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

Pradeep Tomar, *Member, IACSIT,* and Nasib Singh Gill, *Member, IACSIT*

***Abstract*—Component-Based Software Engineering (CBSE) is an approach which is used to enhance the reusability with the development of component-based software from the pre-existing software components or with the components which is developed from the scratch. A new algorithm is proposed for component selection by using best-fit strategy and first-fit strategy through X model which is used to develop component-based software with two approaches likely development for reuse and development with reuse. But when reuse a pre-existing software component through the development with reuse, component selection play an important role. Component selection for Component-Based Software Development (CBSD) is very challenging field for researchers and practitioners. This paper also presents the two component selection problem viz. Simple Component Selection Problems (SCSP) and Criteria Component Selection Problem (CCSP). On the basis of these two problems, this paper presents a new optimal solution with new algorithm for optimal component selection from repositories. Lastly, paper summarizes the factors used in algorithm for optimal selection of components with the help of X model repositories to fulfill the requirements of client by using SCSP and CCSP.**

***Index Terms*—Component, reusability, best-fit, first-fit, X model.**

I. Introduction

CBSE addresses challenges similar to those encountered elsewhere in software engineering. Many of the methods, tools and principles of software engineering used in other types of system which will be used in the same or a similar way in CBSE. There is however one difference; CBSE specifically focuses on questions related to components and in that sense it distinguishes the process of "component development" from that of "system development with components". Components raise the level of abstraction to that which can be easily used by a domain expert who is not necessarily an expert designer and programmer. A typical development effort using components would be importing the components of interest and customizing each one it without explicit coding and finally wiring together the components to form an application. The components in software development help in increasing productivity gained by reuse of design and implementation, help in increasing reliability by using well tested code, help in lower down the maintenance costs because of a smaller code base, help in minimizing the effects of change since black box programming tends to rely on interfaces as compared to explicit programming [1]. Components are built to be used and reused in many applications. Software development with components is focused on the identification of reusable entities and relations between them, beginning from the system requirements and from the availability of components already existing. In reality the processes are already separate as many components are developed by third parties, independently of software development. Even components being developed internally in an organization which uses these very same components, are often treated as separate entities developed separately. CBSE is an approach which is used to enhance the reusability from the pre-existing software components. But when reuse pre-existing software components, components selection factors play an important role to enhance the reusability, productivity and quality [2]. So this paper presents new algorithm for components selection to suggest how to select component due to which researches and practitioners select optimal set of components from repository to fulfill the requirements of client. In the whole process CBSD, component selection is the main issue because developers want components that can satisfy all the requirements and also minimizing the overall cost [3]. Therefore there are two ways of selecting the components. In one way developers can define system architecture and according to that selecting the appropriate off-the shelf component and the other way is to select components according to the requirements and functionality. Only selecting the appropriate component is not the major issue but also developers have to check the adaptability, composition and dependency between their requirements. Each component is assigned a cost which is the overall cost of acquisition, composition and adaptation of that component. In general, there may be different alternative components that can be selected, each coming at their own set of offered requirements. Our main aim is to refine a selection approach that guarantees the optimality of the generated component system. This paper is focusing on the issues related to component selection problems and their solution by proposing new algorithm. Component selection is a crucial problem in CBSE and finding that component among all the others has been identified a major problem in CBSE. This paper presents a new algorithm, which we apply after the design phase of X a component-based software life cycle model to select a subset of component and components by using best-fit and first-fit strategy through SCSP and CCSP, which satisfy the clients'

Manuscript received March 1, 2013; revised May 3, 2013.

Pradeep Tomar is with Department of Computer Science and Engineering, School of Information and Communication Technology, Gautam Buddha University, Greater Noida-201310, Uttar Pradesh, India (e-mail: parry.tomar@gmail.com).

Nasib Singh Gill is with Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak-124001, Haryana, India (e-mail: nasibsgill@gmail.com).

requirements according to application.

II. X Life Cycle Model for CBS Development

X model [4], [5] in which the processes start in usual way by requirements engineering and requirements specification as shown in Fig. 1. In a non component-based approach the process would continue with the unit design, implementation and test. Instead of performing this activities that often are time and efforts consuming, we simply select appropriate reusable components and integrate them to build the software. However, two problems appear here which break this simplicity: 1) It is not obvious that there is any component to select, and 2) the selected component only partially fits to our overall design. The main characteristic of this software life cycle model is reusability in which software is developed by building reusable components to build the software and component-based software development from simple and reusable components by using two approaches, software development with modification in reusable component and development without modification in reusable component.

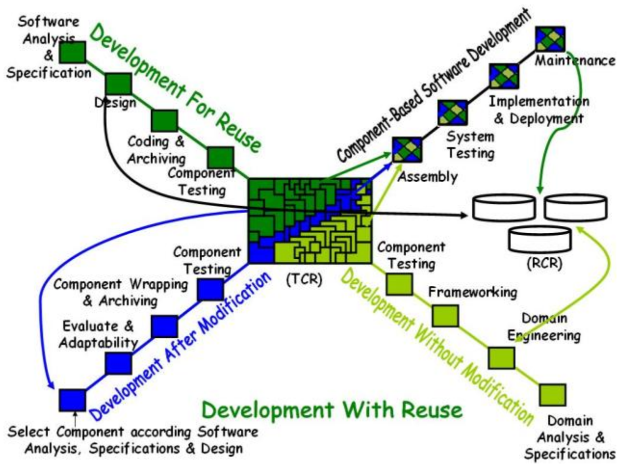
Software Analysis & Specification
Development For Reuse
Design
Coding & Archiving
Component Testing
Component-Based Software Development
Maintenance
Implementation & Deployment
System Testing
Assembly
(RCR)
Component Testing
Component Wrapping & Archiving
Evaluate & Adaptability
(TCR)
Development After Modification
Development Without Modification
Frameworking
Domain Engineering
Domain Analysis & Specifications
Development With Reuse
Select Component according Software Analysis, Specifications & Design

Fig. 1. X: A life cycle model for component-based software development.

Evolution and the production of potentially reusable components are meant to be useful in future software projects. Reusability not only involves reusing existing components in a new software system but also producing components meant for reuse. When software has been developed, the developer may realize that some components can be generalized for potential reuse in the future. Reusability implies the use of composition techniques during software development; this is achieved by initially selecting reusable components and assembling them, or by adapting the software to a point where it is possible to pick out components from a Reusable Component Repository (RCR) and Testable Component Repository (RCR). This model have two main phases, first one is building reusable components for software development and second one is building software from simple and reusable components [4]-[6] of X model which help in developing a quality component-based software with the help of reusability. Therefore to select the component, we propose an algorithm which is used for selection of optimal set of component after completion of software analysis & specification and design phase.

A. Building Reusable Components for Software Development

A component is built to be reused and reusability implies generality and flexibility, and these requirements may significantly change the component characteristics. The development process of building components can follow an arbitrary development process model as shown in Fig. 1. The generality requirements imply often more functionality and require more design and development efforts and more qualified developers. The component development will require more efforts in testing and specification of the components. The components should be tested in isolation, but also in different configurations. Finally the documentation and the delivery will require more efforts since the extended documentation is very important for increasing understanding of the component.

B. Building Software from Reusable and Testable Components

The main idea of the component-based approach is building systems from pre-existing components as shown in Fig. 1. This assumption has several consequences for the system lifecycle. First, the development processes of component-based software are separated from development processes of the components; the components should already been developed and possibly used in other products when the system development process starts. Second, finding and evaluating the components. Third, the activities in the processes will be different from the activities in non component-based approach; for the component-based software development the emphasis will be on finding the proper components and verifying them, and for the component development, design for reuse will be the main concern. X model has two phases during CBSD and two component repositories. In this paper we present the algorithm to select the optimal set of component from TCR and RCR according to the requirement of client through X model. According to the Fig. 1, after completing the software analysis & specifications, and design, developer check the RCR and TCR to select the optimal component.

III. Component Selection Problems

In component selection, a number of software components selected from a subset of components or from component repository in such a way that their composition satisfies a set of objectives. Components selection plays an important role in CBSD. Component selection decisions are often made in an ad-hoc manner. Component selection processes have been proposed to improve upon the efficiency and effectiveness of informal methods. Existing selection processes do not fully address the specification and evaluation requirements. Component selection problem is the most important task in development of component-based software from testable and reusable component because the developed component-based software depends on the selected components. So there are various methods and algorithm to search the optimal set of components from repositories. Researchers and practitioners generally use the simple component selection problem and criteria based component selection problems. [7]-[9]

A. Simple Component Selection Problem

Simple Component Selection Problem (SCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives [7], [8].

Consider *SR* is the set of the *System Requirements* as

$$SR = \{r1, r2, \ldots\ldots\ldots\ldots\ldots\ldots, r_n \},$$

SC is the *Set of Components* available for selection as

$$SC = \{c1, c2, \ldots\ldots\ldots\ldots, c_m\}.$$

Each component *ci* can satisfy a subset of the requirements from SR,

$$SR_{ci} = \{ri1, ri2 , \ldots, ri_k\}.$$

The goal is to find a subset of components Sol (solution) in such a way that to every requirement *rj* from the set *SR* can be assigned a component *ci* from Sol where *rj* is in SR_{ci}.

B. Criteria-Based Component Selection Problem

Criteria-Based Component Selection Problem (CCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives and using various criteria. The goal is to find a set of components Sol in such a way that to every requirement rj from the set SR can be assigned a component ci from Sol where rj is in SRci, while minimizing the number of components in the solution Sol and/or while minimizing ci € Sol cost (ci). Another criterion for selection of components in CCSP is dependencies involved between the components [7], [8].

IV. Proposed Algorithm for Component Seclection

Here the solution for the component selection problem has been presented with proposed algorithm, when developers search the component after requirement analysis and design phase of X Model from TCR and RCR. Here set of components will be addressed as set of component, where set of component will be used to satisfy set of requirements. This algorithm also used the *Best-Fit Strategy* and *First-Fit Strategy* to search the component and set of components through SCSP and CCSP.

```
Step1:  Begin
Step2:  Select (UR, CR,CS, TCR and RCR)
        UR = User Requirements,
        CR = Component Requirements,
        CS = Component Sets,
        TCR = Testable Component Repository,
        RCR = Reusable Component Repository,
Step3:  TCR = {1 to n}
        RCR = {1 to n}
// n is the numbers of components available in repository
Step4:  Sol=Ø
// Sol=Solution and at this time, we have no optimal component solution according to the client requirement.
Step5:  Sum=0
// Sum of components in the solution set
Step6:  for i ← 1 to n
// i is the index number of component in repositories (TCR and RCR)
Step7:  If UR = CR
// If User Requirement (UR) is equal to the Component Requirement (CR) of Component Sets (CS) available in component repositories. Than we use the Best-Fit Strategy and First-Fit Strategy to select a subset of component and component through SCSP and CCSP, which satisfy the clients' requirements according to the following factors* .
Step8:  Return Solution (Sol)
Step9:  Sol← CS
Step10: Sum← Sum + CS
Step11: Return "Sol" with or without modification
Step12: ElseIf UR = CR
// If User Requirement (UR) match more than 50 % to the Component Requirements (CR) of Component Sets (CS) available in component repositories. Than we use the Best-Fit Strategy and First-Fit Strategy to select a subset of component and component through SCSP and CCSP, which satisfy the clients' requirements according to the following factors* .
Step13: Than development of component with modification and repeat the step9 to step11.
Step14: ElseIf UR ≠ CR
// If User Requirement (UR) is not equal to the Component Requirements (CR) of Component Sets (CS) available in component repositories.
Step15: Return "No Sol" than
Step16: Call Modify Requirements (UR, CR, Result) and repeat the step2 to step11.
Step17: ElseIf no change in requirement than go to step18
Step18: Than developed the components from scratch
Step19: End
```

*Find the optimal set of components according to the client-specific requirement by applying these factors like; Performance[A], Size[B], Reliability[C], Fault Tolerance[D], Time[E], Complexity[F] [10].

A. Performance

Performance of software component very helpful for in maintaining the quality of software during software development, so performance is a main factor during the selection of component. The degree to which a system or component accomplish its designed function within given constrains such as accuracy, availability, efficiency, response & recovery time, resource usage, speed etc. [11]. Performance can be increased by selecting those components which contains high cohesion, less coupling and less number of interfaces of components according to the following equation.

Performance $\propto$ 1/Interface $\propto$ Cohesion $\propto$ 1/ Coupling

We know IDC= #I/ #I_{Max} {I=Actual Interactions, I_{max} = Maximum Available Interactions, IDC= Interaction Density of a Component}

So I_{Max} = #I/ IDC

for $i \leftarrow 1$ to n

$X \leftarrow$ Select the components set which contains minimum $I_{Max.}$

B. Size

Components size depends on programming language and components code may be written in high level or low level language. User wants the size of system should be less. So researchers and practitioners select those components which use high level language. It means select those components which depend on high level language according to the following equation in which size and high level language are inversely propositional to each other.

Size $\propto$ 1/ High Level Language
LOC$\propto$1/ High Level Language
For $i \leftarrow 1$ to n
$X \leftarrow$ Select the components set which has written in high level language where as possible.

C. Reliability

The ability of a system or component to perform its required functions under stated conditions for a specified period of time [12]. Reliability can be measure in terms of reliability metrics MTTF, MTTR, MTBF, ROCOF and availability. This paper gives here main focus on software availability. Software availability means a program should be accessible according to requirement at a given point of time. So reliability improves the performance and availability of the components according to the equation in which reliability and availability are directly propositional to each other.

Reliability$\propto$ Availability

Availability= MTBF/ (MTBF+MTTR) = MTTF/MTBF

//MTBF= Mean Time Between Failure, MTTR= Mean Time to Repair, MTTF= Mean Time to Failure.

D. Fault Tolerance

The ability of system or component to work for long time continuously without any hardware or software faults [11]. Fault tolerance can be increased by increasing the Mean Time to Failure (MTTF) according to the following equation in which fault tolerance and MTTF directly propositional to each other. Fault-tolerance or graceful degradation is the property that enables a system to continue operating properly in the event of the failure of some of its components.

Fault Tolerance $\propto$ Mean Time to Failure

E. Time

Software development activities require more time for best software quality. But client always want to reduce the development time. So this study proposes to use maximum number of COTS component because COTS components save the development and testing time and improve quality according to the following equation in which time and COTS are inversely propositional to each other.

Time $\propto$ 1/COTS

F. Complexity

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

CC = E − N + 2P// *CC* = Cyclomatic Complexity, *E* = the number of edges of the graph, *N* = the number of nodes of the graph, *P* = the number of connected components.

This 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. This algorithm produces an optimal solution that approximates a global optimal solution. This algorithm will neglect the fact that a component which can satisfy the most requirements may not be suitable for heuristic choice since it may not be good in the field of cost. The basic objective is to maximize the ratio of number of requirements to the cost of components. So if we have to select a component whose cost is 5 and fulfills the 10 requirements and the other component has a cost 10 fulfills the 15 requirements, so we will select the first component because it is decreasing the cost. Thus 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.

V. Conclusion

Selection of optimal set of components not only improves selection process of component but also has a positive impact on the searching a good quality software component for software development by using X model. This algorithm helps in finding the optimal set of component according to the user demand. 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, which satisfy the clients' requirements according to the factors discussed above. In our future work, we implement this proposed algorithm to reduce the time and cost of development and develops component-based software by searching the optimal set of components from X model repositories according to the client requirements.

References

[1] C. Krzysztof and U. Eisenecker, *Generative Programming: Methods, Tools and Applications*, Addition-Wesley, ISBN 0201309777, 2000, pp. 25.

[2] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th Edition, McGraw Hill, 2006.

[3] M. R. Fox, D. C. Brogan, J. Paul, and F. Reynolds, "Approximating Component Selection," in *Proc. the 36th Conference on Winter Simulation Conference*, 2004, pp. 429-434.

[4] N. S. Gill and P. Tomar, "X Model: A New Component-Based Model," *International Journal of Engineering and Technology*, vol. 1, no. 1-2, pp. 1-10, 2008.

[5] N. S. Gill and P. Tomar, "Verification & Validation of Component with New X Component-Based Model," in *Proc. 2010 International*

Conference on Software Technology and Engineering, San Juan, Puerto Rico, USA, vol. 2, 2010, pp. 365-371.

[6] P. Tomar and N. S. Gill, "Software Component Technology: An Easy Way to Enhance Software Reusability," in *Proc. 94th Indian Science Congress*, Annamalai University, Tamilnadu, India, 2006, pp. 18-19.

[7] A. Kaur and K. S. Mann, "Component Selection for Component-Based Software Engineering," *International Journal of Computer Applications*, vol. 2, no. 1, 2010, pp. 109-114.

[8] A. Vescan, "Pareto Dominance - Based Approach for the Component Selection Problem," *Second UKSIM European Symposium on Compute*, 2008, pp. 58-63.

[9] N. Haghpanah, S. Moaven, J. Habibi, M. Kargar, and S. H. Yeganeh, "Approximation Algorithms for Software Component Selection Problem," in *Proc. Asia Pacific Software Engineering Conference*, 2007, pp. 159–166.

[10] A. Kumar, P. Tomar, N. S. Gill, and D. Panwar, "New Optimal Process for Selection of Software Components," in *Proc. 1st National Conference on Next Generation Computing and Information Security, jointly organized by Computer Society of India and IMS*, Noida, U.P., India, 2010, pp. 376.

[11] IEEE Standards Board, "IEEE Standard Glossary of Software Engineering Terminology," *Computer Society of the IEEE*, 1990.

[12] B. Boehm, "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, 1986, pp. 14-24.

Pradeep Tomar is working as faculty member in the Department of Computer Science and Engineering, School of Information and Communication Technology, Gautam Buddha University, Greater Noida, Uttar Pradesh, India since its inception in the year 2009. Dr. Tomar earned his Doctorate in Computer Science in the year 2011 under the supervision of a renowned academician and researcher - Professor N. S. Gill from Department of Computer Science & Applications, Maharishi Dayanand University, Rohtak, Haryana, India. Before joining Gautam Buddha University he worked as Software Engineer in a multi-national company, Noida and lecturer in M. D. University, Rohtak, Haryana and Kurukshetra University, Kurukshetra, Haryana. Dr. Tomar has good teaching, research and software development experience as well as vast administrative experience at university level on various posts like research coordinator, examination and admission coordinator, time table coordinators, proctor and hostel warden. Presently he is also working as a Treasure in CSI, Noida Chapter. Dr. Tomar is also a member of IEEE, IEEE Computer Society, Computer Society of India (CSI), Indian Society for Technical Education (ISTE), Indian Science Congress Association (ISCA), International Association of Computer Science and Information Technology (IACSIT) and International Association of Engineering (IAENG). He served as a reviewer of journals and conferences and worked as advisory board members in national and international conferences. Dr. Tomar has qualified the National Eligibility Test (NET) for Lecturership in Computer Applications in 2003, Microsoft Certified Professional (MCP) in 2008, SUN Certified JAVA Programmer (SCJP) for the JAVA platform, standard edition 5.0 in 2008 and qualified the IBM Certified Database Associate - DB2 9 Fundamentals in 2010. Two books "Teaching of Mathematics" and "Communication and Information Technology" at national levels have been authored by Dr. Tomar. Dr. Tomar has been awarded with Bharat Jyoti Award by India International Friendship Society in the field of Technology in 2012. He has been awarded for the Best Computer Faculty award by Govt. of Pondicherry and ASDF society. His biography is published in Who's Who Reference Asia, Volume II. Several technical sessions in national and international conferences had been chaired by Dr. Tomar and he delivered expert talks in FDP, workshops, national and international conferences. Three conferences have been organized by Dr. Tomar: one national conference with COMMUNE group and two international conferences, in which one international ICIAICT 2012 was organized with CSI, Noida Chapter and second international conference 2012 EPPICTM was organized in collaboration with MTMI, USA, University of Maryland Eastern Shore, USA and Frostburg State University, USA at School of Information and Communication Technology, Gautam Buddha University, Greater Noida, Uttar Pradesh, INDIA. Apart from teaching, researches in the areas of Software Engineering are being guided by Dr. Tomar. He has also contributed more than 74 papers/articles in national/international journals and conferences. His major current research interest is in Component-Based Software Engineering.

Nasib Singh Gill is a professor and head in the Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak, Haryana, India and is the senior most faculty working in the Department since its inception in the year 1990. Besides, he is also currently working as Director, University Computer Centre as well Director, Maharishi Dayanand University Alumni. Professor Gill is working as Head of the Department since 2006. In addition to this, Professor Gill has been Director, Directorate of Distance Education from 2005 to 2007 and is known for its major reforms in Distance Learning. He also has been Director, Public Relations from 2007 to 2008. Before joining the University in 1990, he worked as a Programmer in Haryana State Electronics Development Corporation Limited - A State Government Undertaking (HARTRON), Chandigarh, INDIA and acted as Incharge of Computer Center at Electronics Research Development & Facilities Center (ERDC), Gurgaon, Haryana, INDIA. Professor Gill earned his Doctorate in Computer Science in the year 1996 under the supervision of a renowned academician and researcher - Professor P. S. Grover of Delhi University, Delhi, INDIA. Prof. Gill carried out his Post-Doctoral research at Brunel University, West London, U. K. during 2001-2002. Dr. Gill is recipient of Commonwealth Fellowship Award of British Government for the Year 2001. Besides, he also has earned his MBA degree. He has published more than 145 research papers in National & International Journals, Conference Proceedings, Bulletins, Books, and Newspapers. He has authored three popular books, namely, 'Software Engineering', 'Digital Design and Computer Organization' and 'Essentials of Computer and Network Technology'. Professor Gill is a fellow of several professional bodies including IETE. He has been awarded with 'Best Paper Award' by Computer Society of India in the year 1994 for contributing the best paper "A New Program Complexity Measure" in their Journal. Professor Gill is presently guiding researchers in the areas - Measurement of Component-Based Software Systems, Complexity of Software Systems, Component-Based Testing, Data Mining & Data Warehousing, and Natural Language Processing.