A New - Knot Model for Component Based Software Development

Rajender Singh Chhillar[1], Parveen Kajla[2]

[1] Department of Computer Science & Applications, Maharshi Dayanand University,
Rohtak-124001, Haryana, India

[2] Department of Computer Science & Applications, Maharshi Dayanand University,
Rohtak-124001, Haryana, India

Abstract

Component Based Software Engineering (CBSE) is a process that emphasizes the design and construction of computer based systems using reusable software components. The component is a unit, which is almost independent and may be replaced or changed without effecting the environment. This property motivates the programmer to design and develop software using component based software development.

Traditional methods for software development approach to the functionality of the system and mainly follow the sequential models like waterfall, which are mostly overridden by the Iterative and Evolutionary models like increment, prototyping, Bohem's Spiral Model.

To enhance the reusability feature of the software development new process models are required and in the light of that, in this study a new Knot Model is proposed for the Component Based Software Development (CBSD) which lays emphasis on reusability, estimation, risk analysis, feedback in every phase to improve the quality and to reduce the cost. This model effectively works with large and complex systems within short period of time.

Keywords: *Knot, Component, CBSD, Process model, Risk analysis, Feedback*

1. Introduction

A Software Life Cycle Model is an expressive and diagrammatic representation of the software process. Software life cycle model describes the phases of the software cycle. It also depicts the order in which these activities are to be commenced. A life cycle model maps the different activities performed on a software product from its inception to retirement. To develop a software product in a systematic and disciplined manner, the development team must identify a suitable life cycle model for the specific project and then follow it. The basic activities are included in all life cycle models, though the activities may be carried out in different orders in different life cycle models. The general basic model is shown below :

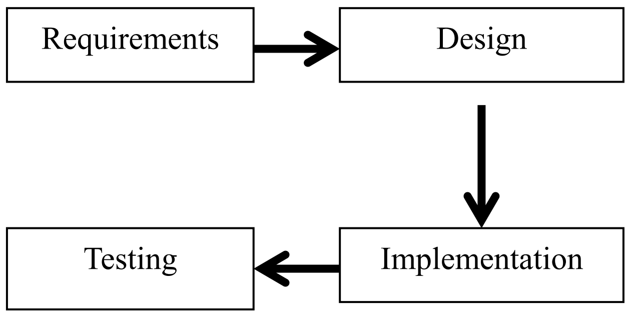

Fig. 1 Basic Model of Software Development.

2. Component Based Software Life Cycle Process Model

Various process models have been designed by a number of researchers so far for component based software development. Most common among them are studied and described briefly.

2.1 Rapid Application Development Model

The RAD (Rapid Application Development) Model achieves rapid development using a component based construction approach. But this model has various drawbacks such as efficient RAD teams, proper modularization is required, inappropriate when technical risks are high.[4]

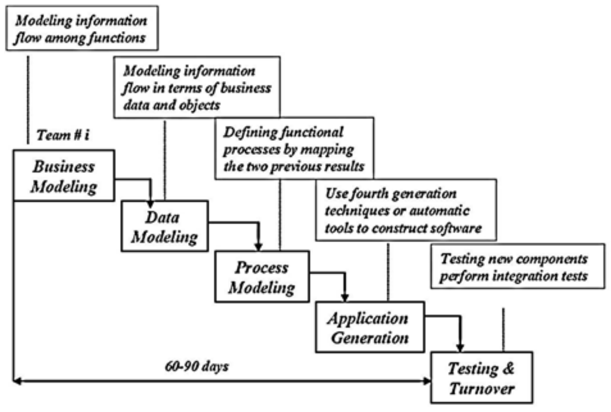
Modeling information flow among functions
Modeling information flow in terms of business data and objects
Defining functional processes by mapping the two previous results
Use fourth generation techniques or automatic tools to construct software
Testing new components perform integration tests
Team # i
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing & Turnover
60-90 days

Fig. 2 RAD Model of Software Development.

This model is proposed when requirements and solutions can be modularized as independent system or software components, each of which can be developed by different teams. Then after it smaller components are developed, they are integrated to produce a large software system.[5]

2.2 The X Model

In this X Model, the processes are started by requirement engineering and requirement specification. The main characteristic of this software life cycle model is reusability in which software is developed by building reusable components for software development and software development from reusable and testable components. In software development, it uses two main approaches, develop software component for reuse and software development with or without modification in reusable component.[2]

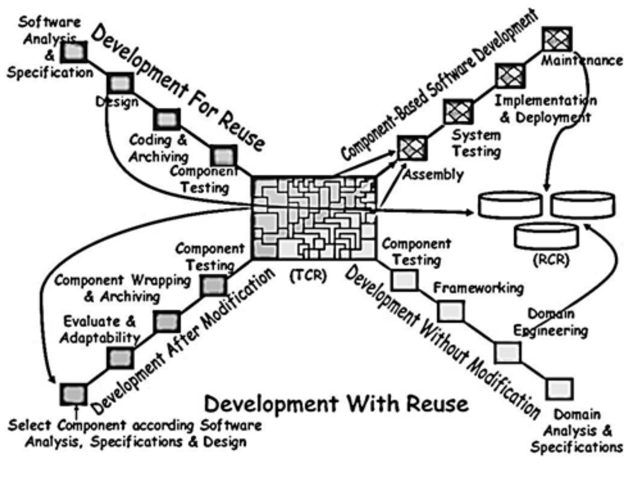
Software Analysis & Specification
Development For Reuse
Design
Coding & Archiving
Component Testing
Component-Based Software Development
Maintenance
Implementation & Deployment
System Testing
Assembly
(TCR)
(RCR)
Component Testing
Component Wrapping & Archiving
Evaluate & Adaptability
Development After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

Fig. 3 X Model of Software Development.[2]

The X model appears to be the best for large software development and has main focus on reusability but it does not depict risk analysis, feedback in various phases which is one of the reasons for the very success of Boehm's Spiral Model. Moreover the X-shape represents overlapping and increases complexity.

2.3 The Y Model

The Y Software Life Cycle Model (Luiz, 2005) describes software reusability during CBSD. The Y Shape of the model considers iteration and overlapping. Although the main phases may overlap each other and iteration is allowed, the planned phases are: domain engineering, frame working, assembly, archiving, system analysis, design, implementation, testing, deployment and maintenance.[3]

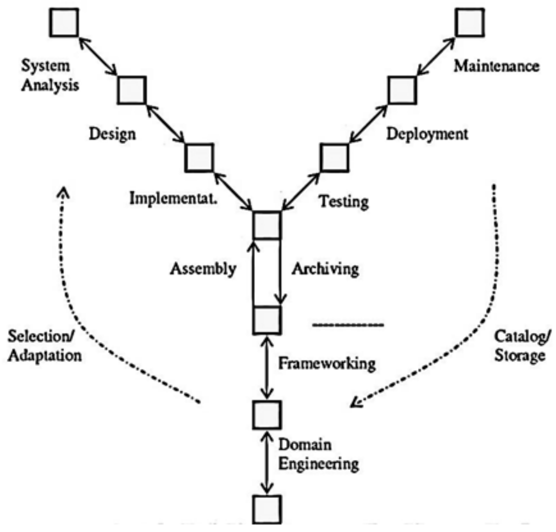

Fig. 4 Y Model of Software Development.[3]

The reusability within this life cycle is smoother and more effective than within the traditional models because it integrates at its core, the concern for reuse and the mechanisms to achieve it. There is an upper layer that shows the component templates and a lower layer that consists of run-time objects that depicts the behavior of a particular software system. The development of a component should therefore be with generality and reuse in mind placing perhaps less emphasis on satisfying the specific needs of an application that is being developed.

3. - Knot Model for Component Based Software Development

In this study, a new Component based model is described in which reusability in the form of component is applied.

The utmost emphasis is given on reusability, modularity, risk analysis and feedback in each phase, which results in simple, error free and a profound new system with proper feedback in each phase of software development. For a new CBSD, it uses all the three: the newer one, the existing component and the modified component to build up a new development. It consists of four phases.

1. Reusable Component Pool
2. Development of New Component
3. Modification of Existing Component
4. Development of New Component Based Software Development.

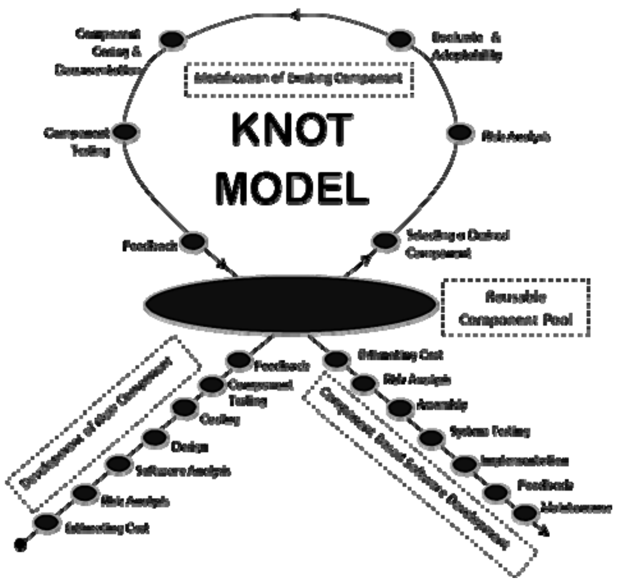

Fig. 5 A New Knot Model of Software Development.

The major advantages of this software life cycle is to handle the complex systems using modularity, reusability, life time process, evaluation & testing in each phase to reduce the risk.
And moreover the developed CBSD itself is available in pool for the newer version of the software. Older versions of the software were kept intact in pool for reverting as or when required during testing and hence reliability of the system is enhanced.
This model is easy to adopt as it follows the existing models like increment, prototyping and spiral model during various phases.

3.1 Reusable Component Pool

This phase of Knot Model in CBSD life cycle acts as a central reservoir of the components. All the other three phases are directly interact with it. It is considered as a warehouse of the Knot Model. It is helpful in selecting the desired component which is used for development as per design and specifications. Whenever a new component is developed that becomes the component/member of the pool. And more over the final product also lies with the pool to enhance the library.

3.2 Development of New Component

In this phase a new component is developed whenever the existing component in library/reusable component pool does not satisfy the required software analysis, specifications and design. The by-product of this phase enhances the warehouse.
This phase started with an estimating cost, risk analysis, and then software analysis and its specifications followed by design, coding, testing and feedback.
These steps are carried out considering the services of the required component, portability, in the current system and as well for the future use. Depending upon the component design, one can opt for a suitable existing model like increment, prototyping, and spiral model to carry out this phase. Modularity is the key concern so that a system is decomposed before starting of this phase.[6]

3.3 Modification of Existing Component

This phase of Knot Model adopting reusability whenever the existing components in the reservoir does satisfy the required software analysis, specifications and design up to an extent.

This phase plays a significant role in this model as most of the components are to be modified according to the specifications and design from the existing pool. The reusability lies as most of the component specifications are carried with the new system. But it requires a categorized/indexed form of the components so that at the beginning, a right component is being selected which satisfies most of the specifications in the new system.

3.4 Development of New Component Based Software Development

This is the final phase of this model which again starts from the component pool. This phase provides the shape to the new system as all the desired components are now assembled by resolving the inter-component risk and the process is carried out till the retirement of the software.

As this is the actual software development, so by assembling the components available in the repository, the whole system testing and implementation is carried out. During this, process techniques like black box or white box for testing or other traditional techniques/ tools for development are used. The maintenance stage is not meant for feedback solutions but it is a life time process till the retirement of the software. Maintenance includes the update of software, addition, modification, scalability or any type of improvement on latter stages.

4. Comparisons of Software Process Models

Table 1: Comparisons of Software Process Models

Process Model	***Strengths***	***Weakness***
Waterfall	Simple , concise, easy to execute, logical ordering, For well understood problems, short duration's projects.	Requirements are frozen early, user feedback & changes not allowed, Linear series of actions
Prototyping	Help in requirements elicitation, constructive feedback, reduce risks, leads to better systems.	Heavy process, disallow later changes
Iterative	Quick delivery , reduce risk, allows user feedback, when requirement are not known earlier.	Each iteration have planning overhead, cost increase during iteration, effective management to control iterations
Spiral	Planning and negotiations easier, flexibility in development, for larger projects	Requirement not clear , needs confirmations and high risks
X Model	Reusability, clear requirements, for large systems	Increases complexity, no risk analysis, increases cost
Y Model	Reusability, Solving by analogy, Follows both top down and bottom up approach	Iteration and overlapping during process
Knot Model	Reusability, Easy Planning , requirements clear , no complexity of software applications, reduces risk and development time, reduces cost, applicable on larger & complex systems	Selecting a right component is difficult, reservoir may be huge or difficult to manage

5. Conclusion

In this paper a new - Knot Model for Component Based Software Development Life Cycle is proposed which lays emphasis on reusability considering risk analysis and feedback in each and every phase. This model is best suited for medium or larger complex system's development. It is based on three states of the component 1) When a new component is not available in the repository, then it develops a new component for reuse and place it in the pool 2) When a component is partially available, then modifies it for reuse and places it in the pool. 3) When a component is available in the repository then reuses it during the new proposed system. An utmost care should be taken that each component is created for reuse, so the component is not based on particular application's specification but must carry general specifications. In this model risk is resolved in early stages of each phase. This results in the reduction of cost and time and makes the software more reliable and efficient. Moreover feedback at the end of each phase results in further improvement and revised form of component. It also reduces the cost and time by better management as it resolves the conflicts, if any, during that phase.
This model may require further improvement with the change of technologies in this fast emerging field.

References

[1] Boehm B., "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes, 1986,Vol. 11, No. 4, pp. 14-24.
[2] Gill N. S. and Tomar P., "X Model: A New Component-Based Model", MR International Journal of Engineering and Technology, 2008,Vol. 1, No. 1 & 2, pp. 1-9.
[3] Luiz Fernando Capretz, " Y: A new Component-Based Software Life Cycle Model ", Journals of Computer Science1 (1) : pp.76-82.
[4] Pressman, R.S., Software Engineering: A Practitioner's Approach, McGraw Hill, 6th ed. ,2005.
[5] Srivastava, Chauhan, Raghuraj, "Square Model- A proposed Software Process Model for BPO base software applications", International Journal of Computer Applications, Vol 13, No. 7,2011,pp. 0975-8887.
[6] Yogesh Singh, K.K. Aggarwal, Software Engineering, New Delhi: New Age International Publisher Limited, 3rd Ed.

Dr. Rajender Singh Chhillar is a Professor in the Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He is the former Head (2003-2006) of Department of Computer Science and Applications, M. D. University, Rohtak. Headed the Department of Engineering and Technology from April, 2006 to August 2007. Worked as Director (2003-2010) Computer Centre, MDU and member, monitoring committee of campus wide Networking, M. D. University, Rohtak. He obtained his Master's Degree in Computer Science from Kurukshetra University, Kurukshetra and Doctorate from Maharshi Dayanand University, Rohtak. His research interest includes Software Engineering focusing on Software Metrics, Testing, Data Warehousing and Data Mining. He has more than 70 publications in International and National journals. Dr. Singh authored two books; Software Engineering: Metrics, Testing and Faults, Excel Books publisher, New Delhi; and Application of Information Technology to Business, Ramesh Books House, Jaipur. He is a member of various academic bodies like Academic Council, MDU and Computer Society of India (CSI).

Mr. Parveen Kajla is a Research Scholar in the Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He is coordinator of PG Courses at Vaish Mahila Mahavidyalya, Rohtak and Lecturer in Department of Computer Science and Applications, Vaish Mahila Mahavidyalya, Rohtak. He obtained his Master's Degree in Computer Science from Maharshi Dayanand University, Rohtak and M.Phil(Computer Science) from Chaudhary Devi Lal University (CDLU), Sirsa. His research interest includes Software Engineering focusing on Object oriented and component based metrics.