ISSN: 0975–833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 3, Issue, 5, pp.065-068, May, 2011

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

REVIEW ARTICLE

CHALLENGES AND SOLUTION OF SOFTWARE ENGINEERING AND DEVELOPMENT: A REVIEW

*** Rev. Engr. Friday O. Okafor**

Department of Mathematics/Statistics/Computer Science, Michael Okpara University of Agriculture, Umudike

ARTICLE INFO

Article History:
Received 13th February, 2011
Received in revised form
9th March, 2011
Accepted 27th April, 2011
Published online 14th May 2011

Key Words:
Challenges,
Software engineering,
Software components,
Software Architecture,
Component-based development,
Component-based software engineering.

ABSTRACT

The ever-increasing demand in the use of software in business, industry, administration, games and researches have made software engineering and development more complex. This calls for the need for higher-level abstraction of software systems in order to develop real-world systems that meets the new challenges facing software engineering and development process. This paper presents component-based software engineering (CBSE) as a solution to the complexities facing software usability and applicability. This technology (CBSE) addresses the development of systems as an assembly of components, the development of components as re-usable entities, and the maintenance and upgrading of systems by customizing and replacing such components (parts). The author assesses the challenges of this emerging technology and discusses its solutions, merits (attractions), and implications for the software engineering and development process in a developing country like Nigeria.

INTRODUCTION

With the number of software users expanding geometrically on daily basis and the proliferation of more advanced software functionalities in communications, business, industry, games, administration, education and research to mention but a few, software engineering development has become more complex and large. This indeed has created the need for higher but better approach to software development other than the traditional software engineering approach. Software is no longer marginal in technical systems but has become a central factor in many fields of endeavor. System features are now based on software functionality instead of other characteristics and have become most vital factors in competing on the market, for example, in education and research, in car industry and in the service sector. More so, majority of users of software today are non-professionals. All these and more have placed new challenges on software development. Thus, usability, robustness, simple installation and integration are the most important features. Due to the wider area of software utilization, the demand for the integration of different areas increased. This has resulted to software becoming increasingly large and more complex with attendant problems such as: Delay in completion of software project and delivery with the deadline, Failure to meet the software budget, and Quality requirement. To meet these challenges, component-based development (CBD), narrowed down to especially its defined systematic approach – the component-based software engineering (CBSE) is the solution. The component – based development is a pragmatic software development approach that is capable of coping with this complexity and at the same time capable of adapting quickly to the changes. This relatively new software engineering and development practice – the CBD has removed a lot of chores and proffered solutions to these challenges facing software development. The challenges and solutions, advantages, disadvantages and risks in using component-based software development and other vital issues relating to software engineering development are discussed in the subsequent sections of this paper.

Corresponding author: revmachi_4@yahoo.co.uk

New challenges in software engineering and development and the possible solutions

The leap in the utilization of software in all facet of life, have indeed made software development large and complex. However, the traditional software engineering development was able to address the challenges of increasing complexity and dependence on external software (outsourcing) by focusing on one system at a time and on delivery deadlines and budgets without consideration to the evolutionary needs of the system. The ignoring of the evolutionary needs of the system by the traditional software development process has led to quite a number of problems which may include: In ability of most projects to meet their: deadline, budget, quality requirements and ever-increasing cost associated with software maintenance. These challenges call for software development that must cope with complexity and adapt quickly to changes. If new software products are each time to be developed from the scratch, these goals cannot be achieved. The key to the solution to this problem according to Crnkovic (2001:2), is reusability and component-based development (CBD), which seems to be the best approach to software

engineering development. Daramola and Bamigbola (2006:17) defined component-base development as the conceptualization of software systems as a composition of individual reusable parts (components), that are integrated for coherent interaction in order to achieve the specified requirements of a system. They maintained that component-based development is a software componentization that involves the dividing software systems into components for the purpose of easy development, deployment and maintenance. Crnkovic, Stanford, and Larsson (2000) identified a software component as a modular, deployable and replaceable part of a system that encapsulates and exposes a set of interfaces. On the other hand, an interface according to Daramola and Bamigbola (2006:17) specifies the set of services provided by a component. However, component-based development which forms a higher-level abstraction of software systems encourages the development of software system using an approach that is similar to industrial component-based assembly line procedures. In practice, software systems are built in component-based development by assembling components (parts) already developed (e.g. COTS) and prepared for integration. **Rainer (2008),** identified and summarized software development challenges as that of communications, collaboration and human issues. However, below are some of the challenges presently facing component-base development:

1. **Problem of compatibility, components maintenance and upgrading:** Compatibility principle requires a component to be replaced easily or added to new parts of a system if such component is compatible with its previous versions. Thus for a component to be reusable, there must be compatibility between various versions of the components. This is also applicable to the system maintenance and upgrade. When compatibility does not exist, it becomes difficult to maintain and upgrade components due to various system evolutions (viz: evolution of system requirements, technology, society and business changes) [Crnkovic and Larsson (2002:61) vol 3]. However, selecting the right people with skill, experience and right communication ability will reduce the chores involved with this challenge.
2. **Configuration management challenge:** Due to the heterogeneous nature of these systems constituted from several components with different behaviors, managing the configuration and possible integration becomes a great challenge. The configuration management is not only of technology but of business also. The solution to this challenges also boils down to gathering a vast number of candidate of proven managerial / technical experience and tools. Finding the components which may be used in the system requires high level of collaboration among the candidate and this is a times complex.
3. **Quality control and profitability issues:** Component-based development is made up of components bought or outsourced. These components (parts) would have come from different providers, and hence, quality control becomes more difficult. Similarly, copyrights, maintenance and upgrade of components become more complex. This complexity and outsourcing of several components reduces the net profit.
4. **Component generic, Efforts And Lead-Times:** It is very difficult to build a reusable component that is simple to use and sufficiently generic to cover the various aspects of its use. Thus creating a proprietary component to be used in the system becomes less attractive as it requires more efforts and lead-time. More so, the demand on the component interface and behavior are usually varied in different components and products. While some components require a high level of abstraction, others may require the interface to be on a more detailed level. These different types of requirement often lead to several variants of component, which can be used on different abstraction [Daramola and Bamigbola (2006:19)].

Other challenges: There are several other research and general challenges (issues) affecting the practice of component-based development: These include: Legal issues, Standardization and certification issues, Software metrics, Availability of procedures for verification, testing and checking of component-based systems, Migration between different platforms, Development environment tools. e.t.c.

However, the solution to the above challenges remains a resort to selection of skilled experienced people (candidates), communication and interactive collaboration as well as systematic approach to component-based development at the process and technology levels. Developing a reusable component requires a lot of time and demands for more resources than developing a component that serves a particular case. Several human efforts goes into building the reusable components and certifying their behavior and functionality. A solution to these challenges is to form teams and assign them with the responsibility of developing each a component of software in parallel with effective on-line communication. There is the need to also foster closer communication among the developer teams, so as to avoid a situation where the components will be at much variance with one another. This will also foster easily and efficient integration. Team leaders of each group must frequently meet regularly. Another challenge is the problem of evolutionary changes such as creation of new departments, splitting of existing ones etc. that may have occurred after the development of the system has commenced. These kinds of challenges will surely cause disruption of the initial design of some of the components and hence resulting to emergence of new interfaces and methods. This type of situation therefore calls for need for integration and interoperability among components.

Merits of component-based system

Component-based system has numerous advantages and benefits. Some of which include:

1. Component-Based system provides support for the development of systems as assemblies of components, the development of components as reusable entities and the maintenance and upgrading of systems by customizing and replacing their components. Hence component-based development

gives the opportunity of practicing abstraction at component level, higher than the object level.
2. It gives a better understanding and perception of software systems composition.
3. It encourages reusability which in turn breeds reliability of composed systems.
4. It deals with the complexities associated with most traditional software development approaches, since what the developer does is more of integration instead of building from the scratch. This also reduces time and cost of development
5. Component-based system is more efficient, reliable and has performed better in many situations (applications)

The disadvantages and risks in using component-based development

In spite of the numerous merits / benefits of component-based system, there are many demerits and risks in using this system. These include:

1. **Component Development Time And Effort Required:** Mrva (1997) stated that as a rule of thumb, the overhead cost of developing a reusable component, including design plus documentation, is recovered after the fifth reuse. Hence the building of a reusable unit requires increased time and effort than that for one specific purpose. This increased time and effort in development process is a discouragement.
2. **Requirement management:** The requirement management of every development process is a vital aspect and is aimed at defining the actual but consistent component requirements. In component-based systems most of the components requirements are unclear, complex, unknown and difficult to predict. This issue applies to both functional and non-functional requirements.
3. **Maintenance costs:** In component-based development systems, the application maintenance cost is low while the component maintenance cost is high. This is because, the component must respond to the various requirements of various applications running in various environments, with different reliability requirements that may require different level of maintenance support.
4. **Reliability and Sensitivity To Changes:** Most components and applications have different lifecycles and requirements. Thus, there is the risk that a component may not completely satisfy the application requirements or that it may have some characteristics not known to the developers; which can introduce changes (such as updating operating systems / other components or changes in application), capable of causing system failure:

However, we need a systematic approach to component-based development at the process and technology levels, if we want to avoid these problems and risks. Hence the component-based software engineering is the solution.

Component-Based Software Engineering

Recent emergence of new technologies has greatly increased the possibilities of building systems and applications from reusable component. Component-based software engineering (CBSE) which is one of these approaches has its major goals as:

i. Provision of support for the development of systems as assemblies of components.
ii. Provision of support for the development of components as reusable entities and
iii. Provision of support for the maintenance and upgrading of systems by customizing and replacing their components.

Other objectives of component-based software engineering include: component specification, technologies, methodology for application of software engineering discipline in component-based development. In fact the success of software development in the future will depend very much on the successful establishment of CBSE and this is supported by Gartner Group (2001) who stated that "By 2002, 70 percent of all new applications will be deployed using component –based application building block". Nevertheless some of the relevant trends and challenges in the near future are discussed in the proceeding sections.

Component Specification

According to Szyperski (1998), a software component is a unit of composition with contractually specified interface and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parts. The most important features of a component is the separation of its interface from its implementation. This separation is different from those found in many programming languages (such as ADA or Modula-2) in which declaration is separated from implementation or those in object-oriented programing languages in which class definitions are separated from class implementations. Indeed, the integration of the component into an application should be independent of the component development lifecycle and there should be no need to recompile or re-link the application when updating with a new component. Secondly, another vital characteristic of the separation is that the component implementation is only visible through its interface, especially for components delivered by a third party. The above definition which focuses on the use of components was silent on how to design, implement and specify a component. However, below are other definitions that point to other aspects of component-based development. For example, a strong relation exists between object-oriented programming and components. Component models (component standards) COM/DCOM, .NET, Enterprise Java Beans (EJB), and CORBA component model (CCM) relate component interface to class Interface (CrnKovic (2001:4)]

Conclusion

Reusability and component-based development demonstrated in component-based software engineering is the best approach and seems to be key to solution to the challenges facing software engineering and development. In this paper, the author has successfully assessed the challenges and discussed the possible solution, benefits and implications of component-based development for software engineering and development. The paper presented software engineering as a technology that addresses the development of systems as an assembly of components, the development of components, the

development of components as reusable entities, and the maintenance and upgrading of systems by customising and replacing such components(parts). The ultimate objective of component-based development is to build software that will significantly change the development of software and software use in general.

REFERENCES

Crnkovic, I. 2001:2,4: Component-Based Software Engineering-New Challenges In Software Development, Software Focus, vol. 2(14), Sweden.

Crnkovic, I.,Stafford, J., and Larsson, S. 2000: Composing Systems From Components; IEE Conference And Workshop On Engineering Of Computer-Based Systems, Lund University, Lund.

Daramola, J. O. and Bamigbola, M. O. 2006:17,19: The Attraction And Challenges Of Software Componentization: An Experience Of Auto Budget, The Journal Of Computer Science And Its Applications, Vol. 12, No. 1, Olaejinson Ventures, Lagos.

Gartner Group 2001: Workshop on Component-Based Software Engineering. Sourced: http://www.gartner.com

Larsson, M and Crnkovic, I. 2002:61: Challenges Of Component-Based Development' Journal Of Systems And Software vol. 3.

Mrva, M. 1997: Reuse Factors In Embedded Systems Design, High Level Design Techniques Dept. Siemens A. G., Munich, Germany.

Rainer, Otterbach, 2008: Trends, Challenges and Solutions For Automotive Software Development, User Conference, North America.

Szyperski, C. 1998. Component Software-Beyond Object-Oriented Programming, Addison Wesley, London.

****