ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Enhanced Software Development for Reuse Process Model in Component Based Software Engineering (CBSE)

[1]Virendra Kumar, [2]Shabina Ghafir
[1]Dept. of Computer Science, CET-IILM-Academy of Higher Learning, Greater Noida, UP, India
[2]Dept. of Management and Information Technology, Jamia Hamdard (Hamdard University), Hamdard Nagar, New Delhi, India

Abstract
In Software Engineering, Component-Based Software Engineering (CBSE) is becoming an important emerging topic and it the centre of many new research projects. Component Based Software Engineering (CBSE) is concerned with the assembly of pre-existing software components (reusable components) that leads to a software system that responds to client-specific requirements. In this research paper, We have developed the enhanced software development for reuse process model and gave the future prospect of Component Based Software Engineering (CBSE), and reusability component which will reduce the time ,effort and cost of the software development.

Keywords
Component-Based Software Engineering (CBSE), Reusability, Reusable Component, COTS.

I. Introduction
Component-Based Software Engineering (CBSE) is concerned with the development of software systems from reusable parts (components), the development of components, and system maintenance and improvement by means of component replacement or customization. Component-Based Software Engineering (CBSE) is expected to drastically reduce the time spent on developing software through the use of prefabricated components.

II. Reusable Software
Reusable Software is already existed software which is used for developing the new software products, with less effort, time and cost. Software reuse has a positive impact on software quality, as well as on software costs, and productivity. According to Krueger's general view of software reuse [Kru92]: "Software reuse is the process of creating software systems from existing software rather than building them from scratch".

A. Reusable Software Components
Software components and software reuse complement each other perfectly. Using software components to build software systems almost automatically leads to software reuse.
Reusable software components are designed to apply the power and benefit of reusable, interchangeable parts from other industries to the field of software construction. Other industries have long profited from reusable components. Reusable electronic components are found on circuit boards. A typical part in your car can be replaced by a component made from one of many different competing manufactures. Lucrative industries are built around parts construction and supply in most competitive fields. The idea is that standard interfaces allow for interchangeable, reusable components. Reusable software components can be simple like familiar push buttons, Calculator (fig. 2), Slider Beans (fig. 3), text fields list boxes, scrollbars, dialogs, some of them are given below.

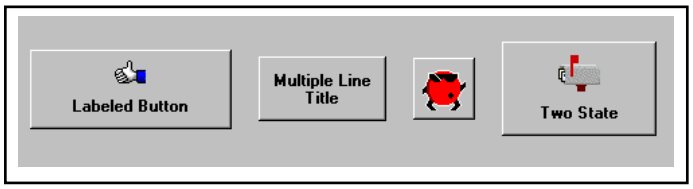


Fig. 1: Button Beans

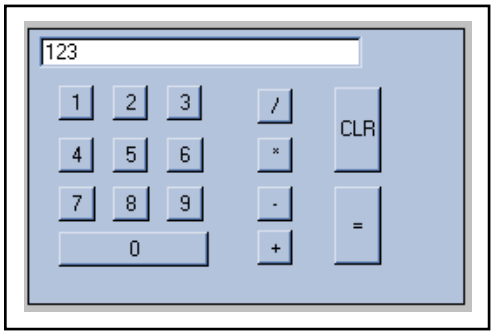
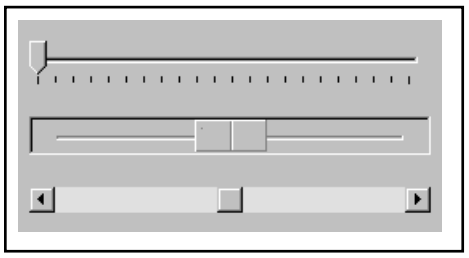


Fig. 2: Calculator


Fig. 3: Slider Beans

B. Types of Software Reuse
Application System Reuse: reusing an entire application by incorporation of one application inside another (COTS reuse),development of application families (e.g. MS Office) Component Reuse: components (e.g. subsystems or single objects) of one application reused in another application Function Reuse: reusing software components that implement a single well-defined function

III. Commercial-Off-The-Shelf (COTS) Based Component Development
Commercial-Off-The-Shelf component development is mostly a bottom-up approach: Assemble a system from existing components. The architecture of the system is secondry to the combinations of components that are available.
From the perspective of the system developer, the strongest driver for this approach is time-to-market A typical question developer will ask is Which components can be used to quickly assemble a functioning system?. For this approach to work, there must be component developers that aim at developing highly reusable functionality. Overall, this approach assumes that the system development process is completely separate from the development process for components.

While the Commercial-Off-The-Shelf (COTS) approach gives instant value in new functionality, there are a number of problems with this approach that can complicate its realization: If components were not designed to comply to a component model, or the components were not designed to fit with the other components in the intended system, this approach may run into problems of composability. This may occur if the semantics are not precisely stated or if the architectural properties of the components are not properly and adequately documented. For Commercial-Off-The-Shelf based development, component assessment plays a much more important role than in component or product-line development architectures.

As an example of this approach consider a Web store. Web store architecture has a fair amount of commonality with the architecture of many business information systems. A Web-store can be assembled by buying components that are sold as building blocks on the software market. Typical components include: Database, Web-Server, Search Engine, Payment Handler, and an inventory control system. In this case, a selection is made of which combination of components is the most appealing, and then an architecture is designed which successfully integrates these components.

IV. Enhanced Software Development for Reuse Process Model

The given model (fig. 4) is based on "Software development for reuse process model" which can be combined with any of the software life cycle models (for example Waterfall Model, Spiral Model etc.). This model depicts how the stored component will be, select for developing the desired application, compatibility with another components during the software development. Here we can see the different parts of this model which are used for software development using reusable components which we have discussed above.

There are the parts of this models which are discussed below.

A. Domain Analysis

Domain analysis means to identify the requirements for the different system to be developed; here we gave the model for multiple system requirements. For example to develop an operating system to add date and time component we will use the preexisting code for date and time.

B. Architectural Design

During architectural design extra effort is now spent to make components more general and more reusable. These components are then incorporated into the storage together with any information that might be of help at a later stage, when searching for or retrieving components. It contains four parts of design .Where the required component is searched in the storage .

1. Design Architecture

Design architecture means to select, evaluate and make the component reusable the required reusable component.

2. Select Reusable Components

To select the reusable component from the storage.

3. Evaluate Components

Evaluate the components according to the new application development.

4. Make Component Reusable

Make the component reusable for another application development.

After the architectural design phase, the storage may contain information about design details, implementation, and testing. For example, test cases might be retrieved and used when testing the system, or we may be able to add our own test results to the storage.

C. Storage Contains the various reusable components

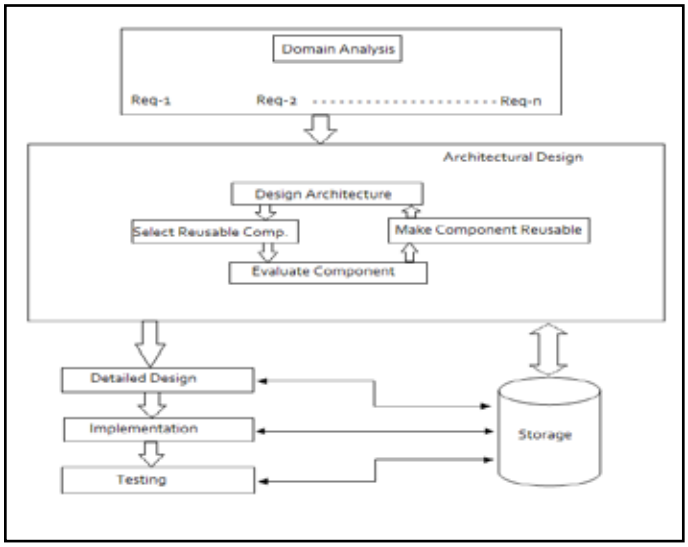


Fig. 4 : Enhanced Software development for reuse process model

V. Benefits of Software Reuse

In the following we elaborate on quality improvements and effort reduction in more detail. Software reuse results in improvements in quality, productivity, performance, reliability and interoperability.

A. Reliability

Using well-tested components increases the reliability of a software system. Furthermore, the use of a component in several systems increases the chance of errors to be detected and strengthens confidence in that component.

B. Performance

Extensive reuse can be worth the effort invested in optimizations. This may yield better performance of a reused component than might be practical for a component that is developed and used only once. However, generalizations that make components more reusable can have a negative influence on overall performance.

C. Quality

Error fixes accumulate from reuse to reuse. This yields higher quality for a reused component than would be the case for a component that is developed and used only once. However, this requires the administration and maintenance of components and is not achieved by simply reusing.

D. Productivity

A productivity gain is achieved due to less code that has to be developed. This results in less testing efforts and also saves analysis and design labor, yielding overall savings in cost. When reuse is being installed, productivity may decrease shortly due to increased

learning effort and the need to develop reusable components. This temporary decrease in productivity should easily be compensated by a long-term increase.

E. Interoperability

Various systems can work better together if their interfaces are implemented consistently. This is the case when they use the same components for these interfaces. Even though written standards improve interoperability, different implementations might differently interpret parts of these standards.

VI. Economical Benefits of Reuse

A. Quality

With each reuse additional component defects are identified and removed which improves quality.

B. Productivity

Since less time is spent on creating plans, models, documents, code, and data the same level of functionality can be delivered with less effort so productivity improves.

C. Cost

Savings projected by estimating the cost of building the system from scratch and subtracting the costs associated with reuse and the actual cost of the software as delivered.

D. Cost analysis using structure points

Can be computed based on historical data regarding the costs of maintaining, qualification, adaptation, and integrating each structure point.

VII. Conclusion

The term software crisis was coined in the late sixties to describe the increasing burden and frustration of software development and maintenance. Programmers have been reusing code, subroutines and algorithms since the early days of programming. Reuse of software components is becoming more and more important in a variety of aspects of software engineering. The given model will help to find the required and refined reusable software component which can be used with any "Software Development Life Cycle " . Software reuse has a positive impact on software quality, as well as on software cost, and productivity.

References

[1] Arvinder Kaur, Kulvinder Singh Mann,"Component Selection for Component based Software Engineering", International Journal of Computer Applications, Vol. 2, No.1, 2010.
[2] Gregory Booth,"Component-based Software Development".
[3] Ivica Crnkovic, Stig Larsson, Judith Stafford,"Component-Based Software Engineering, Building systems from Components", at 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems.
[4] Ivica Crnkovic,"Component-based Software Engineering New Challenges in Software Development".
[5] Xia Cai, Michael R. Lyu, Kam-Fai Wong,"Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes", Software Engineering Conference, pp. 372-379, 2000.
[6] Johannes Sametinger,"Software Engineering with Reusable Components", Springer-Verlag.
[7] Pressman, R.S.,"software Engineering: Apractioner's Approach",7th Edition, McGraw Hill, 2006.
[8] [Online] Available: http://www. http://centurion2.com/SEHomework/SoftwareReusability/SoftwareReusability.html



Mr.Virendra Kumar(Assistant Professor in CS Department), His Educational Qualification is :B.Tech(Computer science and Engg.) from UPTU, Lucknow India , M.Tech (P) (Computer science) from Jamia Hamdard University New Delhi, India. Current Employer is CET-IILM-Academy of Higher Learning ,Greater Noida India, His achievements are Online Certification on "Web Publisher" From Cambridge Intercontinental University USA, Certification in VPN from HCL Infinate Ltd. Lucknow Academic Achievements are: two books are Published on the title 'Computer Organization' &' Information Security and Cyber Law', International Paper published on "Robust RSA for Digital signature ",Paper presented in National Conferences on different topics like GIS based Cloud Computing, Component Based Software Engineering etc.



Ms.Shabina Ghafir (Assistant Professor in CS Department), Her Educational Qualification is M.Tech (Software Engineering) from Aligarh Muslim University, Aligarh, U.P. India ,B. Tech (Computer Engineering), G.B.Pant University of Agriculture & Technology, Pantnagar (Nanital), India. Presently working as Assistant Professor in Jamia Hamdard (Hamdard University) New Delhi Her Working Area: Software Engineering and Case Tool , System Analysis and Design Operating System.