ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 5, pp. 16858-16865, May, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

SPIRAL MODEL FOR COMPONENT-BASED SOFTWARE DEVELOPMENT

Shreeram Hudda*

School of Computer Science and Engineering Lovely Professional University, Phagwara, India

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0805.0232

ARTICLE INFO

Article History:

Received 06th February, 2017
Received in revised form 14th March, 2017
Accepted 23rd April, 2017
Published online 28th May, 2017

Key Words:

COTS, Software Life Cycle Model, Component-Based Development, Waterfall Model, RAD Model, Component-Based Software Engineering, Spiral Model, Component Assessment.

ABSTRACT

Industry is moving towards components based software development. Component is a ready-made code or reusable code. The advantage of component based development process is development time and cost both will be reduces. In the hardware development process, components are directly used in the project without changes but in the software development, project to project the functionality will be differ. So the components are customized rather than direct use. To develop large and complex system from scratch, it requires more number of resources such as time, money, and manpower. So Component Based Software Engineering (CBSE) takes place to develop such system. The role of CBSE is to handle entire framework activities of system development with component, and component development separately. The lifecycle activities for component-based development are different from non-component based development. Therefore, there is a need of a specific life cycle model to develop system with components or component based development. In this paper, we introduce the spiral model as component based development model, with some adaptions. We named it as "Spiral Model for Component Based Software Development."Here, we are describing framework activities for system development with components, component assessment process, and component development separately.

INTRODUCTION

Several different SDLC models are exist in software engineering. These models are considering some specific technical and non-technical goals. Examples of such models are different traditional models such as Waterfall model, V model, RAD model, Incremental model, Iterative model, and Spiral model. These models are delimited by some development constraints. We can categorize these SDLC models into two main group: i) Sequential model ii) Evolutionary model. In sequential model, developers can perform next activity only after the completion of current activity; and in case of evolutionary model, developers can perform several activities in parallel also. Component-Based Software Development is a very new and young discipline, and still required some quality research.

For an example, we are considering two banks here. These banks are: State Bank of India (SBI), and Punjab National Bank (PNB). An organization XYZ has developed a system for SBI. Now PNB contracted to same XYZ organization for developing a system with almost same functionalities. For XYZ, using the component(s) from SBI system into PNB system is more beneficial rather than developing from scratch. Since both systems have almost same functionalities. In similar manner when developers use a pre-existing component, with some adaptions if necessary, then it becomes more beneficial for them rather than develop it from baseline.

When an organization used components that are developed by a third party or vender then they are off-the-shelf component. So off the shelf components are not available in organization's library, and same product are not developed by the same organization. When an organization used a component that are extracted from own organization library then they are fully experienced component. When the component is partially developed means some functionality are developed by own organization, and some functionality are developed by other vender then it is partially experienced component. An organization is not interested in off the shelf component then development of a component is new component.

S. A. Fahmi *et al*. mentioned, "now software systems have become more complex and larger than earlier. Using the traditional lifecycle models for development of large and complex system resulted problems such as failure to meet budget, deadline, and quality requirements. The development of such system from baseline is always not more beneficial than development with reusable components", in research study [5].

Corresponding author:* **Shreeram Hudda
School of Computer Science and Engineering Lovely Professional University, Phagwara, India

In study [5], researchers highlighted that "when every time large and complex software products are developed from scratch, it would not be possible to overcome such problems that are encounter during the development of products. So overcome those problems the concept of re-usability was occurred which created the idea of component-based software engineering (CBSE).Now CBSE has become very popular and effective way to develop the large and complex software products."

In research study [17], authors have published first paper about component based software development. In [17], authors have used the spiral model as component based software development. They have also presented several reusability challenges. They have described component assessment model, and provided introduction about component development process. However, they not described component development process in detail. Now in this paper authors describe component based software development process using spiral model with some adaptions, and component development process in detail. Here, authors divide component development process into two parts. These two parts are: (i) enhancing existing functionalities or add new functionalities into an existing component, and (ii) developing new component from scratch. The authors use waterfall model to enhance the functionalities (as available under section 6), and RAD model to develop new component (as available under section 6).

The remainder of this paper is organized as follows. Section 2 summarizes background and related work. Section 3 summarizes non-component based spiral model. Section 4 discusses spiral model for component-based software development. Section 5 presents a process for component assessment. Section 6 describes component development process. Section 7 concludes this paper and states future work.

Related Background Work

A number of development processes for component-based development (CBD) have been proposed, e.g. [1], [2], [3], [4], and [5]. I. Crnkovic *et al*. proposed a research work [1], in which they presented that "a component-based approach cannot be fully utilized when the development processes, and even the development organization are not adopted according to the basic principles of CBSE."The researchers described the principle differences between component-based and non-component-based processes.

I. Crnkovic *et al*. in study [2] described "different phases of component-based system lifecycle. These phases are described in a set of framework activities for a particular process model. The process of component development and component-based system development differs in many significant ways from the classical development process of software systems. System development process with components is different from component development process. This separation has result on several other activities such as programming issues are less emphasized, while verification process requires more efforts." They also analyzed the basic characteristics of component-based approach and the generic lifecycle of component-based systems, and lifecycle of components. "Y: A New Component-Based Software Lifecycle Model [3], presented by L. F. Capretz that considered multifaceted activity that involves domain engineering, frame working, assembling, archiving, and design of software components." S. A. Fahmi *et al*. described a work [5], in which they discussed some of the popular approaches and given a comparative discussion among those approaches considering the challenges faced by component-based development.

"The W model for Component-based Software Development" [4], discussed by K. K. Lau *et al*. that addresses V & V (verification and validation) process. The W model is extended version of V model. Basically W model defines two V – one V for the process of component development and one V for the process of system development, and they combine the two processes into a single CBD process. The V for component development defines a process for identifying and defining repository components from domain requirements as well as V & V for such components. The V for system development defines a process for assembling repository components into a system according to system requirements as well as V& V for such component composition.

M. Morisio *et al*. presented a work in which they described an investigation of COTS-based software development for a particular NASA environment [9]. They analyzed fifteen COTS projects, these projects represent different software domains. These projects use more than thirty COTS products. They defined a new process and a set of activities for COTS-based software development.

I. Crnkovic *et al*. presented a study in which they highlighted several important issues related to the development and maintenance of reusable components [15]. They showed a successful example of the component-based system development by the ABB Advant Open Control Systems (OCS).

I. Crnkovic proposed a work in which he described several challenges that are facing by CBSE [8]. Some challenges out of them are: "trusted components, component certification, composition predictability, tool support, requirements management and component selection, development models, component configurations, dependable systems and CBSE, and long-term management of component-based systems." Researcher also described several disadvantages and risks that are present in CBD those can affect success of CBD. Several disadvantages are: "time and effort required for development of components, unclear and ambiguous requirements, conflict between usability and reusability, component maintenance costs, and reliability and sensitivity to changes."

Spiral Model

When the requirements are not clear to develop the product functionality from base to advance level spiral model is used. Here, we are summarizes Spiral Model framework activities using a diagram. These activities authors have presented in detail in study [17]. Non-component based Spiral model contains following six activities in the framework. Figure 1 describe the framework activities of spiral model. These are [17]:

1) Customer communication, 2) Planning, 3) Risk analysis,
4) Engineering, 5) Construction and release, and 6) Customer evaluation.

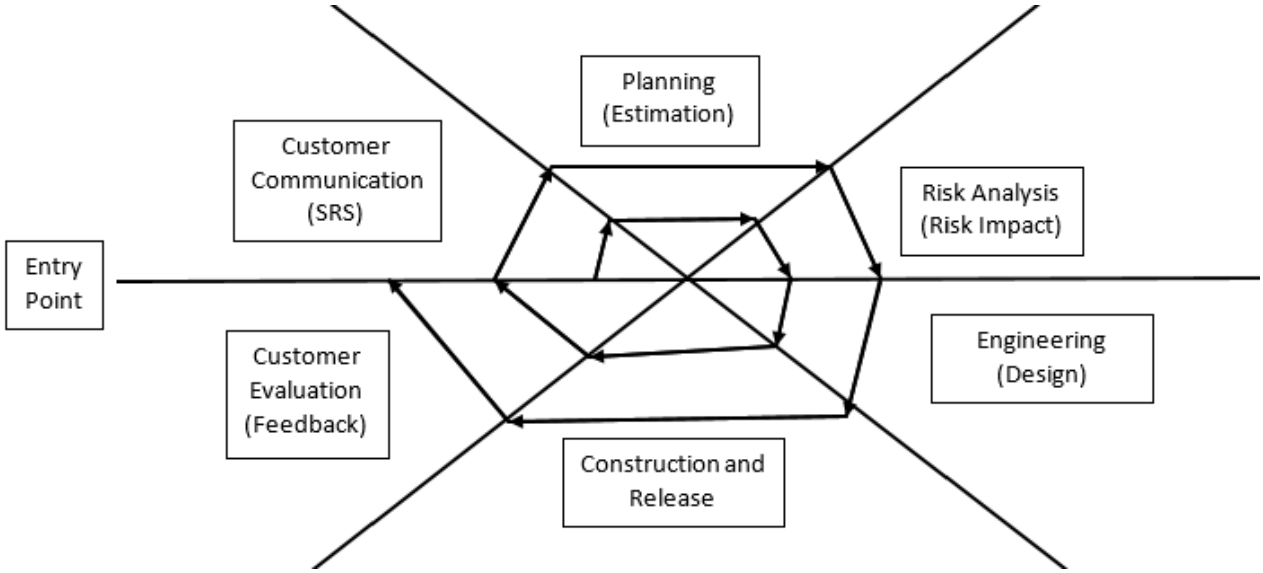

Figure 1 Spiral Model [17]

Spiral Model for Component-Based Software Development

The primary idea of the component-based development approach is to reuse the pre-existing components instead of developing new components from scratch. So the availability of existing components must be considered in requirement analysis and design phase of system development. This consideration has several results for the component-based system development lifecycle: i) the process of system development with components is separated from process of components development, ii) a new process for component assessment, and iii) several adaptions are required in the lifecycle activities of the development processes. Hence, component assessment process takes place before component development. It means developer's team search a candidate component into own organization library rather than developing a new component. When candidate component is available then use it directly or with some adaptions, if necessary, otherwise development starts. Usually, components are built for reuse; in sometime not yet existing systems, and in some time possibly planned systems.

The system development with components is based on the identification of reusable component and relation between them [13], [14]. More effort is required in finding, evaluating, and testing the components; and less effort is required in implementation of components [9]. Figure 2 shows a spiral model, with some adaptions, adopted to component-based approach [17].

Customer Communication

In this phase, requirement engineers are communicating with customers to collect the requirements. In component-based approach requirement gathering is quite more different from normal traditional approach. In component-based approach, one important activity is to analyze the availability of existing components so that developers can fulfill more number of these requirements by available components. This means that the requirement engineers must aware about availability of existing components. It need not be necessary that appropriate components can always be found.

So developers can do two things, they caneither start implement of new components or modify the requirements in such a way so that they can use existing components. The SRS document contains three things. These are: i) Goal of the system ii) Functional requirements iii) and Non-functional requirements. After preparing the software requirement specification (SRS) document, the developer can understand how many functionalities are developed in final product.

Planning

This phase is similar to the phase of non-component-based spiral model. In this phase, various attributes of the software product are estimated by using different empirical models. The various software attributes are size, cost, effort, and schedule. When the developers are using existing components then the estimation of cost, size, effort, and schedule will minimal.

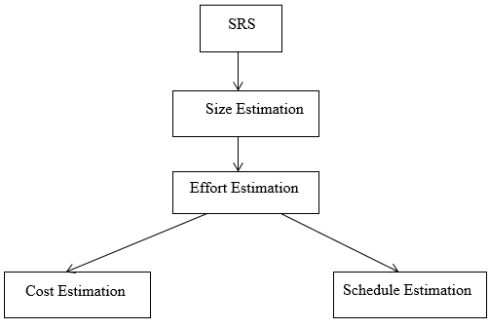

Figure 2 Attribute Estimation Sequence

Figure 2 shows the attribute estimation sequence. After prepare the SRS document the developer can understand how many functionality are going to be developed. So based on this knowledge the size of the software is estimated. After calculate the size of the software by using different empirical model, developer can estimate the effort required to develop the software. After estimate effort for the software, developer can estimate the cost and schedule.

Risk Analysis

This activity is also similar to the activity of non-component-based spiral framework. Different risk factorsare involved in

the software project. These risk factors are analyzed, and the risk factor impact is estimated, and mitigate it.

Engineering, Construction And Release

Before designing the software design description (SDD) document, components are introduced in the development process since designer must be aware about the availability of components. After preparing the SDD document, developers are able to understand how many functionalities are required by the system. First of all, verify the library for the probable components. To verify the library for the probable components, developer's team is analyzing how many requirements can be fulfilled by available components. These requirements can be a part of system being developed, or of a system plan to be developed. When a candidate component (i.e. a component that fulfill the required functionalities) is available in the library then select it, extract the candidate component from library, and use it. More than one candidate components may available in organization library. Whenever more than one candidate components are available then compare all the candidate components, and ranked them. The ranking of components should be maintained throughout the system development so the alternatives for a function can quickly be found. These candidate components may be: i) Commercial off the shelf (COTS), ii) fully, and iii) partially components.

Hence, when the required component is available in library then use it directly into the system, or do some adaption, if necessary, whenever required. When the component of required functionality is not available in organization library then develop the functionality from baseline, and make the component available for reuse in future. Before releasing the product into customer's location, keep the new components in the library.

Therefore, it may become existing components for future systems. Hence, component assessment process takes place before component development. The component assessment and component development process described in two separate sections below.

When building a component-based system, a system may build by directly connecting components, if they available. This connection is done by "glue code". The environment, and usage of components are not clear so the testing of components should be performed very carefully. Several different techniques of testing, different test suites should be applied on the component for verification. Test document together with component should be delivered to system developer. The components could have been developed in another system with some other purpose. Since system developers not having any control over the components functions. Hence, the unit testing of components is not sufficient since the behavior of the component can be different in other environments. The components should be created in such a manner so that they can easily integrated with other components of system. Integration with other components also possible to provide several specific functionalities. So integration of component with other components must be tested separately since an assembly of correct components may be incorrect. The testing is performed at many different times through unit testing, integration testing with subsystem, and system testing.

Whenever in the developed system if any portion is changed in one component that will be causes different errors in the entire system. So to cover those errors there is a need of retest the system, this is regression testing. When the software is developed based on customer priority list then some of components are added, and some pf components are deleted.

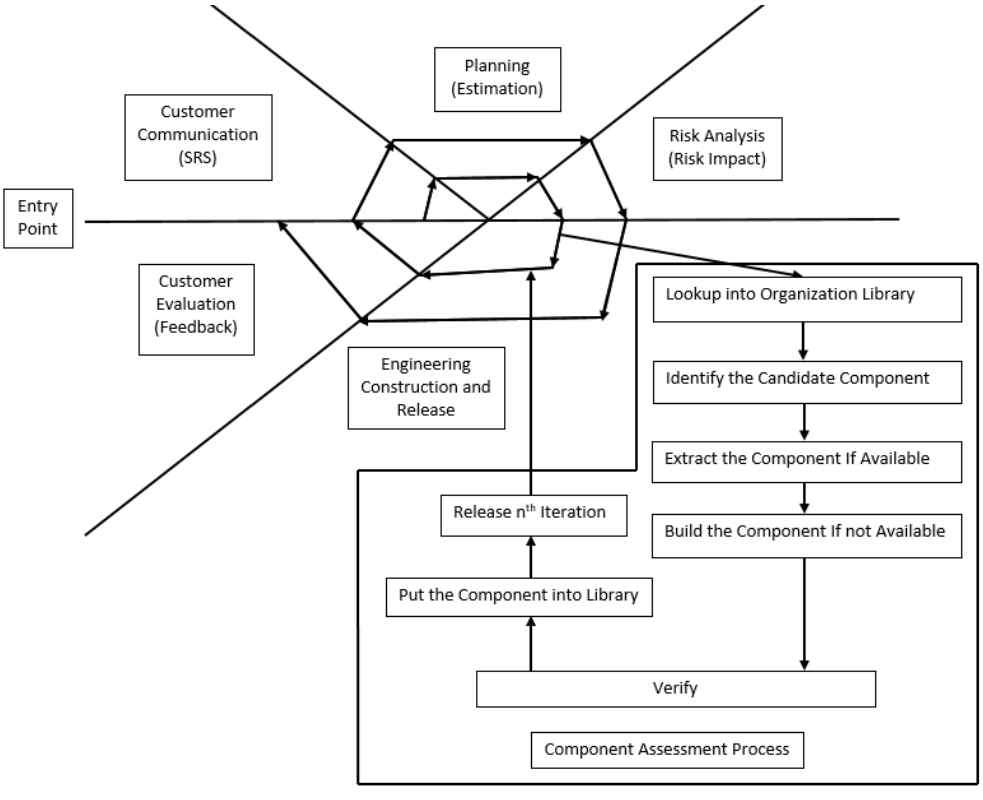

Figure 3 Spiral Model for Component-Based Software Development [17]

During this process functionality will be changed. So to cover the functional errors different black box test plan are implemented, this is smoke testing. Therefore, regression testing, and smoke testing also takes place in system development with component. After the testing, first release of product is deployed into the customer location

Customer Evaluation

This phase is same as non-component-based spiral model. In this framework continues feedback is accepted from the customers to improve the product functionality.

Figure 3 shows Spiral Model for Component-Based Development. In this model, release by release the product functionality are increases according to customer requirements. Therefore, customer's satisfaction level is high. In each level development process, developer skill level are elaborated. Therefore, this model is also named as "Win-Win Spiral Model for Component-Based Software Development."

Component Assessment

The component-based system development decreases the effort required in design, and implementation of system development, it needs additional effort in several other activities. It requires more efforts in searching of existing components that satisfies more number of system's functionalities, instead of implementing the required functionalities by new components. The developers must verify two things in the selected components: i) selected components must provide required functionalities, and ii) they can be successfully integrated with other components of system also. Selected components can't be the "best component", but it can be fit together effectively. In non-component-based approach process starts by requirement gathering and requirement analysis (i.e. customer communication), and end with customer's feedback but in component-based approach selection of appropriate components, and integration of them into the system will also be considered. Hence, here two problems appear: (i) It need not be necessary that the components of required functionality are always available in organization's library and (ii) It may possible that selected component fits partially in our system's design. So there is a requirement of a process for finding components and a need of component testing before it can be integrated into the system. A component assessment process for may include the following activities [17]:

Lookup into Organization Library

Developers must look up into organization's library to identify availability of required components. There may available more than one components in organization library. So developers should keep more concentration in searching of a component. To search the component into library, the developers match the required functionalities with the component's functionalities.

Identify the Candidate Component

Identify the component from organization library that can fulfill the required functionalities. These functionalities can be part of the system being developed, or of systems plan to be developed. There can be more than one candidate component. So developers should identify the candidate component, ranked them, and use best candidate component.

Extract the Component If Available

There may available more than one candidate component in organization library. When more than one candidate components are available in the organization's library then compared them, ranked them, and select the "best component". A component is a "best component" if it provides best function, or more number of functionalities compared to other components. The ranking of components should maintain throughout the system development such that developers can select alternatives for a function quickly. When a component of required functionality is available then select it, and use it in system development. The developers must verify two things in the selected components: i) selected components must provide required functionalities, and ii) they can be successfully integrated with other components of system also. Selected components can't be the "best component", but it can be fit together effectively. These components defined by three different categories. These are: (i) Commercial-off the self, (ii) fully, and (iii) partially.

Build the Component If Not Available

Whenever the required components are available in library then use them directly into the system, or do some adaption, if necessary, whenever required. When the component of required functionality is not available in library then develop it from baseline, and put it into the own organization library for reuse in future, this component is called as new component. The New component should be developed in such a way so that developers may reuse it in future systems. Before releasing the product into customer's location, keep the new components in the library. They may become existing components for future systems.

Verify

When building a component-based system, a system may build by directly connecting components, if they available. This connection is done by "glue code". The environment, and usage of components are not clear so the testing of components should be performed very carefully. Several different techniques of testing, different test suites should be applied on the component for verification. Test document together with component should be delivered to system developer. Components could have been developed in another system with some other purpose since system developers not having any control over the components functions. Hence, the unit testing of components is not sufficient since their behavior can be different in other environments. Components should be created in such a manner so that they can easily integrated with other components of system. Integration with other components also possible to provide several specific functionalities. So integration of component with other components must be tested separately since an assembly of correct components may be incorrect. The testing is performed at many different times through unit testing, integration testing with subsystem, and system testing. Hence, verification process divided into two levels. At first level developers testing functional and extra-functional properties of the component separately. The second level includes the testing of the component with other components.

Put the Component in Library

Developers make sure that they have placed new components in the library. These components may become existing components for future systems. The library contains not only components but also some additional information (i.e. metadata) related to components such as tests results, passed tests, failed tests, and test procedures that can be useful in future.

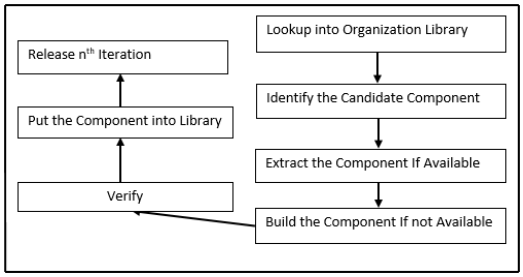

Figure 4 Component Assessment Process

Figure 4 describe the component assessment process. All these activities of component assessment process are need not be necessary to performed in the order as explained above. Some activities are more important, and some are less important. For example, if an organization uses only internally build components then identify the candidate component activity will not important.

Component Development Process

The component development process is similar to system development process. For development of a new component, developers can use any arbitrary model either software development lifecycle model or reuse other components as for system development. Components are developed for reuse in many different systems, many of them yet to be designed or yet to be planned. So reusability of component implies adaptability. Since developers can add some new functionality to components, edit existing functionality, and remove any unnecessary functionality from the components according to system requirements. To develop reusable components developers faces several challenges. These are: (i) More difficulty in managing requirements and (ii) More efforts are needed to develop reusable components.

We divide the development process for a component into two parts. These parts are: (i) Add new functionalities or features to an existing component; or enhance the existing functionalities of a component (ii) Develop new component from the scratch.

Enhance the Existing Functionalities of a Component

First we discuss about the first part – enhance the existing functionalities of a component. We are suggesting to the developers that they can use waterfall model to enhance the existing functionalities of a component, if component does not provide more functionalities to developers. Since waterfall model is best suited when more requirements are available. In the case of enhancing the existing functionalities or add some new functionalities to an existing component, more requirements are available. Since some requirements are already available in the component, and some are to be add. Therefore, waterfall model is best suited in this case. The activities of waterfall model are: 1) Analysis, 2) Design, 3) Coding, 4) Testing, and 5) Support. Here, we discussing these activities in detail.

Analysis

In this activity, software requirements are gathered based on the problem statement. The problem statement translated into the symbolic representation. Based on the goal of the system different data objects are gathered by using the different elicitation techniques. All the data objects are documented into data dictionary. The data objects are divided into functional and non-functional requirements. The output of analysis stage is documented called as Software Requirement Specification (SRS). SRS document contains: - (i) Goal of the system, (ii) Functional requirements, and (iii) Non-functional requirements.

Design

In this stage by using different design principles the generalized problem statements are translated into procedural descriptions (i.e. algorithms). The symbolic representation can in the form of ER-diagrams, Data Flow diagrams, and State Transition diagram. After symbolic representation, different technical specifications are finalized to implement system. Based on the specifications, component functionalities are described in the algorithm. The output of this stage is documented is called as Software Design Description Document (SDD). SDD contains algorithms, flowcharts, pseudocodes, and data structures. So an overall system and software architecture is established.

Coding

In this stage, the procedural description (i.e. algorithm) is translated into machine-readable format by using the appropriate programming languages.

Testing

Different test classes are implemented on the developed program to cover the structural and functional errors. The objective of testing is to prove the system at what extent it becomes failure. Unit testing, integration testing, and system testing should be performed. After testing the system will be deployed into the customer location.

Support

When the system is in customer sight then different kinds of support is required to maintain the system from uncovered errors, platform changes, functional requirement changes, and frequent changes.

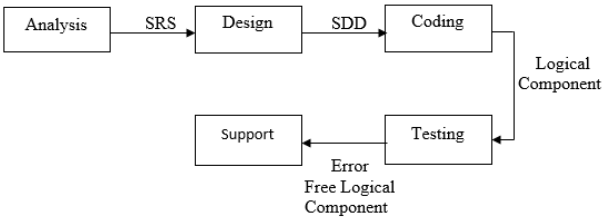

Figure 5 Waterfall Modeltoenhance the Existing Functionalities of a Component

Develop New Component from the Scratch

Now we discuss about the second part-develop new component from the scratch. In research study [17], authors have discussed

issues related to incorporating reusability in software. In [17], authors stated that "The process of developing reusable components separately from their target system reduces the interconnection among parts of the system – a quality called as low coupling. This low coupling makes the resulting application easier to understand, modify, and test. Since developers require low coupling, and high cohesion." The developers can achieve these objectives by using modularity design i.e. RAD model. When many components are developed from scratch then developed them independently or separately, and later integrate them into the system. To integrate them into the system unit testing, integration testing, and system testing takes place. The connection of components with system done by "glue code". The RAD model is best suitable when developers have tight time schedule. The RAD model uses modularity design to develop time critical applications. In modularity design, developers decompose the application into modules, and later integrate it into the system. In RAD model two factors are considered. These are: 1) Cohesion, and 2) Coupling. Cohesion is a quantitative measure of what extent the module i.e. component is independent from other modules. The cohesion represents functional strength of the module. Coupling is a quantitative measure of what extent the module is dependent on other module. The effective modularity design maintains the high cohesion and low coupling. The framework of RAD model consists these activities: 1) Business Modelling, 2) Data Modelling, 3) Process Modelling, 4) Application Generation, and 5) Test and Turnover.

Business Modelling

In business modelling activity the existing application under goes analysis to finalize the business objectives. The analysis phase and design phase both together known as system engineering. Software is a subset of the system. So to finalize the objectives there is a need of analyzing the existing system. The output of system engineering is business objective or business function.

Data Modelling

Based on the business objective various data objects are finalized in the data modelling state.

Process Modelling

In process modelling state information flow is created either by adding, deleting or modifying the objects.

Application Generation

In application generation the information flow is translated into machine-readable format. This phase is similar to coding phase in waterfall model.

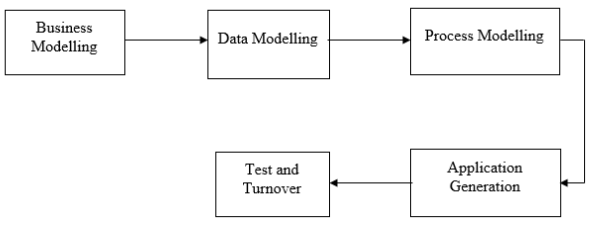

Figure 6 RAD Model to develop the New Component from the Scratch

Testing

In testing stage various test plans are implemented on the module. At the end of testing error free module is available called as component.

CONCLUSION

In this paper, we have presented the development process of system with components, component assessment process, and component development process. Here, we presented a component based development model named as "Spiral Model for Component-Based Software Development". We made some adaptions in spiral model, and presented it as component based software development model. The component assessment process takes place before component development since identify the availability of candidate component before a new component development starts. The component development process is divided into two parts. In one part we use waterfall model, and in second part we use RAD model for component development. Developers can integrate component into the system directly or they can change some functionality. The development of a component from scratch requires more cohesion, and less coupling. Hence, RAD model is useful for component development from scratch.

References

1. I. Crnkovic, S. Larsson, and M. Chaudron. "Component based development process and component lifecycle."27th IEEE Int. Conf. on Information Technology Interfaces (ITI), pp. 591-596, June 2005.
2. I. Crnkovic, S. Larsson, and M. Chaudron. "Component based development process and component lifecycle." In Proc. of the Int. Conf. on Software Engineering Advances (ICSEA06), pp. 40-44, October 2006
3. L. F. Capretz. "Y: A New Component Based Lifecycle Model." Journal of Computer Science, pp. 76-82, 2005.
4. K. K. Lau, F. M. Taweel, and C. M. Tran. "The W model for Component-based Software Development."37thEUROMICRO Conf. on Software Engineering and Advanced Applications (SEAA), pp. 47-50, September 2011.
5. S. A. Fahmi, and H. J. Choi. "Life Cycles for Components-Based Software Development."8thInt. Conf. on Computer and Information Technology Workshops (CIT Workshops), pp. 637-642, July 2008.
6. B. W. Boehm. "A Spiral Model of Software Development and Enhancement," IEEE Computer Society vol. 21, pp. 61-72, May 1988.
7. A. W. Brown. "Large Scale, Component-Based Development." Prentice Hall PTR Upper Saddle River, USA, ISBN- 013088720X, December 2000
8. I. Crnkovic. "Component-based Software Engineering-New Challenges in Software Development."In Proc. of the 25thInt. Conf. on Information Technology Interfaces (ITI), pp. 9-18, June 2003.
9. M. Morisio, C. B. Seaman, A. T. Parra, V. R. Basili, S. E. Kraft, and S. E. Kondon. "Investigating and Improving a COTS-Based Software Development Process." In Proc. of the 2000 Int. Conf. on Software Engineering, pp. 32-41, 2000.
10. T. C. Lethbridge, and R. Laganiere. "Basing Software Development on Reusable Technology." Object-

Oriented Software Engineering-Practical Software Development Using UML and Java, 11th reprint 2011, Tata McGraw-Hill Education, pp. 61-100, 2004.
11. B. Hughes, M. Cotterell, and R. Mall. Software Project Management, 5th edition, 6th reprint, Tata McGraw-Hill Education, New Delhi India, pp. 68-96, 2013.
12. R. S. Pressman. Software Engineering: a Practitioner's Approach, 6th Edition, New Delhi, India, McGraw Hill Education (India), 2010, pp. 103-126.
13. L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, Addison-Wesley, 1998.
14. D. Garlan, R. Allen, and J. Ockerbloom. "Architectural Mismatch: Why Reuse is so hard." IEEE Software, pp. 17-26, vol. 12, issue 6, November 1995.
15. I. Crnkovic, and M. Larsson. "A Case Study: Demands on Component-based Development." 22nd Int. Conf. on Software Engineering, pp. 23-31, ACM Press, 2000.
16. L. Sommerville. "Software Engineering." 7th Edition, Addison Wesley, June 2004.
17. S. R. Hudda, R. Mahajan. "Spiral Model for Component-Based Software Development." 3rd Int. Conf. on Comp. Sci. (ICCS), pp. 817-836, November 2016.

How to cite this article:

Shreeram Hudda.2017, Spiral Model for Component-Based Software Development. *Int J Recent Sci Res.* 8(5), pp. 16858-16865. DOI: http://dx.doi.org/10.24327/ijrsr.2017.0805.0232

****