Component Based Software Engineering: Benefits And Risks

Noopur Goel

ABSTRACT

Component Based Software development (CBSD) is an approach to develop a software system with the help of reusable software components. It brings certain benefits such as increased quality, productivity and reduced time to market. Yet, like the traditional single system software, CBSD has some risks also. This paper investigates few risks associated with the CBSD process.

Keywords: Components, Component based Software development, Risks.

1. INTRODUCTION

The software systems are growing in size and complexity, and the customers are demanding more dependable software that is delivered and deployed more quickly. This leads to the practice of reuse rather than *reimplementing* the software components. Components are higher level of abstractions than objects and are defined by their objects. All implementation details are hidden from other components. Component-based software engineering (CBSE) is the process of defining implementing and integrating or composing loosely coupled independent components into systems [1]. It consists of two parallel activities [2]:

i. Domain Engineering: investigates an application domain with the definite purpose of finding functional, behavioral and data components that are candidates for reuse. These components are placed for reuse libraries.

ii. Component-Based Development: considers requirements from the customer, select an appropriate architectural style to meet the objectives of the system to be built, and then selects, qualifies, adapt and integrates the components to form a subsystem and the application as a whole.

Essentials of CBSE [1] are:

* Associate Professor, Department of Computer Aplications, VBS Purvanchal University Jaunpur, U.P., India, Email: noopurt11@gmail.com

i. Independent components which are completely specified by their interfaces.

ii. Component standards that facilitate the integration of components.

iii. Middleware that provides software support for component integration.

iv. A development process that is geared to CBSE.

Practitioners identify the following key CBSD advantages in future software development efforts [3][4]:

i. Reduced lead time. Building complete business applications from an existing pool of components;

ii. Leveraged costs developing individual components. Reusing them in multiple applications;

iii. Enhanced quality. Components are reused and tested in many different applications; and

iv. Maintenance of component-based applications. Easy replacement of obsolete components with new enhanced ones.

v. Effective management of complexity

vi. Increased productivity

vii. Greater degree of consistency

viii. Wider range of usability

2. RISKS IN SOFTWARE DEVELOPMENT

Though risks in a project are specific to the project, some are common and they can happen in any Software development process. Listing these risks would be good for identifying these risks in a particular project. Based on surveys of experienced project managers, Boehm has produced a list of the top ten risk items likely to compromise with the success of a software project:

1. Personnel shortfall: this involves just having fewer people than necessary or not having people with specific skills that a project might require.

2. Unrealistic schedule and budget: this risk happens very frequently due to business-related and other reasons. It is very common that high-level management imposes a schedule for a software project that is not based on the characteristics of the project and is this could be unrealistic. This risk applies to all projects.

Project-specific risks in cost and schedule occur due to underestimating the value of some of the cost drivers.

3. Developing the wrong software function: projects run the risk of developing the wrong software if the requirements analysis is not done properly and if development begins too early.
4. Developing the wrong user interface: this item is also related to requirements. Often improper user interface may be developed. This requires expensive rework of the user interface later or the software benefits are not obtained because users are reluctant to use it.
5. Gold plating: gold plating refers to adding features in the software that are only marginally useful. This adds unnecessary risk to the project because gold plating consumes resources and time with little return.
6. Continuing stream of requirements changes: some requirement changes are to be expected in any project, but sometimes frequent changes are requested, which often reflect the fact that the client has not yet understood or settled on its own requirements.
7. Shortfalls in externally furnished components: if the project depends on externally available components - either to be provided by the client or to be procured as an off-the-shelf component - the project runs some risks.
8. Shortfalls in externally performed task: this type of risk is related to reuse and is described in detail later in the paper.
9. Real-time performance shortfall: if the project behaves like the requirements specified in requirement specification but not within the limits of real-time requirements, the project would fail.
10. Straining computer science capabilities: If a project relies on technology that is not well developed, it may fail. This is a risk due to straining the computer capabilities.

3. RISKS IN CBSD

CBSE has two parallel development processes as mentioned above: Domain Engineering and Component-Based Development. It will not be out of place to mention again that some other types of risks are there in a CBSD. Each of these two processes can be performed independently of each other with proper feedback from the second process to the first process. Risks are associated with both processes and some risks are there which are transferred from one process to another. In spite of a wide range of benefits associated with Component-Based Software Development (CBSD), some risks and challenges also come along with it [4].

i. Time and effort required for development of components: In comparison to building a unit for a specific purpose, it requires three to five times the effort to build a reusable component. Many practitioners have witnessed that the reusable component pays off after its fifth reuse. A reusable component, during its launch, is exposed to changes and after its few reuses comes to a stable state.

ii. Unclear and ambiguous requirements: Requirement management, an important part of the development process, has the main aim of defining complete and consistent requirements of component. In the case of reusable components, since it covers a domain of similar applications, some of the requirements (functional and non-functional) are not known in advance.

iii. Conflict between usability and reusability: It requires a component to be more general, scalable and adaptable to become it widely reusable. This leads to a more complex (thus more complicated to use) and more demanding (thus more expensive to use) of computing resources.

iv. Component maintenance costs: The component maintenance costs can be very high since it must adhere to various requirements of different applications running in different environments, with different level of maintenance support as compared to a single system maintenance costs.

v. Reliability and sensitivity to changes: The components and applications are having different life cycles and different kinds of requirements. There is a risk that a component may not satisfy some applications requirements or some component characteristics may be obscured and not known to application developers. So, if a change is introduced at the application level there is a risk of application failure.

4. CONCLUSION

It is well known that the success of any project depends on the decisions taken during the various activities during the software development process. In the same way, the success of CBSD also depends on the two independent development processes involved in CBSD. The listing of these risks would help the developers and users of the components and software as a whole to identify and prioritize the risks involved in the project. This would further be beneficial to define the ways to address those risks.

REFERENCES:

1. Ian Sommerville, Software Engineering, 9th Edition, Pearson Education India, India, ISBN 978-81-317-6216-5, 2011
2. Pressman R., Software Engineering: A Practitioner Approach, 6th Edition, McGraw Hill: New York, NY 10020. ISBN 007-124083-7, TMH, 847-857. 2001
3. Padmal Vitharana, Risks and Challenges of Component-Based Software Development, Communications of the ACM August 2003/Vol. 46, No. 8 pp. 67-72.
4. Ivica Crnkovic, Component-based Software Engineering-New Challenges in Software Development, Software Focus, John Wiley & Sons Ltd. Volume 2, Issue 4, 2002, pp. 127-133.
5. Barry W. Boehm, Software Risk Management: Principles and Practices, IEEE Software, vol. 8, No. 1, 1991, pp. 32-41.

●