



Optimal solution of software component selection by using software metric

¹Hema Gaikwad, ²Ms. Prafulla Bafna

Assistant Professor

Symbiosis Institute of Computer Studies and Research
Symbiosis International University, Pune, Maharashtra, INDIA

Abstract: The base for component base software engineering (CBSE) is the reusable code/architecture (component) and secondly save the development cost, time and efforts of our resource. This CBSE process consists of four phases qualifying the component, adapt the component, compose the component and update the component. The CBSE concept is derived from iterative Spiral process model. The first most challenging part of CBSE is component selection. Generally the stakeholders requirements are not constant they vary from stakeholder to stakeholder, so it become very difficult to select the component which exactly fulfill the requirements. In very few cases we directly use component otherwise they requires some amount of modification. This paper explores the idea of component selection by using software metric, we know that software metric represents quantitative measurement. There are two categories of software metric Direct metric and Indirect metric. Direct metric includes cost, effort, LoC, response time, resource usage and so on. Indirect metric includes functionality, User-friendliness, Usability, efficiency, ease of use, quality and testability. The paper focuses on Indirect testability metric for software component.

Keywords: CBSE, Software component, Re usability, LoC

I. Introduction

Software is a set of instructions or programs and Engineering is to build something. Software Engineering is a process through which we can develop the software. Pressman stated that Software Engineering is a systematic, disciplined quantifiable approach to the development, operation and maintenance of software; that is the application of engineering to the software [1]. There are number of models are available for development of any software and all the process model base on the principle of PDCA (Plan Do Check and Act) Cycle. Plan means define your objectives, goals and the plan of how you will achieve that goal. Do means execution accordingly. Check or Test means to make sure that we are moving according to the plan and getting the desired result. Act means during the check cycle if any issues are there then take appropriate action accordingly and revise your plan. Generally Developers and other stakeholders of the project do the “Plan, Do and Act” while tester do the “Check” part. Examples of process models are System Development Life Cycle (SDLC), Rapid Application development (RAD), Spiral model and Component base assembly model etc.

System development life cycle (SDLC) is the base for all process models. When we develop any application first we decompose it into small modules or units this process is known as work break down structure. It is the method through which we can easily develop as well as maintain the software. The phases of SDLC are as follows -Requirement engineering, Feasibility study, Planning, Analysis and Design, Code, Verification & Validation and Implementation. Requirement engineering is the first phase of SDLC. We collect the requirement using various tools such as Interview, Questionnaire, Observation and Review of documents and then segregate the requirements into Functional Requirements (WHAT Requirements) and Non functional Requirements (HOW Requirements). Feasibility Study identifies, describes and evaluate the requirements and tell whether this candidate system is developed or not. Planning includes project course of action and determining what is to be done to meet the goals. Analysis & Design means study of the previous/old system and Design means process of developing the technical and operational specification of a system for implementation. Quality assurance and quality control activities represent verification and validation activities. When we are in implementation phase we deploy system to client. The component based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of the software. However the component based development model constructs applications from pre packaged software components. Component base model is associated with Component based software engineering. Component-based software engineering (CBSE) is used to develop/assemble software from existing components [6]. It also leads the software reuse and re usability. Software reuse, the use of existing software artifacts or knowledge to create new software, is a key method for significantly improving software quality and productivity. Re usability is the degree to which a thing can be reused [2]. The integration or assembly of the already existing software components accelerates the development process. Many virtual components libraries are available on the web.

Component integration becomes easy when we select the right component. If the right components are chosen, the integration aspect is made much simpler.

II. Analysis Study

Process Model Name	When to Use	Features	Disadvantages
Software Development Model	Requirements are clearly given	1. Easy to use. 2. Easy to implement.	1. Sequential process. 2. Backtracking is not allowed.
Component Assembly Model	Component based model should be used when a particular application is risk based and when there are so many reusable component are present in that case this is the most important and more useful.	1. Reduced development time. 2. It is a pure re usability model. 3. It is the enhancement of spiral model that's why it also includes risk analysis, which is very important. 4. Increased quality and productivity.	NA

Table 1

III. Re usability Model or Component Base Model

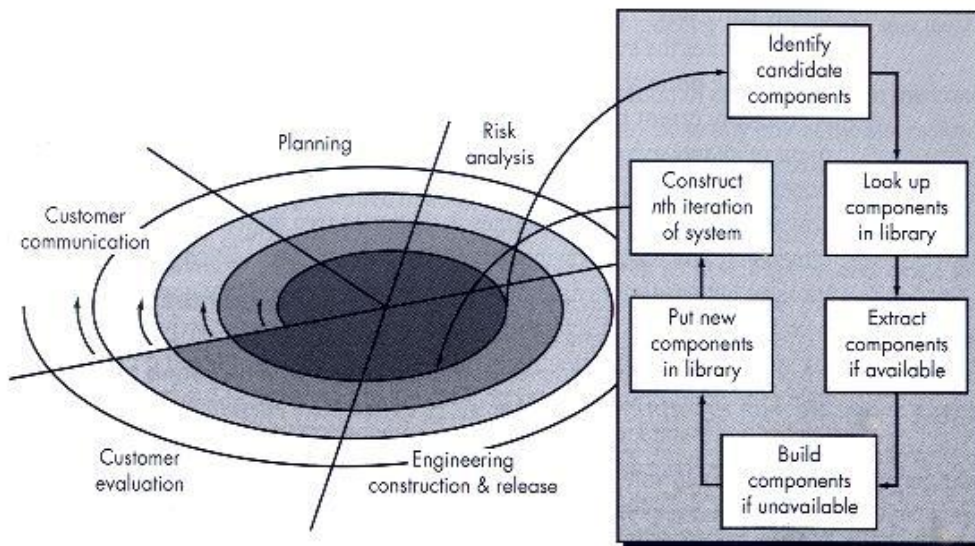


Figure 1

The above model shows component base model. The diagram has two different parts. The left one represent the spiral model but the right one represents the virtual library. In a spiral model we have various spirals and always move in clock wise directions. Inner spiral represent the concept development area and the last spiral represent the Maintenance and future enhancement area. It is the development model and consists of Customer communication, planning, risk analysis, Engineering construction & release and last is customer evaluation. In customer communication we collect all requirements with the help of various fact finding tools such as interview, questionnaires, on site observation and review of documents. In planning phase prepare plan document which includes estimation process, schedule details and resource details etc. Risk analysis is the most important, attracting and unique feature in the spiral model. Engineering includes analysis, designing and construction activity. When we construct any application first we prepare WBS (work Breakdown structure) and get to know the information about modules. After decomposition we start searching the components if it is available we use it and if it is not develop new component and keep in library for future reference.

IV. Proposed re usability model and software metric

Ronald J. Leach stated that “software reuse” is a situation in which some software is used in more than one project. Software is defined loosely as one or more items that are considered part of an organization's standard software engineering process that produces some product [5]. Capers Jones stated that we can reuse the data, architecture, design, programs and module also [5]. Component-based software development is a collection of process that deals with the systematic reuse of existing components often know as commercial off-the-shelf (COTS) and assembling them together to develop an application rather than building and coding overall application from scratch, thus the life cycle of component-based software systems is different from that of the traditional software systems. In general, analysis and design phases for component-based software

development process models take more time than traditional ones and much less time is spent in development, while testing occurs throughout the process [7]. Everybody knows that component based software engineering (CBSE) is very useful concept. Re usability helps to save time, efforts and cost. Following table shows the SDLC phase and corresponding components.

SDLC phase	Components
Requirement Engineering	SRS, RTM
Feasibility study & Planning	Feasibility report & plan document
Modeling	Design document
Coding	Code
Verification & Validation	Verification & Validation document
Deployment	Final Report
Maintenance & Post Maintenance	Maintenance & Post Maintenance report

Table 2

Proposed model says that if we start the component searching from first phase of SDLC, instead of checking only in coding phase we can minimize the TCE (time, cost and effort).

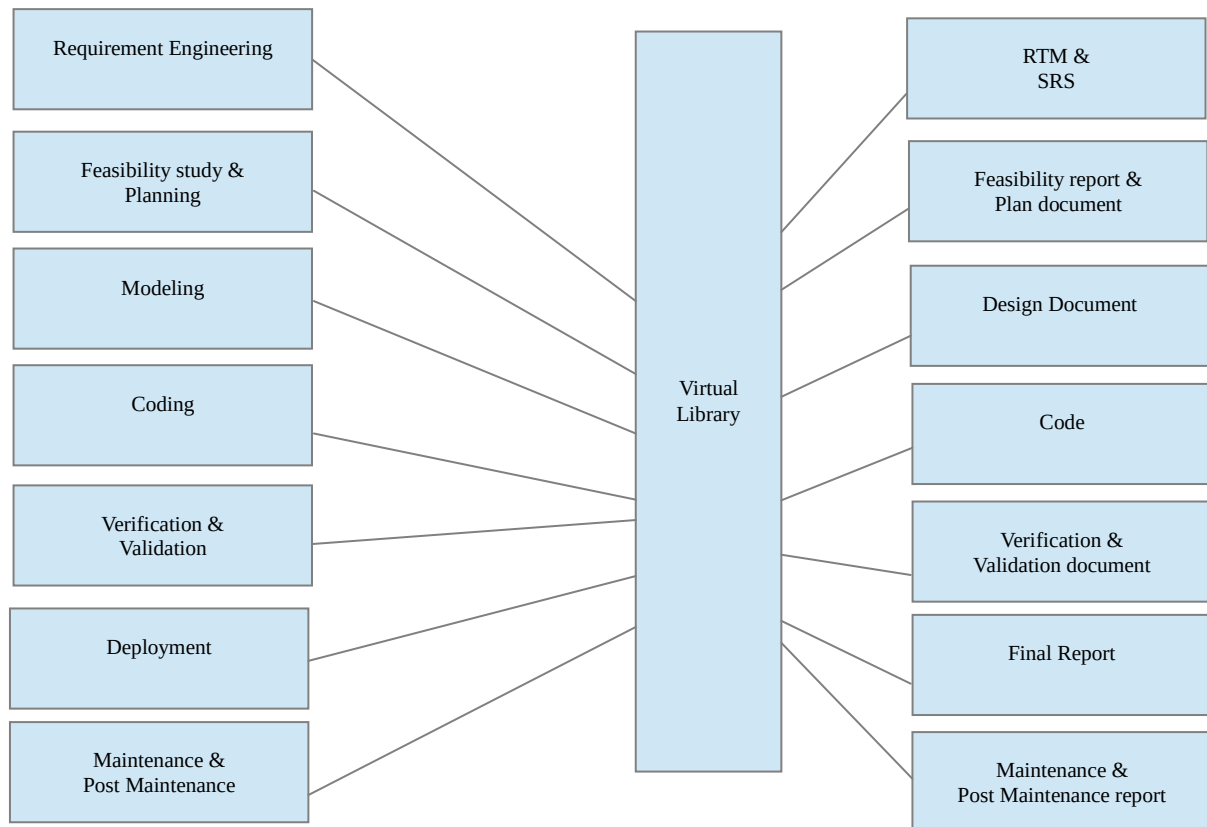


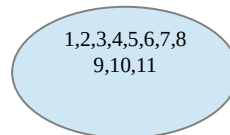
Figure 2

Figure 2 shows the virtual library mapping with SDLC phases and its components. The most challenging part is component selection, as we know that software metric helps in this regards. The paper focuses on the use of Indirect testability metric using control flow graph. Assume that we are searching components for coding phase of SDLC. Suppose P is problem, C is component, L1, L2 and L3 are languages. Source code is the input for control flow graph and output is the numeric value. The steps of Indirect testability metric using control flow graph are as follows:

1. Input any source code
2. Draw control flow graph
3. Find out edges (e) and nodes (n)
4. Apply formula Cyclomatic Complexity $CC=e-n+2$

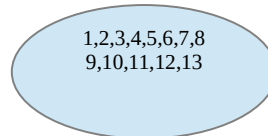
5. Get output (numeric value).

Control flow graph for L1

**Figure 3**No of nodes $n=01$ No. of edges $e=0$

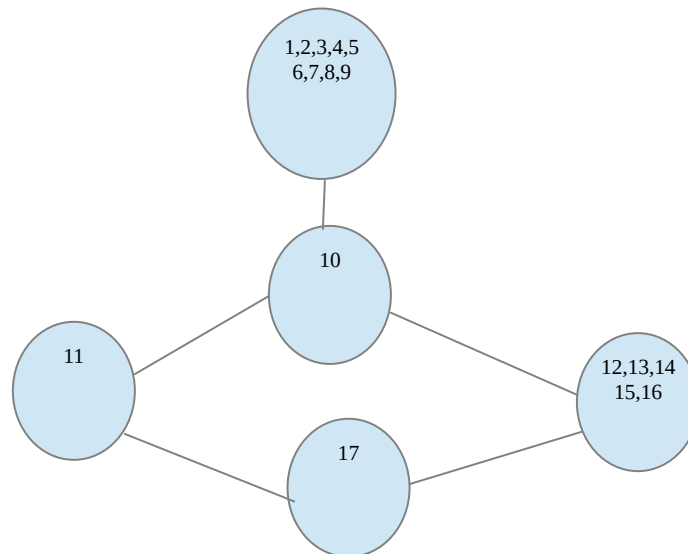
CC=03

Control flow graph for L2

**Figure 4**No of nodes $n=01$ No. of edges $e=0$

CC=03

Control flow graph for L3

**Figure 5**No of nodes $n =05$ No. of edges $e =05$

CC=02

Language	Component Cyclomatic Complexity
L1	03
L2	03
L3	02

Table 3

The above table shows that component complexity in L1 is 03, meaning that write three white box test cases to test the component, secondly component complexity in L2 is 03 meaning that write three white box test cases to test the component and lastly component complexity in L3 is 02 meaning that write two white box test cases to test the component. This is the way we optimally select the component.

V. Conclusion

The software component selection is done through various metric such as requirement metric, design metric, source code metric, validation metric and documentation metric. Metric is a quantitative measurement. After applying the metric we get number and the component which has the least value will be best. We can select this

component and rework on it. Software development can be rushed with lesser re-work by implementing the component. The cost of the software projects can also be reduced optimally and the overall product development cycle can be made faster. This could be pivotal for project development.

References

- [1] Roger Pressman, Software engineering, 7th international edition, 2010.
- [2] Software Reuse Metrics and Models WILLIAM FRANKS Virginia Tech and CAROL TERRY INCODE Corporation.
- [3] Dorothy Graham, Erik van Veenendaal, Isabel Evans and Rex Black. Foundations of software testing, US, 2006.
- [4] Omprakash Deshmukh, Mandakini Kaushik "A Overview of Software Verification & Validation and Selection Process" International Journal of Computer Trends and Technology volume4, Issue2- 2013.
- [5] Software Reuse Methods, Models and costs by Ronald J. Leach.
- [6] An Improved Model for Component Based Software Development by Asif Irshad Khan, Noor-ul-Qayyum, Usman Ali Khan Department of Computer Science, Singhania University, Jhunjhunu, Rajasthan, India Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah, Saudi Arabia.
- [7] Murat güneştaş, "A study on component based software engineering", Master's thesis in Computer Engineering, Atılım University, January 2005.
- [8] Sajid Riaz, Moving Towards Component Based Software Engineering in Train Control Applications, Final thesis, Linköpings universitet, sweden, 2012 .