Component-based Software Engineering: Building Systems form Software Components

Ivica Crnkovic
Mälardalen University, Department of Computer Engineering
PO Box 883, SE-72 123 Västerås, Sweden
ivica.crnkovic@mdh.se

1. Introduction

Component-based Software Engineering (CBSE) is concerned with the development of systems from software components, the development of components, and system maintenance and improvement by means of component replacement or customization [1].

Building systems from components and building components for different systems requires established methodologies and processes not only in relation to development/maintenance phases, but also to the entire component and system lifecycle including organizational, marketing, legal, and other aspects. In addition to objectives such as component specification, composition, and component technology development that are specific to CBSE, there are a number of software engineering disciplines and processes that require methodologies be specialized for application in component-based development. Many of these methodologies are not yet established in practice, some have not yet been developed. Experiences from other areas, such as system engineering can be successfully applied on component-based development, as there are many similarities in the basic concepts (for example, relations between systems and components). Also, with its focus on components and their specifications, CBSE can give better understanding of building systems in general, and in particular of computer-based systems whose significant part is software.

The progress of software and system development in the near future will depend very much on the successful establishment of CBSE; this is recognized by both industry and academia. The growing interest in CBSE is reflected in the number of workshops and conferences with CBSE tracks [2-6].

2. Building systems and building components

CBSE addresses challenges and problems similar to those encountered elsewhere in software engineering. There is however one difference; CBSE specifically focuses on questions related to components and in that sense it distinguishes the process of "component development" from that of "system development with components". There is a difference in requirements and business ideas in these two cases and different approaches are necessary. Components are built to be used and reused in many applications, some possibly not yet existing, in some possibly unforeseen way. Marketing factors play an important role, as development costs must be recovered from future earnings, this being especially true for COTS. System development with components is focused on the identification of reusable entities and relations between them, beginning from the system requirements and from the availability of components already existing [7].

3. Building systems – a top-down approach

A standard approach in a system development is a top-down approach; the system design starts with specification of system architecture. The architecture defines the components of a software system as well as their interactions and can be used to analyze its quality attributes.

The system design process typically consists of three phases, which might be passed in several iterations [8]. The first phase includes functionality-based design (i.e. a design of the software architecture based on the functional requirements). Although software designers generally will not design a system without concern non-functional requirements, these are not explicitly addressed at this stage. Functionality-based design consists of four steps: defining the boundaries and context of the system, identification of archetypes, decomposition of the system into its main components and, finally, the first validation of the architecture by describing a number of system instances). The second phase is the assessment of the quality attributes of software architecture in relation to the quality requirements. The third phase of the software architecture design process is concerned with transformation of design solutions to improve the quality attributes while preserving the domain functionality captured by the software architecture. These transformations result in a new version of the software architecture which in general improve one or some quality attributes while they affect others negatively.

The final result of this stage is a system software architecture which identifies components and interactions between them. Up to now the design model is not specific for component-based approach. In a “classical” approach the next step would be to implement the components identified by the design. In a component-based approach the main idea is re-use already existing components, i.e. to find the most suitable components. The implementation effort in system development will decrease but the effort required in dealing with components; locating them, selecting those most appropriate, testing them, etc. will increase.

4. Building systems from components

The types of components that a system is composed of influence the architecture of the system. In a similar way as the framework into which components are to be plugged influences architecture of the system and, the type of components selected, influences the system design process.

Design freedom is limited to component selection and the way the selected components are integrated. This restricted freedom points out the importance of managing and controlling component integration. Component specifications in from of API do not normally provide enough information about how the component will behave when used in a given environment. This difficulty raises issues related to understanding and verifying both the functional and non-functional properties of the components so that the unexpected mismatches among components are avoided and overall system behavior can be accurately predicted. The verification of component properties is required in order for developers to have confidence that a system will behave as its architect predicted it would. In many cases a component property alone cannot be used to predict the system behavior, but clusters of components (assemblies) must be analyzed separately.

The examples briefly mention above show that much of activity in the design and development phase belong to the component properties and composition issues. This shows that a component-based design can have difficulties in using top-down approach. Rather a mix of a top-down and a bottom-up approach will occur.

5. Challenges of CBSE

A successes of component-based approach is still not guaranteed and it depends which solutions will be found on many, still open questions. Some of them are listed here:

- Trusted components and components certification [9].
- Composition specifications and predictability [10].
- Requirements management and component selection [11].
- Long-term management of component-based systems.
- Component configurations [12].

The question is how these challenges can be met? Solutions can be related to a question how much experience from system engineering and computer-based system engineering can be utilized in CBSE? Can we use the same or similar methods? To which extend we can relate system components to software components?

6. References

[1] Crnkovic I., Larsson M., *Building Component-based Reliable Software Systems,* Artech House, 2002

[2] Bachman, et. Al., Technical Concepts of Component-Based Software Engineering, report CMU/SEI-2000-TR-008, Software Engineering Institute, Carnegie Mellon University, 2000.

[3] 4th and 5th ICSE Workshops on CBSE: Component Certification and System Prediction, Benchmarks for Predictable Assembly, http://www.sei.cmu.edu/pacc

[4] 27th and 28th Euromicro Conferences: CBSE track, http://www.idt.mdh.se/ecbse

[5] First International Working Conference on Component, http://swt.cs.tu-berlin.de/cd02/

[6] Crnkovic I., Larsson S., Stafford J., Component-based SE workshop, Building Systems form Software Components, Engineering Computer-based Systems Conference, Lund, Sweden, April 2002

[7] Bass L., Clements P., and Kazman R., *Software Architecture in Practice*, Addison-Wesley, 1998

[8] Bosch J., *Design & Use of Software Architectures*, Addison-Wesley, 2000.

[9] Morris J., Lee G., Parker K., Bundell G., Peng Lam C., "Software Component "Certification", *IEEE Computer, 2001*, September

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

[11] Kotonya G. and Rashid A., A strategy for Managing Risks in Component-based Software Development, *27th Euromicro Conference 2001 Proceedings*, IEEE Computer society, 2001, pp. 12-21

[12] Crnkovic I., Larsson M., Küster Filipe J. K., Lau K., *Databases and Information Systems, Fourth International Baltic Workshop, Baltic DB&IS*, Selected papers, Kluwer Academic Publishers 2001, pp.237-252