I J C T A, 9(20), 2016, pp. 15-18
© International Science Press

A New Perspective of Measuring The Supererogatory Functionality in Software Components

Jyoti Sharma*, Arvind Kumar and Anil Kumar*****

ABSTRACT

A number of component selection techniques have been designed. But the Component selection with its pleonastic functionality in component based software engineering based on the concept of packages or header files is a new area. The parameters maintainability, complexity and reliability and performance have to be measured. Keeping in view all these issues in this research report we are measuring the supererogatory functionality of the component with the help of our previously defined metric. This research work will presents an approach for assessing the reusable components in a different way which results a number of peerless and flawless components.

Keywords: componentcomplexity, coupling, cohesion, peerless, flawless, pleonastic, supererogatory

1. INTRODUCTION

CBSE is a basically a traditional approach but now a days it is requiring the new technologies. If the most favourable and the gilt-edge components are selected then it will automatically affect the various parameters concerned with the software.CBSE consisting of the “purchase, don’t make philosophy” due to the again and again use of the component, the component based software engineering is quite different from the traditional waterfall approach as it emphasizes on developing new software from prebuilt components. Various characteristics of the component are independent, compo sable, and deployable. A large number of models, metrics and algorithm have been proposed for the selection of components from the component repository but this time this is the new concept and can be useful further with some extension.

Capretz L.F. Suggested a process model provided conceptual dovetailing across all phases of model. The software concepts remain the same from implementation to maintenance. The research also focused on problem faced while selecting component from repository. Reusable libraries usually large that creates problem to find potentially reusable components. Component selection becomes more difficult when discordance in terminology i.e. the candidate component in library described by unfamiliar or irrelevant terminology [1].

Dellarocas C. researched that the traditional software development approaches were not optimal for developing the complex software applications using the reusable software components than component-based software development approaches. Traditional approaches tried to develop the system through scratch. It means, this approach followed the software development approach to build the software system. This paper described a new approach to build the complex software system. So researcher introduced new viewpoint which was known as Synthesis. Synthesis was a software development environment that tries to decrease the human effort for development of independent software components. After development integrate all independently developed software components for creating new applications [3].

* Ph. D [CSE] Research Scholar, SRM University, Delhi-NCR, Sonepat, Haryana, India, *Email: kaushik.jyoti27@gmail.com*

** Department of Computer Science & Engineering, SRM University, Delhi- NCR, Sonepat, Haryana, India, *Email: k.arvind33@gmail.com*

*** Ph.D [CSE] Research Scholar, SRM University, Delhi-NCR, Sonepat, Haryana, India, *Email: Anilarora382@gmail.com*

Crnkovic I. et al. investigated that component based software engineering approach cannot be amply utilized if development process and development organizations were not adopted according to basic principles of CBSE. By adjustment in development processes can achieve the aim of increased reusability of existing components, the efforts for implementations decrease and system verification efforts increase.

Generally the basic steps employed in component based freeware systematizing are as follows:-

In our proposed work we are measuring the supererogatory functionality of component and its metric will get related to the concept of coupling and cohesion. Whenever we are considering the white box reuse then it will get very troublesome or recalcitrant of a component selection. A component is basically having a large amount of functionality wrapped up in between that component. Our concern is to find out the asymmetric functionality in a component so that the component selection for reuse will get easier and for this purpose a new metric has been designed in our work based on a concept of coupling and cohesion.

The rest of the paper is organized as follows. Fragment II gives the general steps which we were followed for the component selection or for the best component reuse and also explains the measuring of supererogatory functionality of components with previous defined metric. In the section III we have the conclusion and future scope.

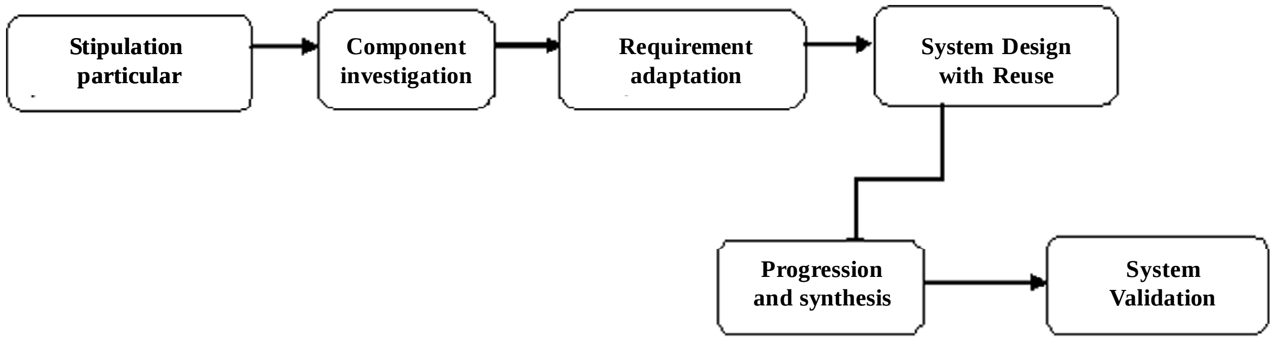

Figure1: Basic steps in Component Based Software Engineering

2. THE FLAWLESS SELECTION OF COMPONENTS

Some of the basic steps which we were followed in our previously defined algorithm are:

1) Congregation of components which may be in any type of source code I.E object oriented code(generally we are considering the C,C++ and java code)
2) Various parameters are calculated.
3) Apply the required steps.
4) Select the components or the segment with the congenial or the capital values of SFM with the help of speculation analysis.

Above defined steps will work in component based software engineering. Various parameters will be defined as:-

Let C denotes the component from the component repository

EM(C, S): It is the set of exorbitant methods of component C that delivers the functionality for the usage scenario.

VM(C, S): It is the set of vital methods that delivers the functionality for the usage scenario.

LM(C, S): It is the set of lingering methods.

DUM: It is the set of all direct connections between classes and methods.

NDIUC: It is the number of used direct or indirect connections in the case study.

NDIUC= m (m-1)/2

PC3M: Package cohesion component complexity metric [5]

PC3M = (DUM)/NDIUC

EM(C, S) = VM(C, S) – LM(C, S)

Now the next thing is to calculate the set of needed methods for a particular usage scenario so that after this calculation the set of supererogatory methods will get easily calculated.

Finally the metric for the supererogatory functionality will be as:-

Supererogatory functionality metric:

$$\boxed{\text{SFM = summation of (PC3M (EM(C, S)))}}$$

Now we will consider the calculation of SFM for the single components with their measurement of supererogatory functionality as follows:-

Considering the Circle area program as component C1 as single component as:-

Here, total numbers of components or packages in the given example are = 3

So the value of m = 3

Now the calculation of NDIUC will be performed which is as follows:-

NDIUC = m (m-1)/2

= 3 (3-1)/2

= 3

DUM = 2(as number of direct connection between classes and methods is 3)

So the value of NM(C, S) will be 2 as it is supposed to be equal to the value of DUM and the number of direct connection between classes and methods are 2)

RM(C, S) is 3 as it is supposed to be equal to NDIUC that is total number of direct or indirect connection between classes and methods.

EM(C, S) = VM(C, S) – LM(C, S)

= 3-2

= 1

Therefore SFM = 2/3(1)

= 0.6

Similarly the SFM value can be calculated for the various number of components and the component selection process or the complete process of software reuse will get more efficient and easier.

3. CONCLUSIONS AND FUTURE WORK

This paper analysis the various techniques for measuring the extravagant functionality of a component by selecting the individual components and calculating their set of values. During the optimal selection of components if the value of SM(C, S) is calculated then it will be very useful approach for selecting the quality components for the development of software.Our results verified its integrity and operability Now by using this above approach selecting a set of components to satisfy a set of requirements while minimizing the cost is becoming more possible.

REFERENCES

[1] Capretz L.F., " A software Process Model for Component-Based Development ", Information Technology Journal 3 (2), pp. 176-183, 2004

[2] Mao M., Jiang Y., "A New Component Based Configuration Management 3C Model and its Realization", International Symposium on Information Science and Engineering, IEEE, pp. 258-262, 2008

[3] Dellarocas C., "The SYNTHESIS Environment for Component-Based Software Development",in Proc. of 8th Int. Workshop on Software Technology and Engineering Practice, London UK, 1997.

[4] Crnkovic I., Larsson S., Chaudron M., "Component Based development process and component life cycle", journal of computing anf information technology, pp. 321-327, 2005

[5] Jyoti.S and Arvind.K ' "Design a new software metric in order to increase reliability" , IJIACS,Volume 4,September 2015.

[6] V.Swathy and P.Suresh' "A Dynamic approach for the retrieval of software components using genetic algorithm", IEEE Transactions on software Engineering, 978-1-4799-8353-7,2015.

[7] William B.Frakes and kyo kang, "Software reuse research:status and future",IEEE Transaction on software engineering, vol 31,no.7,july,2005.

[8] Schmauch Chales H.,ISO 9000 For Software Development: Revised Edition,Wisconsin ASWC Inc.,1995.

[9] P. Niranjan , C.V Guru Rao, "A Model Software Reuse Repository with an Intelligent Classification and Retrieval Technique", Scientific and academic publishing,15-21,2011

[10] H.Nima , M.Shahrouz , H.Jafar and K.Mehdi, "Approximation algorithms for software component selection",IEEE Computer Society, 1530-1362, ASPEC 2007.

[11] L.Yu, Kai.C and Srini.R, "Multiple Parameter coupling metrics for layered component based software" ,University of Arkanas at little rock,USA,2008.

[12] A.Yadav and RA.Khan, "Class Cohesion Complexity Metric" ,Babasaheb Bhimrao Ambedkar University,ICCCT,2011

[13] J.AWhittaker, "Software Invisible Users, IEEE Software, vol. 18,*pp. 84-88*, june 2001.

[14] E Da-wei, The software complexity model and metrics for object oriented. School of computer Engineering, Jimei University,Xiamen, China, 16-18 April 2007,464-469.

[15] J.Gao,M.C.Shih. A Component Testability model for verification and measurement [C]// Proceedings of the 29th annual international computer software and application confrences(COMPSAC '05), San Jose,USA ,IEEE,2005:0730-3157/05.

[16] Capers Jones, Applied software measurement: Global Analysis of productivity and quality, McGraw-Hill, New York,2008.

[17] Hu Xiuwen, Tian Zhonghe, Research on Software Metrics[J].Information Technology, 2003, 27(5) :58-60

[18] S.Mahmood and R. Lai, "A complexity measure for UML component-based system specification."Software-Practice and Experience,John Wiley and sons,2008, 38(2),117-134.