will have a score instead of a simple binary value. This score is calculated using the “Feature Analysis Technique”.

The author further uses weighted scoring method on the scores for the requirements obtained.

The author used this technique developed to find components for email application and selected Catalyst Internet Mail Control v4.5 as the component.

2.10 A Fuzzy-Based Approach for the Multilevel Component Selection Problem

The author in this paper proposes multilevel approach to select a set of components from a components repository which can satisfy a given set of requirements while minimizing/maximizing various criteria.The author was motivated to use multilevel approach by the multilevel structure of components and the non functional requirements such as cost of component.

A formal defintion of the problem is:- Consider SR the set of final system requirements SR={r1,r2..,rn } ans SC as the components available for selection. SC={c1,c2,..cn} Each component ci can satisfy a subset of the requirements from SR denoted SPci = {pi1 , pi2 , ..., pik } and has a set of requirements denoted SRci = {ri1 , ri2 , ..., rih } The goal is to find a set of components Sol in such a way that every requirement rj (j = 1, n) from the set SR can be assigned a component ci from Sol where rj is in SPci (i = 1, m), while minimizing/maximizing various criteria.

The algorithm followed is described as:-

1. select candidate components:-selects the components that satisfy at least one of the requirements from the SR

2. ComputeMetrics:-computes the metrics values for the Functionality and Cost quality criteria

3. FuzzyPartitionDet:-subalgorithm computes a fuzzy partition of the set Candidate components. The fuzzy sets obtained after the first splitting are A, B, i.e. the candidate components set is partitioned in two sets A, B considering Functionality and Cost metrics values

4. SelectBestComponent:-There are two different cases that may appear: all these components belong to the same cluster, or some of them are in the first cluster and the others in the second one.

   In the first case we select the component with the maximum membership degree for that class In the second case the best candidate from each cluster is identified, and then some criteria (first Functionality, then Cost) are considered to choose one of them

5. AddToSol:-adds the selected component to the final solution.

6. UpdateSR:- updates the set of SR by the dependencies appeared by selecting component c and removes the requirements that were provided by selecting component c.

2.11 Improving Retrieval Effectiveness using Ant Colony Optimization

In This paper the author introduces a new paradigm to retrieve software components which is based on ant colony optimization. The ant colony optimization algorithm is used for solving problems which are computational in nature and which can be reduced to finding good paths through graphs. it is a probabilistic technique.

Components are defined using terms like linked list and also the domain of application like banking.Synonyms of the terms are also added.

The steps for component retrieval is as follows:-

1. Keywords are identified from the user query
2. using the database created, the query keywords are compared and all the matched components are stored in a table
3. Initially every component has same pheromone value, Pheromone manipulation is done on the components retrieved.
4. the steps are again repeated till the desired component is given.

The quality of the component retrieved is dependent on four term False Positive,False Negative,True Positive and True negative.

Based on this quality value the pheromone values are updated and the component having the highest pheromone value is selected.

2.12 A Hybrid Technique for Searching a Reusable Component from Software Libraries

Formal methods is the use of techniques from formal logic and discrete mathematics in the specification ,design and construction of computer systems and software.Examples are Z,VDM,Larch,Promela etc.

The author uses Z notation to specify software components.The query is also provided in Z notation.

The paper uses a hybrid approach which uses both formal method and natural language for the library construction and for retrival of reusable software components.

Formal specifications of the software along with URL and relevant keywords in natural language are stored in the library.

The system compares query and components according to features.The attributes used to compute the similarity are:-

1. Ratio of number of keywords matched to number of keywords entered.
2. Ratio of number of keywords matched to number of keywords in a database for a component specification.
3. Declaration part of schema
4. Ratio of number of LATEX markup matched to the total number of input markup in query specification

5. Ratio of number of LATEX markup matched to the total number of markups in specifications present in the database

Matching algorithm contains the following steps:-

1. Find the specifications in the library whose keywords matched with input keywords
2. Find similarity on the basis of number of keywords matched between query and available specifications
3. Find similarity of specifications of each input method to the specifications of the every other method of that particular component schema
4. Method with highest percentage of similarity with input method is considered and its similarity is added to the overall percentage of similarity between the specifications of component
5. Overall similarity is computed on the basis of different attributes and it is placed at a proper position in descending order of similarity in a list of similar component for user
6. The above steps are repeated until no other match is found.

2.13 New Algorithm for Component Selection to Develop Component-Based Software with X Model

The author in this paper suggests a software development model known as X model in which the development process starts the usual way by requirements engineering and requirements specification.According to requirements the next step in model is to look for components which partially or fulfill the requirements.

In the first step for X model software components that can be reused.Reusuability implies generality and flexibility.The generality requirements imply often more functionality and require more design and development efforts and more qualified developers.The components should be tested in isolation, but also in different configurations.

the development processes of component-based software are separated from development processes of the components.for the component-based software development the emphasis will be on finding the proper components and verifying them.

The author proposes the following component selection algorithm which is based on Simple component based selection algorithm and criteria based component selection algorithm.

1. select user requirements, component requirements, component sets , testable component repository and reusable component repository
2. for each component in TCR and RCR
3. if UR=CR , the Best-Fit Strategy and First-Fit Strategy is used to select a subset of component and component through SCSP and CCSP, which satisfy the clients requirements.

4. else if UR=CR , user requirement matches more than 50 percent than the Best-Fit Strategy and First-Fit Strategy is used to select a subset of component and component through SCSP and CCSP, which satisfy the clients requirements with some modification in components.

5. else if UR $\neq$ CR , no solution available modify user requirements.

6. if user requirement modification is not possible develop the component from scratch.

The selection of components will be based on the following factor:-

1. Performance:-based on degree of cohesion , coupling and interface

2. Size:-size of the component should be less

3. Reliability:-The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

4. Fault Tolerance:-The ability of system or component to work for long time continuously without any hardware or software faults

5. Time:-the study proposes to use maximum number of COTS component because COTS components save the development and testing time and improve quality

6. Complexity:-The cyclomatic complexity of a section of source code is the count of the number of linearly independent paths through the source code.

The algorithm solves the component selection problem by selecting optimal set of component at each stage by using Best-Fit Strategy and First-Fit Strategy, with the hope of finding a global accepted result of the problem.The basic objective is to maximize the ratio of number of requirements to the cost of components.set of components is verified for the all the respective features, if any component is not feasible then the object are removed out and the set is reduced according to our requirements. The process is repeated until an optimal solution is obtained.

Algorithms offer significant advantages due to its ability to naturally represent qualitative and qualitative aspect of optimal set of components. Software engineer can identifies potentially optimal components from existing reusable component by using best-fit strategy and first-fit strategy to select to select a subset of component and component through SCSP and CCSP.

2.14 Intelligent Component Selection

In this paper the author proposes AI techniques to be used for classifying software components based on an ideal specification.

The author proposes to approach the selection of third party components by creating a ideal specification of the requirements for the component(s) in a particular project.The ideal specification is used to generate training data for the AI classifiers, as well as for the generation of tests for dynamic assessment.

The author considers component selection problem as a classification problem which is assinging each project to a class based on its assessed suitability to the project.

The steps proposed are:-

1. Specification of the ideal component that is needed in XML
2. shortlist of candidate components using AI techniques
3. test cases generated to test if the component is correct
4. execute the tests on the shortlisted components
5. rank the components
6. report the results

The author uses two classifiers in the namely C4.5 and a neural network.the two classifiers were trained on the same data, generated from the ideal specification.The data is generated using boundary value analysis on test case generation.C4.5 generates a tree incorporating all instances within the data set and their classifications. It then groups the data to allow 'pruning' to create a manageable decision tree.

The neural network classifier is based on a simulation of the neurons of the human brain, organised into interconnected layers. In Weka's implementation, a backwards propagation algorithm updates the weights connecting the neurons and reinforces those that result in a correct classification. . Using the XML Schema and the instance document for the ideal component, the data generator creates an internal model of the component. The training data is automatically labelled into classes, overcoming one of the difficulties with supervised learning

2.15 A Novel Approach of Components Retrieval in Large-scale Component Repositories

Automatic Tags Extraction(ATE) retrieval, is proposed in this paper. In this method, component tags are extracted automatically from application domain terms, highfrequency terms, high-weight terms and facet terms in description document of component at first, and then the improved VSM (Vector Space Mode) similarity algorithm is used to retrieve on the tags.

Firstly, component tags are extracted and indexed by the ATE algorithm, and then an improved VSM similarity algorithm is used to match the component tags.

The author argues that the application of Automatic tags extraction in the description section of software components finds more useful information and vastly improves the recall ratio, the improved Vector space method similarity also increases the precision.the ATE retrieval is base on the faceted classification scheme.The component description documents which contain more information than the faceted classification are stored as a child node in the XML documents.

The steps involved are:-

1. Query input by the user
2. Retrieval sentence segment and semantic expand
3. Functional term extracted in keywords

4. Seek intersection of functional term:-The components which meet all the functional requirements are selected as candidate set by seeking in the inverted index of functional facet

5. Similarity calculated

6. Retrieval results sorted according to the similarity

The components are described using tags.The tags which will be used are:-

- DT:-terms belonging to domain
- ATC:-terms contained in component description
- HWT:-set of high weight terms
- HFT:-set of high frequency terms
- FT:-set of faceted terms

The terms which will be included is given by :- $\{[(DT \cap ATC) \cup HWT] - HFT\} \cup FT$

Matching the retrieval keywords with the component tags, the similarity formula used in this paper is based on VSM.

The retrieval keywords and component tags are identified in vector space as weight vector V_q and V_c.The VSM similarity is represented using the cosine of the angle between the vectors.Factor like FacetBoost and Hits are added in similarity calculation.

- FacetBoost(t):-is a boost of faceted term t when t is hit. For improving the differentiation of term, different facet has different FacetBoost when a term is hit. For example, the function and quality of service facet can be set greater boost than other facet when higher requirements are needed in function and QoS. These factors can be set by user for various requirements of retrieval.

- Hits:-the components can be further rewarded through Hits when more keywords are matched. The Hits is calculated using $Hits = n_h/n_q$, where n_h is matched number and n_q is the number of all retrieval keywords

2.16 Clustering software components for program restructuring and component reuse using hybrid XNOR similarity function

The requirement for clustering comes from the need for decision making such as software component classification and component clustering, user behavior prediction, performing software component search and in component retrieval from software library. The input to clustering algorithm usually may be any set of entities or patterns or text files or images or software components. The output of clustering algorithm will be a partition of cohesive groups.

In this paper the author proposes to first obtain frequent item sets for each document using existing association rule mining algorithms either by horizontal or vertical approach. Once frequent item sets is found in each document then a a Boolean matrix is formed

with rows indicating documents and columns indicating unique frequent items from each document. The authors perform clustering using XOR similarity function.

In XNOR similarity function if the attribute is not present in both the components then the similarity value is ignored,if the attribute is present in either of the component then the value is 0 and if present in both the component then it is 1.

The complete algorithm involves the following steps:-

1. for each software document, all the stop and stemming words is removed, the count of unique words is calculated and then frequent item sets is found.
2. a word set W if formed from frequent itemsets.
3. dependency Boolean matrix is formed with document is rows and word in columns.
4. feature vector similarity matrix is calculated and similarity value is calculated using the XNOR similarity measure discussed above.
5. the corresponding cells of matrix is replaced by count of 0s in tri state feature vector.
6. at each stage document having maximum value is merged to create a new cluster.
7. the final clusters are labelled.

2.17 A Model Software Reuse Repository with an Intelligent Classification and Retrieval Technique

The author in this paper proposes a new methodology for efficient classification and retrieval of multimedia software components based on user requirements by using attribute classification scheme with genetic algorithm.

The scheme that is proposed by the author describes each potential reusable component in 9 characteristics.The scheme encodes the characteristics into a series of bit strings of 36 bits in length.The user will retrieve the desired component by specifying the required attributes of the component.The system retrieves the component which match the user specified characteristic.

There are two parts to the scheme first is classification and second is retrieval.Classification of components using genetic algorithm the author argues will result in fast retrieval of correct components.In the encoding phase the characteristics are encoded to binary form that is understood by genetic algorithm. The 9 characteristics that are taken is:-

1. Component Name
2. functionality
3. domain
4. Operating System
5. Algorithm
6. Implementation Language
7. Developer

8. Time Complexity

9. Price

The genetic Algorithm classifies the software components into homogeneous set in terms of characteristics.

In component Retrieval phase the user enters the required characteristics for the component .The system will encode the user request and then will compare with the classifiers that were generated in the previous phase.The components of the most closest matched classifier will get returned.

The Genetic algorithm used follows the following step:-

1. random population of 100 chromosomes is created.

2. for every generation crossover and mutation is performed according to set probability.

3. The component classification is performed on these 100 classifiers

4. classifier value is compared with each component, if component is close enough to the threshold then the component is assigned to the class represented by the current classifier.

5. top 20 classifiers are selected according to the number of components assigned.average fitness is calculated for these.

6. if the average fitness value of the current generation is greater than previous generation than new population generated by selecting chromosomes according to fitness values.otherwise new population is not created,

2.18 A PERSONALIZED FACET-WEIGHT BASED RANKING METHOD FOR SERVICE COMPONENT RETRIEVAL

This paper proposes a novel personalized facet-weight based ranking method for software component retrieval, which assigns a weight for each facet to distinguish the importance of the facets, and constructs a personalized model to automatically calculate facet-weights for users according to their historical retrieval records of the facet values and the weight setting.This paper exclusively deals with active services.

The components and user queries are specified under facet classification in the form of vectors,a facet weight is specified on each facet for a user to decide which facet value in his query is more important. According to the historical records, we a personalized model is constructed. The model is used to automatically calculate weights for users when it brings forward new retrieval facet values, so that the user will save much time in facet-weight setting.

A user query is brought forward on the foundation of facet classification scheme of the component repository; so it can also be formally defined based on Facet Vector same as the structure of Component Vector.

Facet matching degree is calculated between user query vector and component vector which is the inner product between the vectors.After getting the weights for each facet

from the user a general matching degree is calculated which is again inner product of facet matching degree and weight vector.

For personalized modeling two kinds of information were used :-

1. User's historical retrieval records:-a retrieval log for each user, restoring his historical retrieval records is stored. If the component user wants is not in top 10 of the resulting list,this retrieval operation will be regarded as an invalid one and will not be restored in the log.

2. Fading value:-since user's preference changes with time a fading value which is used to determine if only the latest log is to be used or previous logs will also be used.

The steps involved are:-

1. The similarity between user's current query and each record in retrieval log is calculated.

2. the facet weight vector of each record is updated by multiplication of similarity and fading value.

3. the facet weight vector is normalized after adding.

2.19 A π-Calculus Based Approach for Software Composition

In this thesis the author develops a formal language for software composition that is based on $\pi\ell$-calculus, which is a variant of π-calculus in which agents communicate by passing extensible, labeled records, or so-called "forms",rather than tuples. The π-calculus has previously been used to model various aspects of object-oriented languages.

In the L-calculus parameters are identified by names rather than positions which provides a higher degree of flexibility and extensively for software composition.

2.20 A calculus for reasoning about software composition

The author demonstrates the use of calculus to simplify compositions by limited evaluation of software compositions.The author describes calculus as convenient for expression and reasoning of software components.The author in this paper attempts to answer the questions like how are software components plugged into an application, In which way are the software components configured and composed.

The author defines a composition language as language for specifying operators for connecting the software components, attach the abstractions for modifying the component interfaces and programming scripts that initializes and connects the components.The author has developed the software names JPiccola for the implementation of composition of java components.

The view used in the paper is that, component based applications are made up of off the shelf components and scripts that plug them together.Scripts use high level connectors to coordinate the services of various components.

The author argues that for modelling software components 4 things are required:-

1. Compositional Styles:-A compositional style describes the structure of software application in terms of components, connectors and rules governing the composition.

2. Scripts:-Scripts configure and compose components using the connectors defined for a style.

3. Coordination abstractions:-Both connectors and glue code may need to express coordination of concurrent activities

4. Glue Code:-Glue code is needed to package, wrap or adapt code to fit into a compositional style

For fulfilling all the above requirements the author has developed a process calculus based on higher-order asynchronous π-calculus in which communication is done using extensible records.The calculus is named piccola calculus.

The author working upon the piccola calculus developed piccola language for composing software components that conform to a particular compositional style.The Piccola calculus is not only helpful for modeling components and connectors, but it also helps to reason about the Piccola language implementation and about compositional styles. Piccola is a small composition language that supports the requirements mentioned above and the denotational sematics of whom is defined using the terms of piccola calculus. At the lowest level a abstract machine is present that implements piccola calculus.At the second level Piccola language is present which is implemented by translation to the abstract machine, following the specification of the denotational semantics.piccola provides Piccola language, which is implemented by translation to the abstract machine, following the specification of the denotational semantics.

The third level provides a set of standard libraries to simplify the task of programming with Piccola.The fourth layer is provided by the component framework designer. At this level, a domain expert encodes a compositional styles as a library of components, connectors, adaptors, coordination abstractions, and so on.

At the last level application programmer scripts together components using the abstractions provided by the lower layers.

The Piccola calculus can help to bridge mismatches by supporting reasoning about wrappers that adapt component contracts from one style to another.

Chapter 3

METHODOLOGY

Chapter 4

RESULTS and DISCUSSION

Chapter 5

CONCLUSION AND FUTURE SCOPE

One and half page conclusion with future scope is enough.

Appendix A

Appendix Title

Bibliography