Overview
Component-based Software Engineering
State of the Art Report

Ivica Crnkovic[1], Magnus Larsson[2]

[1]Mälardalen University, Department of Computer Engineering, 721 23 Västerås, Sweden,
Ivica.Crnkovic@mdh.se

[2]ABB Automation Products AB, LAB, 721 59 Västerås, Sweden
Magnus.Larsson@mdh.se

Abstract: Component-based development has many potential advantages such as shorter time to market and lower prices. These advantages are especially attractive for customers, who often do not recognize the risks of lower reliability, possible problems with maintenance, etc. Many software companies are forced to use imported components in their products, but are not able to keep the development process under control. Component-based development is still a process with lot of problems, not well defined either from theoretical or practical points of view. The lack of knowledge is probably the biggest problem and the need for component-based software engineering (CBSE) is urgent. This was the motivation to a company and a university to organize a Ph.D. course on CBSE. Both Ph.D. students and practitioners from the industry participated in the course. The aim of the course was to increase the knowledge and understanding of CBSE, and to analyze the needs for software components in different engineering areas. The course consisted of lectures, seminars and student reports on chosen CBSE topics. This paper describes the course, gives an overview of the reports, and discusses the course result.

1 Introduction

There is a high trend of using components, especially COTS (commercial off the shelf) components, in software development. Both customers and producers share the enthusiasm for the CBSE approach because of the obvious advantages: The development time dramatically decreases, the usability of the products increases, the production costs usually decrease, and so on. Indeed, in many domains significant improvements in efficiency of development have been achieved. However, inclusion of components, over which the producer does not have complete control over, increases the risk of getting unexpected results. Even good components can corrupt a good product if they are managed in the wrong way. In some domains, such as industrial automation, this risk is unacceptable, and additional measures are required to minimize it. One of the problems is the lack of established procedures and, in general, a lack of knowledge of CBSE. To highlight the problems and to see which is the primary interest for industry and academia, Mälardalen University has, together with ABB, a leading industrial automaton company, arranged a Ph.D. course on CBSE. The motivation for the course is described in chapter 2. The course performance discussed in chapter 3. The state of the art report that was the outcome from the CBSE course is presented in chapter 4 with an overview of the contents. An analysis of the reports is given in chapter 5.

2 Importance of CBSE for the Industry

ABB, one of the leading companies in industrial process control, used to be a hardware company selling large electrical equipment and similar products. Now it is becoming software oriented. ABB is trying to focus on the core competence using, instead of developing itself, standard hardware and software components. There is also a strong demand from customers for products that use standard technologies and are open to integration with systems from different vendors. For example, a paper plant customer may want to have an operator station from ABB and controllers from Simens. To mix systems from different vendors were much more difficult when

the products were proprietary. Using COTS and standard components helps to open up the system.

It looks very promising to use components as reusable entities, but there are many traps for developers that can lead to higher cost than benefits. There are many problems for both developing components for reuse and with using reusable components. For example, ABB has made an architecture that is both developed with and for reuse [1]. The system was designed to be flexible, robust, stable and compatible. However, to achieve these properties a higher price had to be paid. Reuse principles placed high demands on the reusable components. The components had to be sufficiently general to cover the different aspects of their use. At the same time they had to be concrete and simple enough to serve a particular requirement in an efficient way. As a value of thumb, developing a reusable component requires three to four times more resources than developing a component for particular use [1]. There is also a problem with evolving of functional requirements. The development of reusable components would be easier if functional requirements did not evolve during the time of development. As a result of new requirements for the products, new requirements for the components will be defined. The more reusable a component is, the more demands are placed on it from products using that component.

The use COTS components and their integration into proprietary reusable components raises even more concerns because the proprietary components may loose their characteristics. The unexpected behavior of COTS, non-proper adaptation, and features that do not completely fulfill the requirements, may lead to a degradation of higher-level components [8].

There are also other issues that require different management than before: For example, dynamic and on-line configurations of component in the systems, dynamic architectures [6][7] and component configuration management [3] at run-time.

All these and similar issues are apparently becoming important for a successful development and marketing of industrial system. ABB is very much aware of this importance and that is the reason why it has started a common project with Mälardalen University. The main purpose of the project is to increase the knowledge and raise interests for CBSE in both industry and academia. The project includes different research and training activities, and one of them is giving a Ph.D course – CBSE.

3 The CBSE Ph.D. Course

The aim of this course was to collect and systematize knowledge related to CBSE. One goal was also to find the areas where CBSE is used or where there are interests to use CBSE. The final goal was to produce a “state of the art” report and to present the work in a seminar with attendees from both industry and academia.

During the course different issues from a development life cycle [5] were taken into consideration: Component development, component selection and adaptation, deployment and integration, system architecture issues, maintenance, and of course, use and management of components.

The course was structured in four parts: lectures, seminars, workshops and report writing [4].

During the lectures an introduction to CBSE and a framework for students self studies were given. The different problems with using existing components and component models like COM, CORBA, and EJB were presented. One lecture was focused on the risks of having components in safety critical and real-time systems. New configuration management problems that appear when dealing with components were also discussed. The lectures were problem-oriented, i.e. the existing problems were emphasized and possible directions to their solutions indicated.

After the lectures the students selected one of the proposed topics or one of their own interests. All students had a five-minute presentation, where they presented the topic and the particular issues that they planned to address in their reports, to make sure that the chosen topics were in the scope of the course.

Several seminars with invited speakers, the experts in this area, were organized during the course. In conjunction with these seminars workshops were organized to follow up the students' work. Between the workshops the students made investigations into the topics, and wrote the reports. The intention of the reports was primarily to describe the current status of CBSE and the current requirements on it, although many students were inclined to make a research reports. The papers were presented and discussed at a seminar, which also was open for everyone to get valuable comments from the industry. Finally, the papers were collected in a "CBSE - state of the art" report.

Practitioners from the industry were also invited to take the course and from thirty students who attended the course in total, eight were from the industry.

4 The CBSE Reports - State of the Art

In their reports, the students described different approaches, methods, technologies, problems and possible solutions related to CBSE. The students have chosen themselves their area of interests. The benefit of letting the students to chose topics of their free will is that they picked topics that were interesting to them which led to higher motivation and a better quality of the produced reports. More important, this approach emphasized the problems of using CBSE in different domains, and point to various demands on CBSE from different points of view. To minimize the risk of getting completely different reports, the lectures, seminars and workshops were used as a common background.

The reports are categorized according to component life-cycle phases (finding, selecting, creating, adapting, deploying and replacing components):

- Definitions and Specifications of Software Components
- Component Software Architecture
- Developing Software Components
- Pragmatics of Software Components
- Real-time Software Components

The reports classified in these categories are described below.

4.1 Definitions and Specifications of Software Components

The reports address the problems of non-complete component specifications and discuss the questions if formal specifications can completely describe the components. The following reports fall in this category:

- *On the definition of concepts in CBSE*
- *Semantic integrity in CBD*

4.1.1 On the definition of concepts in CBSE

This report gives an overview of commonly used terms within the area of CBSE. The concepts “components”, “objects”, “contracts”, “interfaces”, “patterns” and “frameworks” and their relations are discussed. In the paper it is found that there seems to be some variations as to what the different terms really mean in different situations and by different authors. However, a general line of definitions has been extracted in which there is, for instance, a clear distinction between a component and an object. The relation between patterns and frameworks is mentioned. Frameworks as defined in UML and Catalysis [9] are further described from the formal aspect, and finally a similarity between frameworks and contracts is discussed.

4.1.2 Semantic integrity in component based development

The purpose of the paper is to investigate how the semantic aspects are described when defining components and to see if pre- and post-conditions or similar traditional methods are used and how such methods can be applicable in a component-based environment. The paper introduces the problem with describing and handling methods and modules in a non component-based environment. After that, a survey on the current state in semantic integrity with regards to components follows. Finally, the paper discusses a possibility to bring in aspects of the traditional methods into CBD and how this would affect the quality and robustness of the components.

4.2 Component Software Architecture

CBSE and software architecture (SA) [10] are separate but related topics in software engineering research and practice. CBSE focuses on the realization of systems through integration of pre-existing components, while SA is concerned with the high-level organization and structure of systems in general. The current high interest in SA is mainly motivated by the possibility of managing complex software by using components. The questions discussed in the papers bring together SA and CBSE methods. The following papers belong to this category:

- *Architectural styles in component based software engineering*
- *Separation of concerns in software components*
- *Towards a component framework for complex mechatronics*

4.2.1 Architectural styles in component based software engineering

One of the important issues studied in SA is recurring architectural patterns and idioms or architectural styles. CBSE and SA are clearly related, and the importance of architectural issues in CBSE is now widely recognized. However, since their motivations differ (and therefore their criteria for what should and should not be viewed as a component), the relationship between the two topics may not be straightforward. This report investigates this relationship with particular focus on architectural styles and component-based technologies such as DCOM and JavaBeans. The following questions are discussed: What is the relationship between CBSE-components and SA-components? Which architectural styles are best suited for CBSE? Which architectural styles do current component technologies support?

4.2.2 Separation of concerns in software components

Separation of concerns was always at the core of modern software engineering. After presenting an overview of separation of concerns and of the latest developments in the mechanisms supporting it, this report presents the problems related to supporting concerns within

components, and crosscutting components. This study focus on how various separation of concerns approaches can respect and adapt to coarse-grain components and their supporting technologies.

4.2.3 Towards a component framework for complex mechatronics

A mechatronics system embodies technologies from several engineering disciplines in the domains of mechanics, automatic control, software and computer hardware. The major objective of the report is to explore and define a component framework for such systems, e.g., robotics. Firstly, the main characteristics of the system are described, and then a principle for identifying various components (e.g. concept component, control component, SW component, HW component, mechanical component) are discussed. Finally, the interdependencies between these components are examined in order to find out how the use one type of component is constrained by others.

4.3 Developing Software Components

The following reports are focused on the development phase of CBSE:

- *The need for more mature life-cycle models in CBD*
- *Applying CBSE theory on corporate resource*
- *A parallel development approach to CBSE*
- *Role based component engineering*
- *Designing components for variability*
- *Building flexible components based on design patterns*

4.3.1 The need for more mature life-cycle models in CBD

The paper discusses differences between CBD and traditional software development and argues for the need of a more mature life-cycle models in CBD. The different approaches to development such as sequential, iterative, incremental and evolutionary are analyzed from the CBD's point of view.

4.3.2 A parallel development approach to CBSE

The paper describes a model of parallel developing components. It is focused on a Configuration Management model that allows parallel development of different functions of a component, error corrections in parallel with new development and even parallel development of entire component families. The authors were encouraged by the success of free software projects and mention the development of Netscape as one example of projects that practice parallel development to keep track of distributed development and bug fixes of various components.

4.3.3 Role based component engineering

The paper discuses a role oriented object/component design, in particular, how one can use roles to glue components together, the importance of roles in designing frameworks and how one can map the role paradigm onto existing programming languages**.** The idea of role based components is that the public interface is split into smaller interfaces that model the different roles a component can take in a system. Users of a component can communicate with the component through the smaller role interfaces instead of using the full interface. The main advantage of this is that by limiting the communication between two components by providing a

smaller interface, the communication becomes more general and it becomes easier to plug in components in the system. This and other aspects of role oriented design are discussed in the paper. Role based modeling is discussed in Reenskaug's work about objects [11].

4.3.4 Designing components for variability

The paper elicits the different aims and requirements regarding the design of component for single-products, product lines, and COTS markets. This is complemented with a description of the different variation aspects, i.e. product variability, deployment variability, and domain evolution. With this as a basis, the authors discuss what the particular problems for each type of variability is, and how these challenges can be met, either by means of management, or by technical solutions such as configuration management, parameterization, or design pattern.

4.3.5 Building flexible components based on design patterns

In the past ten years design patterns have played an important role in software reuse. Design patterns are verified and general solutions to common reoccurring problems. Some of the interesting questions are: What benefits might design patterns bring to CBSE? What is the relationship between design patterns and software components? Is there a set of design patterns defining software components? These questions are elaborated in the paper.

4.4 Pragmatics of Software Components

The papers in this category point to utilization of CBSE, not only methods and technologies but also ideas. The following papers are selected into this category:

- *Applying CBSE theory on corporate resource*
- *Variations in component implementations*
- *Making it possible to use smaller components*
- *Outsourcing, COTS for the new millennium?*
- *Towards a visual working environment*

4.4.1 Applying CBSE theory on corporate resource

This report is an experience report, which discuss some of the problems raised during work with different CBSE technologies. It also describes the project with a goal to create company component-library. The goal was to gather, maintain and develop a C++ class library that would contain different functionality needed by a software developer developing COM-components. It was soon realized that reuse of binary components was not enough. There is also a need to incorporate even knowledge, and other parts, which do not directly belong to executable part of the components, for example test programs and examples implemented in different programming languages, different kind of documentation, etc. The paper describes different examples of reusing components. Finally the idea of the component paradigm is applied on management any kid of resources and aspects, even human resources.

4.4.2 Variations in component implementations

The paper is a report on an exploratory empirical study of multiple independent implementations of software components. The components are developed for reuse by independent teams or individuals following an informal specification of a potential component and possibly a informal description of application domain. Metrics, technical aspects, solution aspects and domain

aspects are considered. The hypothesis this study is based on is that component implementations do vary and the goal is to learn about the nature of the variation, and its reasons.

4.4.3 Making it possible to use smaller components

In the industry, large components have been used for several years. Large components are often easy to find because they are well known. The complex nature of large components makes it obvious that it is easier to reuse them than to develop your own version. Small components on the other hand are difficult to find because the code-per-spec ratio is much lower making them less tempting to use. The problem of using and finding small components are more of administrative than of technological nature. This paper studies some Open Source Software (OSS) projects, which are being developed over the Internet. Two different development models are examined, the Cathedral and the Bazaar [12], and their impact on the use of components is analyzed. The goal of this report is to find some common ground of success factors from the OSS development projects that can be adopted for use in corporate environments.

4.4.4 Outsourcing, COTS for the new millennium?

One of basic problem in CBSE is insufficient knowledge about the component used in the system development process, and uncertainty of the component's future. Outsourcing is a development outside the development organization, but what it differs from COTS is that the development still under control of the component user. Having this control, a lot of problems related to COTS can be avoided. There are, however, disadvantages in outsourcing comparing to using COTS: Time to market is probably slower, the costs can be higher, etc. This paper describes these characteristics and compare them COTS-based development.

4.4.5 Towards a visual working environment

The paper describes a possibility to utilize a "component language" which will automatically generate the glue code between components. The possibility of the glue code generation is discussed by relating problems to Artificial Intelligence methods. The components can be composed in a frame of a visual working environment where the user does not care of how operations are done, but just moves the components to each other to perform different operations. In such an environment the system can warn the user is the operation the user is intended to do in not possible. The paper describes a simulation of such an environment in a specific domain.

4.5 Real-time Software Components

This category includes papers that discuss use of CBSE in domains with higher requirements on reliability, timing characteristic, etc., i.e. where non-functional requirements play crucial role. The specification of non-functional characteristics is a weak side of the specification of the components. Is it at all possible to use components, in particular COTS, for such systems? It is possible to prove a correctness of imported components? These questions are discussed in the following papers:

- *Requirements for real-time components*
- *Formal and Probabilistic Arguments for Component Reuse in Safety-Critical Systems*
- *Components in Intelligent Robots*

4.5.1 Requirements for real-time components

The purpose of this report is to identify specific requirements needed for Real-Time Components. The report includes the following topics:

- Instantiation of real-time components (both functional and temporal)
- Effective reuse of general components (is it possible to reuse only a portion of a real-time component, in order to reduce memory and computing requirements?)
- Homogeneous in relation to heterogeneous component interfaces.
- Verification and testing (real-time systems are often safety-critical. How does this influence verification and testing?)

4.5.2 Formal and Probabilistic Arguments for Component Reuse in Safety-Critical Systems

In this paper presents a framework for specifying and relating component contracts for components used in safety-critical real-time systems. Using both quantitative and qualitative descriptions of component attributes and assumptions about the environment, we can relate input-output domains, temporal characteristics, fault hypotheses, integrity levels, and task models. Using these quantitative and qualitative attributes we can deem if a component can be reused or not, how much and what subsets, of say input-output domains, that need additional functional and safety verification. This framework will give formal and quantitative arguments for reuse of components in safety-critical real-time systems.

4.5.3 Components in Intelligent Robots

The building of a complete mobile robot system, requires expertise in a number of different disciplines such as automatic control, computer science, sensor knowledge, mechatronics, artificial intelligence, etc. Most robot researchers are specialists in one of these areas. Nevertheless, a complete system is needed to prove any work made in a special field. It is the belief that the introduction of components is a necessary step to move the robot technology from research labs to commercial business applications. This paper discusses ongoing work in this area, and proposes further solutions.

5 The Analysis of the Reports

The different backgrounds and areas of interests of students have led to different subjects, different approaches, focuses, and styles of the reports. As a consequence of this, and because the students have chosen the topics themselves, there is a large variety in the reports. In that sense the “CBSE – state of the art” report is more like a proceedings than a homogeneous book. On the other hand the reports show that there exist many areas where CBSE can be successfully applied, and at least some of the approaches and ideas implemented. Many reports discuss the component development and identification issues: How a piece of software can be recognized as a reusable component, how to design a component, how to describe a component, and so on. Another group of reports discusses the usability of components in specific domains. Finally, there are reports which focus on the practical issues – how to easy get information about components and about all aspects around it, and how to utilize the CBSE methods in the software and system development. In general, we can say that the course has fulfilled the purpose. The lectures, seminars and reports with a combination with several shortened seminars for the industry increased the knowledge of students, developers and even software managers.

6 Conclusion

Running this course for graduate students and practitioners has been very valuable and interesting. Many of the written papers have good quality and we have succeeded to highlight some of the areas of CBSE. Some of the papers have been or will be sent to different international conferences. However, the goal of the course was not to create research papers but to give a state of the art and a state of the practice of CBSE. A benefit of building up a "state of the art" material is a good base to continue to perform the seminars for the industry people and spread information about CBSE. Also, new courses of the same type will be organized in the future. We see that there is a strong demand for information in the CBSE area. CBD is a hot topic and the industry must adopt the technology quickly to be able to be competitive. The course has been valuable for the students, especially for those with different research area, to understand the problems and to get overview of interesting research topics. There was also a valuable experience to meet and to listen to the invited speakers, experts in the software engineering. A good forum for questions about component-based development has been created. Finally, creating this course was very good opportunity for lecturers to gather a lot of information about particular topics and share that information by all the members of the course.

7 References

[1] Larsson M., Crnkovic I., "Development Experiences of a Component-based System", *Proceedings of Engineering of Computer Based Systems (ECBS 2000)*, IEEE, 2000

[2] Szyperski C., *Component Software*, Addison Wesely, 1999

[3] Larsson M., Crnkovic I., *New Challenges for Configuration Management*, System Configuration Management Symposium, 1999, proceedings, Springer

[4] Ivica Crnkovic, Magnus Larsson, Kung-Kiu Lau, The CBSE course, http://www.idt.mdh.se/kurser/CBSE

[5] Wallnau K., 2d international workshop on CBSE, Los Angeles, 1999

[6] Goedicke M., Meyer T., *Dynamic semantics negotiation in distributed and evolving CORBA systems: towards semantic-directed system configuration*, Proceedings of Configurable Distributed Systems, IEEE, 1998

[7] Feiler P., Li J., *Managing inconsistency in reconfigurable systems*, IEEE Software Vol. 145 Issue 5, 1998

[8] Josefsson M., Oskarsson Ö., "Programvaru-komponenter i praktiken – att köpa tid och prestera mer", Report from Sveriges Verkstadsindustrier 1999, In swedish

[9] D.F. D'Souza and A.C. Wills: *Objects*, Components, and Frameworks with UML: The Catalysis Approach, Addison-Wesley, 1998

[10] Len Bass, Paul Clements, and Rick Kazman, Software Architecture in Practice. Addison-Wesley, 1998

[11] Reenskaug T., *Working with Objects*, Manning publications, 1996

[12] Eric S. Raymond, *The Cathedral & the Bazaar*, 1st Edition, O'Reilly, ISBN 1-56592-724-9, October 1999