2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Umbrella: A New Component-Based Software Development Model

Anurag Dixit and P.C. Saxena

School of Computer & System Sciences
Jawaharlal Nehru University New Delhi (INDIA)

Abstract. In the present Information Technology era, software research community has lot of pressure to reduce development cost as well as development time. Therefore large number software is being developed by using third party component. The traditional software life cycle model provides a systematic method to separate the different phases with efficient boundaries. A number of software life cycle models existing today primarily focused on building of traditional software systems. The exact nature of the relationships between traditional software life cycle models and life cycle models that incorporate the different phase imposed by Component Based Software Engineering are needed. Hence, Component Based Software Engineering raises questions about whether existing software engineering life cycle approaches are appropriate in this emerging field. Therefore a life cycle model which incorporate testing or verification as a continuous process is being proposed. In this paper we examine the various software life cycle models. We argue that testing phase in life cycle model not only overlaps but also cut across the various developments phase stages. We are proposing a software life cycle model for component based software development from the conventional models known as a Umbrella life cycle model.

Keywords: Component Based Software, Life Cycle, Model

1. Introduction

As the other engineering activities software process also goes through the series of development phases. The series of phases (progresses) through which the software undergoes from requirement exploration to maintenance is termed as a software life cycle. During software development the software goes through a series of phases i.e., requirement, specification design, selection, retrieval, testing and maintenance etc... In the early days of software construction process followed sequential flow know as waterfall model but in late 70s a drastic change come in software industry and there are number of variations introduced. A software life cycle model can be chosen based on the nature of application and environment. The sequential models define a sequence of activities in which one activity follows after a completion of the previous one. Evolutionary models allow performance of several activities in parallel without requirements on a stringent completion of one activity to be unable to start with another one.

2. Motivation

A software life cycle usually defined as a vague concept. The life cycle phase of each software is different. Some process will spend large time and cost as compared to others phases. Software development process is an expensive activity. The development process or life cycles usually start with requirement gathering from the users of system. All the traditional software life cycle models are sequential type it means each phase must be completed before next phase will start. Well known example of sequential models is waterfall model, or V model, and of evolutionary models, iterative and incremental development, or spiral model. Component Based Software Engineering is integration-centric approach,

emphasizing on the selection, acquisition and integration of components from third party or in-house sources.
There are much difference between component based software life cycle as compared to traditional software development life cycle. Majority of component based application are
developed by third party component its means component based software life cycle start with retrieval not from construction. In the component based systems, component just binary object provided by third party without source code. Therefore a need arise to verified specification given by third party.

3. Pre Requisites or Terminology

The success of any software depends upon good design. Like any other product process software systems also need some steps to assured quality of software. Generally these phase are used in every software life cycle model

3.1Requirement

Requirement analysis is the first phase in most of software life cycle models. It is process after a feasibility study has been performed. In the requirement analysis phase we decide user desire in terms of functionality, performance, behavior and interfacing. We make documented for exact requirement of the systems. In this phase user of the system or customer and developer are closely interact to decide all these things.

3.2 Specification

Once the developer understand user requirements in next phase specification document is drawn. In the specification phases as compared to informal requirement phase a detailed specification document is construct which describe functionality of the software. This phase is sometimes split into two sub phases: architectural or detail design.

3.3Design

In the design phase we translate requirements into a representation of the software.

3.4 Implementation

In the implementation phase design is translated into a machine readable form.

3.5Verification

Once code are prepare, program verification begins

3.6 Maintenance

Once a software system passes all the verification and validation, it is delivered to the user and enters in the maintenance cycle. Any modification made to the system after initial delivery is part to this phase.

4. Related work:

4.1 Waterfall model

The waterfall model was first introduced by Royce in 1970. Its linear model where various phase are interconnected so that output of one phase become subsequently input for next phase. The Waterfall model has been long used by software engineers and has become the most prevalent software life cycle model. This model initially attempts to identify phases within software development as a linear series of actions, each of which must be completed before the next is commenced.

Challenges

1. Major drawback of Waterfall Model is inflexible division of phase. Today requirement are changing very fast all phase are overlap.
2. In the waterfall model output are verified in last stage.

Therefore waterfall model is good for such situation where requirement are well defined.

4.2 Incremental Model

To overcome drawback of waterfall model a cycle model know as incremental model was developed by .in the incremental model we combine the property of sequential model and iterative charterstic of prototype life cycle model. In the incremental model partial implementation of systems is construct after refine this implementation require functionality achieve.

Challenges

It is time consuming process. User of the system involved whole process.

4.3 Evolutionary model

Evolutionary prototypes provide incremental software development, so that software systems may be gradually developed and tested, allowing major errors to be exposed and corrected early, which means that they are often cheaper to fix,but without effective management to control iterations, this process can degenerate into uncontrollable hacking.

4.4 Prototyping model

Generally customer of the software system initially define broader requirement of the systems. In this case a quick prototype can be construct. Prototyping provides constructive feedback to designers and potential users so that the system requirements can be clarified and refined early during software development. The prototype is evaluated by the customer and refine his requirement.

4.5 Spiral Model

The spiral model was first developed by Boehm, is an evolutionary software life cycle model. In the spiral life cycle model different phase are represented as a spiral rather than sequence of activates. The Spiral model makes software development more flexible and has been proposed mainly to speed up software development through prototyping. In the spiral model, software is developed in a series of incremental release. The spiral model is divided into a number of regions, typically these are between three to six.

4.6 V Model

In the V model the process starts in a usual way by requirements engineering and requirements specification, followed by system specification. 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 components and integrate them in the system.

4.7 Y Model

A strictly top-down or bottom-up strategy to software production is not quite appropriate. The Y model preaches a top-down or bottom-up fashion for software creation, taking into consideration the knowledge that a software engineer has about the application domain. This knowledge naturally determines the prevailing strategy to software development.

4.8 A Model

The A shape Model developed for supporting parallelism and evolutionary design of the application and of the test plans too.

Challenges

This model does not provide a notation or a set of patterns to increase reusability.

5. Proposed Model

The life cycle of components based software consists of three stages: (i) the Design phase, when components are selected by repository or designed, defined and constructed (ii) the Integration phase, when component are integrate with others component (iii) the *run-time* phase, when component binaries are instantiated and executed in the running system.

A software component life cycle model should define (i) what sequence components composition are follows (ii) Why need of these sequences (iii) how to compose components.

Over the past three decades, several component based software development methodologies have appeared. Such methodologies address some or all phases of the software life cycle ranging from requirements to maintenance. These methodologies have often been developed in response to new ideas about how to cope with the inherent complexity of software systems. Due to the increasing popularity of software reuse , in the last twenty years, research on component based software methodologies has become a growing field of interest.

Software Reuse Approach

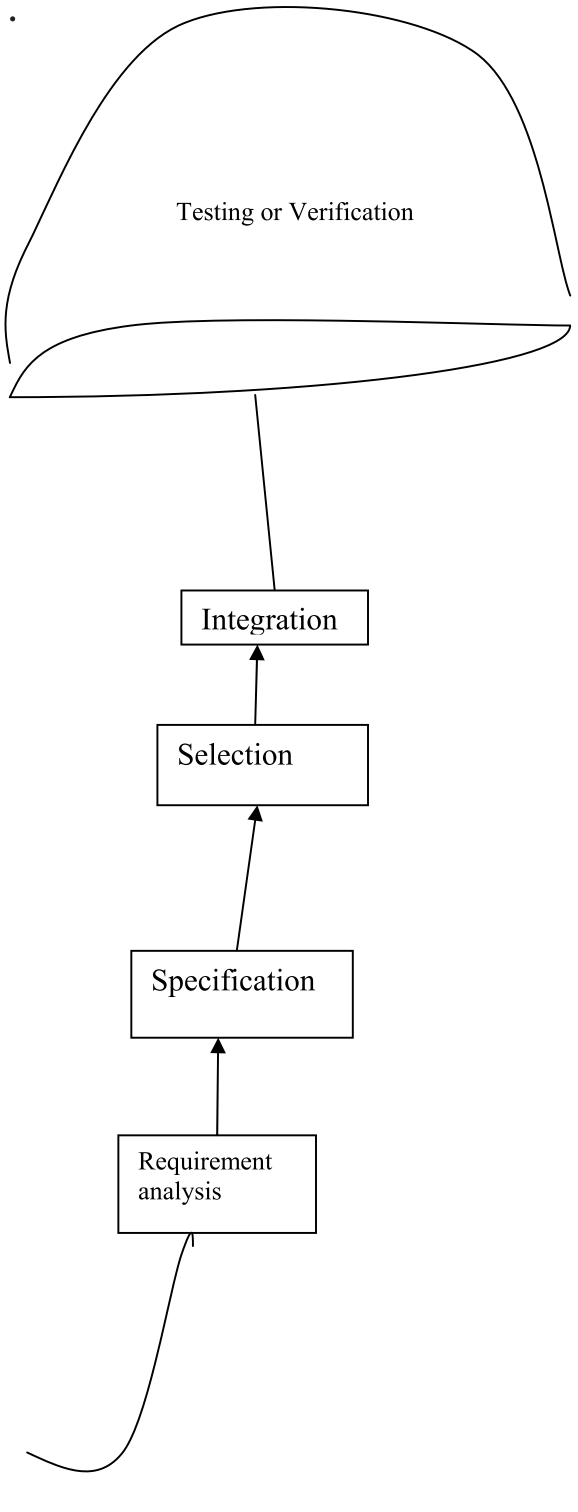
Typical steps:

1. Requirements specification
2. Component analysis: finding the right (closest match) component for the specification
3. Requirements modification (to suit components found)
4. System design with reuse
5. Development and integration
6. System validation

Step Involve In Component Based Software Life Cycle:

1. Requirement Gathering (User->Developer)
2. Requirement Analysis(Developer)
3. Design Component Specification
4. Component Selection (Iterative)

5. System Design / Integrations
6. Modification
•



1. Requirement Phase Testing
It is essential that the requirement be carefully checked by both user and integrator to be certain that it reflects their current needs. There is always a possibility that forces to make changes from both sides.
2. Specification Testing:
A major source of faults in delivered e software is faculty specification design. Any mistake occurs in specification design can be detected after software gone into operational mode. Before the specification phase can be deemed to finish its duty of Quality Assurance peoples to verify all the document with regards contradictions and ambiguities. In the specification testing phase every statement in the specification document trace with the statement made by the client team. Another ways of checking the specification document is by means of a review. The aim of the review process is to determine whether the specifications are correct.
3. Selection Testing
A number of components are required to have a reasonable chance of finding matching components for a given task. Finding that component among all the others has been identified a major problem in component-based software engineering. To support the development of high quality component-based systems, component selection processes need to address the problem of Functional and Non - Functional requirement evaluation of component-based software systems. selection testing is the main quality control activity that is carried out in the early design phase of component based design , it include component specification verification with user requirement.
4. Intergration Testing
Integration testing is verification amounts to assembling the software from the set of component that were developed and tested separated.

Conclusions

All these different software life cycle models have their own advantages and disadvantages. In this paper, we have discussed a number of activity areas that form a life cycle framework for component-based software development. In the component based software system, component selection and integration play important role. These two are interrelated concepts. In this context we are proposing a life cycle model for component based software engineering called Umbrella Life Cycle Development Model (ULCDM). This position paper has provided a component based software engineering life cycle perspective on testing or verification concerns. We have highlighted the various risks that cut across a component-based development cycle.

References

[1] Boehm, B.W., 1988. A spiral model of software development and enhancement. IEEE Computer, 21: 61-72.

[2] Luiz Fernando Capretz , Y: A New Component-Based Software Life Cycle Model , Journal of Computer Science 1 (1): 76-82, 2005

[3] Csaba Szabó, Ladislav Samuelis , The A-Shaped Model of Software Life Cycle , 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics January 25-26, 2007 pp129-135

[4] D'Souza D and A.C. Wills, Objects, Components, and Frameworks with UML: The Catalysis Approach, Addison Wesley Longman, Reading, Mass., 1998.

[5] Szyperski C, Component Software: Beyond Object-Oriented Programming, Addison Wesley Longman, Reading, Mass., 1998.

[6] G.T. Heineman and W.T. Councill, Component-Based Software Engineering, Addison-Wesley, Boston, 2001.