ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# Component-Based Software Development: Linear perspective of Software Engineering

Jawwad Wasat Shareef *
Department of Mathematics & Computer Science,
Rani Durgavati University,
Jabalpur, India
javedshareef@yahoo.com

Rajesh Kumar Pandey
University Institute of Computer Science & Applications,
Rani Durgavati University,
Jabalpur, India
rkpandey18@rediffmail.com

***Abstract:*** The idea of Component-based software development (CBSD) is to build large software system by assembling a set of previously developed software components that can be independently deployed, configured and connected together. The basic foundation of this approach is that common parts should be written once rather writing them again and again from scratch and that common system should be assembled through reuse of these common parts. Component Based Software Engineering (CBSE) is a paradigm that handles efficiently the entire lifecycle of component-based products. It has given more attention on technologies related to design and implementation of software components and systems built from it. CBSE aims at constructing and designing systems using a pre-defined set of software components mainly created for reuse. CBSE embodies the "the 'buy, don't build' philosophy", that shifts the emphasis from programming software to composing software systems [1]. This requires established methodologies and tool support covering the entire component and system life cycle including organizational, technological, marketing, legal and other aspects. The new software development process is much different from the traditional approach; with time it has now been known that pure technologies alone are not enough. The life cycle and software engineering model of Component-based software development (CBSD) is much different from that of the traditional ones [2]. This paper makes an assessment as to how CBD has progressed fulfilling the promises with linear development stages covering different perspectives and challenges faced by this technology in Software engineering.

***Keywords:*** **C**omponents; Component Model, Commercial Off-the Shelf Software (COTS), Component-Based Software Engineering (CBSE), Component-Based Software Development (CBSD).

## I. INTRODUCTION

The use of personal computers and internet is spreading its wings day by day which make computers commodity goods, creating a new market and users. New users, most of them are consumers; require to drastically reducing the price and/or cost of software in order to match the ever-decreasing hardware price. The use of components is the primary source of the productivity and quality. It is the law of nature in any matured engineering discipline [3]. Making applications from software components has been a dream in software engineering community since its very early time. As quoted in the literatures, McIlroy wrote in the NATO conference in 1968 [4];

"My thesis is that the software industry is weakly founded, in part because of the absence of a software components subindustry. … A components industry could be immensely successful".

However, wide spread reuse of software components over the industry has not come true. The last decade has shown that Object-Oriented approach is not enough to meet the requirements as compared with the fast changing requirements of present-day software applications, in spite of having strong features like objects, inheritance, reuse and others. In Object-oriented development the primary concern is of designing quality classes then in component based development, the term object has been replaced by Components. Parts of the system can be obtained and by reusing these parts which have already been 'tried and tested'. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify. In the same way, in CBSE, by reusing an existing component, a developer does not have to build it from scratch, which saves a lot of time thus avoiding hard work in establishing the usefulness and in testing that component [2].

To meet the given challenges, software development must be able to cope with complexity and to quickly adopt those changes. CBSE uses Software Engineering principles to apply the same idea as Object oriented programming to the whole process of designing and constructing software systems. It focuses on reusing and adapting existing components, as opposed to just coding in a particular style. CBSE encourages the composition of software systems. In context of CBSE comes Component-Based Development (CBD), which plays an important role in Software engineering. Main task of CBD is to build systems comprising of already built software units or components, by encouraging reuse of pre-developed system pieces rather than building from scratch [2].

## II. EVOLUTION OF COMPONENT TECHNOLOGY

The foundation of any CBSD methodology is its underlying component model, which defines what components are, how they can be constructed, how they can be composed or assembled and specifies the standards and conventions that are needed to enable composition of independently developed component. Within CBSD we also distinguish development of components from development of systems. In component-based system development, we focus on identification of reusable entities and selection of components that fulfills system's requirements, but in developing component our focus is on reusability [5].

Component definition given by Heineman and Councill's [6] states that:

"A [component is a] software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard".

Which means some technology is needed to make the components work together in a standard way. The most commonly used component technologies that are applied for component-based software development are Sun Microsystem's Javabeans and Enterprise JavaBeans (EJB), Microsoft's COM (Component Object Model), DCOM, .NET Framework and Object Management Group's (OMG) Corba Component Model (CCM). These technologies have been compared based on their functionality and mechanism along with their pros and cons, thus providing a roadmap in selecting the appropriate technology for CBSD [7]. The CBD approach appears to be the right approach. This results in a number of advantages like more effective management of complexity, reduced cost, in-time, flexibility and high quality.

## III. THE PROMISES OF COMPONENT-BASED DEVELOPMENT

In a system, components are small independent parts, whereas a large system as a whole can be seen as a component as well. Expectations from components are reusability, flexibility, interoperability, and maintainability.

### *A. Reusability:*

A component is a type, class of objects or any other work product that has been specifically engineered to be reused. CBD appears to be the best development approach, mainly because of its capacity for reusability and, therefore, it's potential for saving time and effort. CBD enables the development of components which completely implement a technical solution or a business aspect. Such components can be used everywhere. Functionality, be it technical or business oriented, has to be developed and implemented just once, instead of several times. This is a good thing from the point of view of maintainability, robustness and productivity. The market for CBD tools and frameworks has shown a great rise [8]. Reuse is an important feature of the software, and it also is the basis for development of software industry. It runs through the reuse and development of software technology [9].

### *B. Flexibility:*

Run time components can work independently and are less dependent on their environment (hardware, system software, other applications or components) if they are designed properly. Therefore, component-based systems are much more adaptable and extendable than systems traditionally designed and built. Usually, components are not changed, but replaced. This flexibility is important in terms of *hardware and system software and* functionality [8].

***a) Hardware and System Software:*** Component-based systems are less sensitive to changes in the foundation (for example: the operating system) as compared to traditional systems. This results in a more rapid migration from one operating system to another. This results in the possibility of a system which is technically heterogeneous environment [8].

***b) Functionality:*** Component-based systems are at a functional level much more adaptable and extendable than traditional systems, because most of the new functionality is reused or derived from already prebuilt components [8].

### *C. Maintainability:*

In a component-based system, a piece of functionality ideally is implemented just once. It is self-evident that results in easier maintenance, leads to lower cost, and a longer life for these systems. In fact, the distinction between maintenance and construction will become very vague, and completely disappear after some time. New applications will consist of a very large part of already existing components. Building a system will look more like assembly than really building. Moreover, the large monolithic systems will disappear resulting in a blurring of the borders between the systems [8].

## IV. RISKS IN COMPONENT-BASED DEVELOPMENT

The aim of CBD is to build systems as an assembly of components such that the development of components as reusable entities and the maintenance of the system by customizing and replacing such components [2]. The motivation behind the use of CBD is to reduce the development cost, time to market and provide a system that is efficient in meeting the changing customer demands. There are number of risks and challenges associated with CBD. One of the potential risks for the CBD is what if the primary supplier of the component goes out of business, or stops supporting the current version of the component. On the other hand, if the system demands high quality, how we can assure that the component will be compatible with the requirements. The potential risks of CBD [2, 6, 10, 11] are summarized in Table-I. Thus there is a need for careful planning to meet these risks by defining guidelines, standards and open architecture for CBD.

Table: 1 Risks of Component-Based Development (CBD)

| Sr. No. | Risks | Description |
|---|---|---|
| 1. | Locating compatible components | To search for a compatible or suitable component from repositories or on internet, a software engineer must be confident in finding those components, before they routinely include a component search as part of their normal development process. |
| 2. | Interoperability | CBD poses a major challenge of ensuring that component services are provided through standard interfaces to ensure interoperability. |
| 3. | Requirement satisfaction | The component search is performed on a wide variety of component repositories, as well as on internet, even after finding a suitable component there are chances that it might not perform the specific function or might fail to interoperate. |
| 4. | Testing of components. | The components can be used in different set of applications which complicates the testing process, therefore each component must be tested for verification and validation. |

Thus it is necessary to have a systematic approach to Component-based development, to avoid problems and risks and to take full advantage of this technology.

## V. COMPONENT-BASED DEVELOPMENT – PROCESS LIFE CYCLE

The term Component-Based Software Development (CBSD) is an appropriate and methodical approach, which involves the construction of an application by using prebuilt chunks, which were developed at different times, at different locations, by different humans, and possibly with different concept and uses in mind [13]. CBSE is that branch of Software Engineering which covers both component development and system development, both have different approach. In CBD the main focus in on reusability of components, there maintenance and the development of new software systems from reusable components [13]. A component must possess qualities like it must be easy to understand, well specified, sufficiently general, easy to adapt, easy to deliver and deployable and easy to replace. Component must have simple interface and physically and logically separated.

With software development proceeding at Internet speed, in-house development of all system components may prove too costly in terms of both time and money. Large-scale component reuse or COTS component acquisition can generate savings in development resources, which can then be applied to quality improvement, including enhancements to reliability. Thus reducing time to market by shifting developer resources from component level development to integration, increased modularity also facilitates rapid incremental delivery, and offer product upgrades as various components evolve. These advantages bring related disadvantages like integration difficulties, performance constraints, and incompatibility among products from different vendors [14]. If components are selected too early in the process, the system obtained may not meet all the requirements. CBSD uses the concept of integrating pre-existing components where the components are not designed to meet a specific – project requirements. COTS components are built to meet the market requirements. The difference between traditional development and component based development is shown in Table-II.

Table: 2. Difference between Traditional Development & CBSD Approach [15], [16], [17].

| ***Major Points*** | ***Traditional Development*** | ***CBSD*** |
|---|---|---|
| a. User Centered | developer controls the development process | User does not have to depend upon development team. User produces useful systems using application domain knowledge |
| b. Reusability | Development-time | Run-time |
| c. Reliability | Dependent on Developers' ability | Thread-safe and secure |
| d. Design | Dominated by optimization decisions | Pre-built software components are dominated by selection decisions |
| e. Role of Architecture | Monolithic software application | Independent parts of software |
| f. Integration | It is the tail end of an implementation effort | System design involves the selection of components |
| g. Interoperability | Development is restricted with one technology on one platform | Provides communication between different technologies |

## VI. COMPONENT-BASED DEVELOPMENT – DIFFERENT STAGES

Crnkovic, I. [2] compared Water-fall model with Component-based approach. Figure-1. shows the waterfall model and the meaning of the phases. In traditional water-fall model the Identifying of requirements and design is combined with finding and selecting components in CBD approach. The design encapsulates the system architecture design and component identification/selection.

The development cycle different steps are:

a. Find components that may be used in the system. All the components possibly to be used are listed here for further analysis.
b. Select the components that meet the best coverage of requirements and suit the component model.
c. A proprietary component can be created to be used in the system, however this procedure is less attractive as it requires more efforts and takes more time.
d. Adapt the selected components which suit the existing component model.
e. Compose and deploy the components using a framework for components, functionality is provided by component models.
f. Replace earlier with later versions of components,

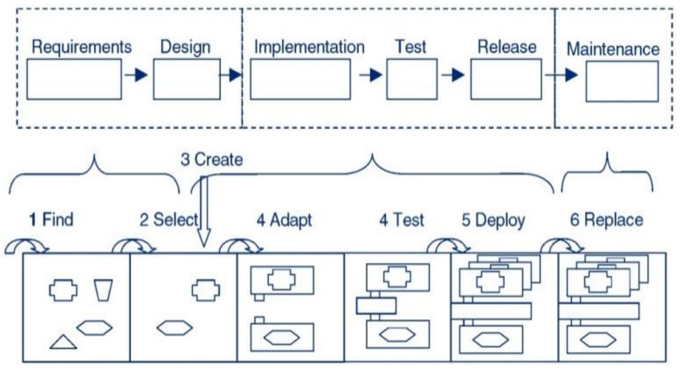

Figure: 1. The Development cycle compared with water-fall model, Crnkovic, I. [2]

In Component based development there are many other sub-areas like software configuration management, software metrics, testing, software configuration management, legal issues, project management, certification and standardization which needs more specific methods and technology management.

## VII. CHALLENGES FACED BY COMPONENT-BASED SOFTWARE ENGINEERING

Component-based software engineering (CBSE) has shown significant prospects in rapid production of large software systems with enhanced quality, functionality and reduced cost, but in spite of these good factors there are many challenges faced by CBSE, according to Crnkovic, I. [18] these are –

### *A. Component Identification:*

To look for components that are available locally or from trusted source, some-times users have fear that these components will not work as advertised. It is quite possible that once a user acquire a component from a component vendor, after some time the firm might winds up or the vendor does not brings an updated version of that component. In such cases how a user can be sure or satisfied that he or she will not face any problems related to component.

### *B. Component Selection:*

Once component has been identified, from a list of specific components, it is not sure the component will meet the requirements. CBD fundamental approach is the reuse of existing components. As there are many uncertainties in the process of component selection, there is a need for a strategy to managing risks in the component selection and evolution process.

### *C. Component trustworthiness:*

Delivery of component is in binary form and the component development process is outside the control of component users, the component trustworthiness becomes of great importance. Associated with the concept of 'trustworthiness' are example- reliability and robustness, but there is no formal definition and understanding of 'trustworthy', no standardized measurement. The impact of different aspects of trustworthiness on system attributes is not known.

### *D. Component certification:*

Certification of components can be done by classifying them. It is a standard procedure in domains, but not yet established in general and especially not for software component [19, 20].

### *E. Prediction of Component Composition:*

Assuming that all the relevant attributes of components can be specified, it is not known how these attributes determine the corresponding attributes of systems of which they are composed. The ideal approach to derive system attribute from component attributes is still a subject of research. A question remains — 'Is such derivation at all possible? Or should we not concentrate on the measurement of the attributes of component composites?' [21].

### *F. Maintaining Component Based Systems on Long term basis:*

Maintaining component-based systems on long term basis is more complex, when systems which include sub-systems and components with independent life-cycles are involved, these raises number of issues –

a. Technical issue: by simply replacing a component can a system be updated technically?
b. Administrative and Organizational issues: which of the components can be updated? Which of the components should be updated? Which of the components must be updated?
c. Legal issue: In case if a system fails, who is responsible for the failure, is the producer of the system, or producer of component?

Maintainability of such systems is a risk that many such systems will be troublesome to maintain.

### *G. Requirements management and selection of components:*

Requirements management is a complex process. A problem of requirements management is that requirements in general are incomplete, imprecise and contradictory. The fundamental approach in CBD is the reuse of existing components. The process of engineering requirements is much more complex as the possible candidate components usually lack one or more features which meet the system requirements exactly. In addition, even if some components are individually well suited to the system, it is not necessary that they do not function optimally in combination with others in the system. These constraints may require another approach in requirements engineering — an analysis of the feasibility of requirements in relation to the components available. As there are many uncertainties in the process of component selection there is a need for a strategy for managing risks in the components selection and evolution process [6, 22].

### *H. Component Development Technologies:*

There are commonly used component development technologies, but they exhibit unclear characteristics, each having their own limitations [7], they are rigid and not an easy to use technologies.

### *I. Composition of Components:*

A large software system may include a number of components these components may include sub-components also. Composition of components sometimes is also treated as components. When complex structures are involved the problem of structure configuration rises. It is quite possible that one same component may be included in two compositions. Then what will be the status –

a. The same component will be treated as two different entities or assumed to be the same entity?
b. What happens if these components are of different versions?
c. Which version will be selected?
d. What happens if these versions are incompatible?

There are still problems with dynamic updating of components, which needs further research [23].

### *J. Component tools for Development:*

Software engineering focuses on providing practical solutions to practical problems, and the existence of appropriate tools is necessary for a successful CBSE performance. Development tools, such as Netbeans, Visual Basic, Eclipse have proven there successful performance, but there is an extreme requirement of many other tools such as –

a. Component selection tools
b. Component configuration tools
c. Evaluation tools
d. Component repositories tools
e. Tools for managing the repositories
f. Component test tools
g. Component-based design tools
h. Run-time system analysis tools,

These are some of the many challenges which are being faced by CBSE. The goal of CBD in software engineering perspective is to standardize and formalize all disciplines related to this field. The success of the CBD approach depends directly on further research and the implementation of CBSE.

## VIII. CONCLUSION

Component-based software engineering shifts the emphasis from programming to composing software systems. Component-based software development is a one step ahead of Object oriented development. CBSD is a linear development and has been accepted in industry as a new effective development paradigm by emphasizing on the designing and construction of large complex systems, using reusable software components. The same approach is being successfully used in other areas of engineering like Automobile industry and Consumer electronics industry. Use of COTS software components increases the productivity and helps in easy development of systems. CBSD has a bright future, if established globally will certainly put strong impact on the quality of software development, but there are certain unusual problems for software development, which have to be handled in a systematic way in order to make it widely acceptable and successful. An attempt has been made to give the reader an understanding of CBD, expectations from CBD, risks involved its life cycle in the field of Software engineering, providing to the reader information about different stages of CBD in a linear path, discussing some major activities and challenges faced by CBSE.

## IX. ACKNOWLEDGMENT

The authors would like to express their cordial thanks to Rani Durgavati University, Jabalpur for providing the facility of Research in their University and also to the reviewers for their valuable advice.

## X. REFERENCES

[1] Roger, S. Pressman., Software Engineering- A Practitioner's Approach, 5th Edition, McGraw Hill International Edition, Ch.27, pp: 721, 2001.

[2] I. Crnkovic and M. Larsson, "Challenges of component based development", J. Syst. Software, 61, (3), pp. 201–212, 2002.

[3] C. Szyperski, Component Software, Addison-Wesley, 1998.

[4] M.D. McIlroy, Mass-Produced Software Components, Software Engineering Concepts and Techniques (1968 NATO Conference on Software Engineering**)**, Van Nostrand Reinhold, 1976, pp. 88-98, 1976.

[5] M.H. Selamat, H. Sanatnama, A.A.A. Ghani and R. Atan, Software Component Models from a Technical perspective. IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.10, pp.-135-147, October 2007.

[6] G.T. Heineman and W.T. Councill, Component-Based Software Engineering: Putting the Pieces Together, Addison-Wesley, May 2001.

[7] R.K. Pandey, J.W. Shareef, Component-Based Software Development with Component Technologies: An Overview, International Journal of Computer Science and Information Technologies, Vol. 3 (1), 2012, 3029 – 3036, ISSN: 0975-9646.

[8] M. Huizing, Component Based Development, www. xootic.nl/magazine/jan-1999/**huizing**.pdf, last access:22.11.2011

[9] Jiyuan Shi, Software Reuse and Component Technology, Third International Symposium on Information Processing, October 2010, pp: 499-501.

[10] P. Vitharana, 'Risks and challenges of component based software development', Commun. ACM, 2003, 46, (8), pp. 67–72.

[11] J.Z. Gao, Tsao, H.-S.J., and Y. Wu, "Testing and quality assurance for component based software", Artech House, 2003.

[12] J.W. Shareef, Component-Based Software Development: An Appropriate and Methodical Approach, International Journal for Electro Computational World Knowledge Interface, Vol.1, Issue 5, Jan. 2012, ISSN No. 2249-541X.

[13] James, K.L., Software Engineering, PHI Learning Private Limited, M-97, Connaught Circus, New Delhi, pp: 78, 2009.

[14] S., Sedigh-Ali, A. Ghafoor, and R.A. Paul. Software Engineering Metrics for COTS-Based Systems, IEEE Computer, Vol.34, No.6, pp: 44-50, 2001.

[15] K.P. Kaur, Bedi, J. and H. Singh, Towards a suitable and systematic approach for component based software development. World Acad. Sci. Eng. Technol., 27: 190-193, 2007.

[16] Sommervilee, I., Software Engineering, 7th Edition, Pearson Education.

[17] I., Crnkovic and M., Larson, Building Reliable Component Based Software Systems, Artech House, Boston, 2002.

[18] Crnkovic, I. Component-based Software Engineering – New Challenges in Software Systems, Artech House, Boston.

[19] J. Voas, J. Payne. Dependability certification of software components. Journal of Systems and Software, 2000; 52: 165-172.

[20] J. Morris, G. Lee, K. Parker, G.A. Bundell, Peng Lam Chiou. Software Component Certification, IEEE, Computer, 2000, pp: 30-36.

[21] K. Wallnau, J. Stafford. Ensembles: Abstractions for a New Class of Design Problem, 27th Euromicro Conference 2001 Proceedings, IEEE Computer Society, 2001: 48-55.

[22] G. Kotonya, A. Rashid. A strategy for Managing Risks in Component-based Software Development, 27th Euromicro Conference 2001 Proceedings, IEEE Computer Society, 2001: 12-21.

[23] I. Crnkovic, M. Larsson, Filipe JK Küster, K. Lau. Databases and Information Systems, Fourth International Baltic Workshop, Baltic DB&IS Selected papers. Kluwer Academic Publishers, 2001: 237–252.