Position paper:

Managing Standard Components in Large Software Systems

Ivica Crnkovic
Mälardalen University, Department of Computer Engineering
Box 883, SE-721 23 Västerås, Sweden
ivica.crnkovic@mdh.se
+46 21 103183

Magnus Larsson
ABB Automation Products AB
SE-721 67 Västerås, Sweden
magnus.ph.larsson@seapr.mail.abb.com
+46 21 342666

Abstract

This position paper consists of two parts. The first part gives an overview of a research project started by ABB and Mälardalen University. The project is concentrated on use of standards technologies and standard components in real-time industry-process systems. The main goal of the project is to increase the knowledge about software development based on standard components from both theoretical and practical points of view. The project can be an interesting case of practices of adopting CBSE, a case of an approach in managing component-based engineering.

The second part of the paper points to some important aspects in use of components at the development, run-time, and maintenance phases. It is a problem of identification and configuration of components in software systems. Configuration Management (CM) disciplines, such as Version Management, Configuration and Build Management, Change Management, etc., are well established for the conventional development. In a component-based development some new requirements on identification and version management arise, and some new methods, similar to those from conventional CM, and possible extensions of the existing methods, should be objects of further research. We propose an extension of CBSE-handbook with a new sub-chapter which is related to component identification and configuration and which could be a part of the Technology for supporting CBSE chapter.

1. A Case study - Use of Standard Technology in Industrial Applications

ABB is a global $30-billion engineering and technology company serving customers in electrical power generation, transmission and distribution, in oil, gas and petrochemicals and, in general, in industrial automation products. ABB employs 200 000 people in over 100 countries.

ABB Automation Products, a $340-million company, is responsible for developing automation products inside ABB and employs 2000 people. The automation products encompass several families of industrial process-control systems including both software and hardware.

The main characteristics of the products are reliability, high quality and compatibility. These features are results of responses to the main customers requirements: The customers need stable

products, running round the clock year after year, which can be easily upgraded without impacts to the existing process. In the recent years the requirements have been, however, somewhat changed. Customers require integration with standard technologies and use of standard applications in the products. This is a high trend on the market but low awareness about the possible problems exists. A improper use of standard component can cause severe problems, especially in the distributed real-time and safety-critical systems, with long-period guarantees. In addition to these new requirements, time-to-market demands become very important factor.

These factors and other changes in software and hardware technology [AOY-98] have introduced a new paradigm in the development process: In the middle of the eighties, ABB control products were complete proprietary monolithic systems with internally developed hardware, basic and application software. In the beginning of the nineties, standard hardware components and software platforms were bought while the real-time additions and application software were developed internally. Now the development process is focused on the use of standard and de-facto standard components, outsourcing, COTS and producing components. At the same time, the final products are not any longer the closed monolith systems, but are instead component-based products that can be integrated with other products available on the market.

This new paradigm in the development process and the marketing strategy has put new problems and questions in the focus:

- The development process has been changed. The developers are not only designers and programmers, they are integrators and marketing investigators. Are the new development methods established, are the developers properly educated?
- What are the criteria for selecting of a component? How can we guarantee that a standard component fulfills the product requirements?
- What are the maintenance aspects? Who is responsible for the maintenance? What are expectations for updating and upgrading of components? How can we manage the compatibility and reliability requirements?
- What is the trend on the market? What can we expect to buy not only today but also what will be present the day we start to deliver our product?
- When developing a component, how can we guarantee that the "proper" standard is used? Which standard will still be valid in five, ten years?

In order to find some answers and to give a theoretical base to the new methods, ABB Automation Products together with Mälardalen University have started a project for research of use of Standard Technology in Industrial Applications (STINA). The main goal of the STINA project is to increase the knowledge of software development based on standard components from both theoretical and practical points of view. The research results will be used both at the university and in the industry. At the university, the accumulated knowledge will be used for further education in order to prepare the students for new aspects in system development. The industry will benefit with direct implementation of methods and knowledge built up in the research activities and well educated students.

STINA is staffed with people both from ABB and the Mälardalen University. In its first phase the research activities are being defined for a three-year period. During that period, or later, some other ABB companies will actively participate in the project. The project is in the initiation

phase, i.e. the activities, milestones and the project results are being planned.

The possible subjects of the research are:

- Technologies for using and developing components. E.g.COM/DCOM, Java Beams, CORBA, Web, Windows 2000 and Linux;
- On-line update;
- Development, run and compose-time configuration management;
- Use of standard components for real-time applications/system;
- Quality assurance and maintainability aspects;
- Reusability of components in both real-time and non real-time systems.

The work will result in several "State of the art" reports, courses, research papers, prototypes and finally Ph.D. thesis.

Our expectations of CBSE are to find standards, disciplines and guidelines so we can investigate them and apply them on the real-time systems of the industry. One objective is also to feed back comments and experience from the industry to the academic world.

2. Standard Components and Configuration Management

In the conventional development/maintenance process Configuration Management (CM) plays an important role. The main purpose of the CM is:

- Identify and manage different versions of source code in a multi-user environment (Version Management and Work Space Management);
- Configure the components and build them (Configuration and Build Management);
- Keep control of changes at a logical level (Change Management).

The software systems based on standard components are results of a combination of pure development and integration of components. The requirements on conventional use of CM remain and new requirements related to component management appear in all phases - in the design, integration and run-time. Especially the integration part becomes important.

We can expect that the source code management becomes less critical, because we expect less internal development. The integration part, i.e. configuration, and version management of the components becomes essential. Change management keeps the same role, but the implementation of the process is different.

The importance of CM, and challenges in research and implementation of CM support, are emphasized of the 1998 CBSE workshop [BRO-98], as quoted: "In particular, high composeability in a product line setting amounts to mass customization and this introduces tremendous configuration management challenges and support challenges."

2.1 Version Management and Configuration Management

In the conventional development, we recognize two different phases - development and run-time.

In the development phase, we design and build configurations that will be used in the run-time. Those parts (typically source code and documentation), which are under intensive change process, are placed under version control. Each item version is identified by a name, version number and different attributes. There is information about who has made a change, when, and often why. A change performed on an item causes a generation of a new item version. In the integration phase, the particular versions of items are selected, defined as elements in a baseline, and used for the system building. In the building process, the new objects, called derived objects, are created from items under version control and from items that make the complete environment (for example, different system libraries, tools, operating systems, etc.). In simpler tools, such as Make, the identification of items outside version control is neglected. In more sophisticated tools, such as ClearMake [LEB-94], we have a possibility to identify the items used in the build process which are outside the version control. This control is however not so precise as for versioned items. It is not supposed that they are changed very often.

In the component-based development we are facing a new situation: The dominant or, at least, the significant parts of building items are components. They can be identified by their names, or by some internal representation, but usually there are difficulties with their version identification. In some cases of components, for example ActiveX, there are possibilities to define a version property, but the management of these properties are limited and not standardized.

Components can easily be replaced and the replacement can be made in an uncontrolled way if not performed cautiously. One common situation is when a new component version automatically replaces another component, which itself is used by some other parts of the system. We can be in a situation where different parts of a software system use different versions of the same component, which can lead to unpredictable system behavior (Figure 1). Even more, a component can be replaced directly in the run-time environment.

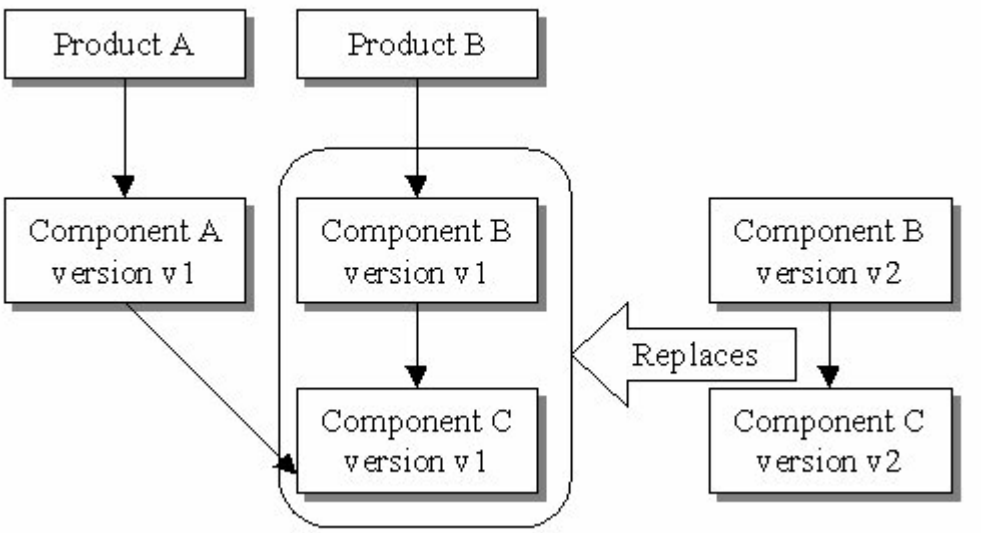

Figure 1. The new version of component B adds version 2 of component A into the system

To minimize the risk to fall into unpredictable situations, we need CM, not only for the parts internally developed, but also for the external components. A minimal requirement is to have a

uniform version identification of the components. The version identification can be applied on components placed in a repository. Such a repository would also include information about other features of components [NAD-98]. The component identification required in the design phase is not sufficient. We also need a mechanism for components identification in the integration (composition) phase, and finally in run-time phase. As different versions of a component may be included in the system, it should be possible to generate a component dependency graph and a non-consistent use of components should be indicated.

2.2 Change Management

The change management process is focused on the logical change introduced in the system and its relation to the physical changes. One purpose of using change management is to simplify the integration process and ensure a logical consistency of changes (i.e. ensuring that all parts being embraced by a change are involved in the integration process). Another purpose is to provide information about the changes introduced in the system - information used by management, quality people, etc. In a component-based development the use of first part will be simplified - it is not expected to be so many items (files) as in conventional development. A change related to a component is in practice a replacement of a component or a component version. The second part becomes more complex. Change Management must consider the questions, such as: What are the reasons for a change, and what are possible consequences of the change?

A component might be updated just because another product uses the new component or it can be required by a new component introduced in the system. A new component version might be added to introduce new functions in the system, or only to change its behavior, (better performance, better stability), but keeping the same interface. When replacing a component or a component version we must consider which type of change is allowed, and which type of compatibility are required.

There are different levels in the compatibility:

- Input and Output compatibility. A component requires input in a specific format and produces result in a defined format. What are the internal characteristics of the component is not of interest. An example of such type of compatibility we can find in different word-processors producing the same document format.
- Interface compatibility (at development time and at run time). The interface remains the same, and the implementation can be different. Typical examples is different implementations of ActiveX objects, with the same interface.
- Behavior compatibility. Internal characteristics of the components, such as the performance, requirements on resources, etc., must be preserved. Such requirements can be appropriate for real-time systems.

The compatibility criteria can be used in the decision process if a component can be replaced or not. This decisions can be especially important by a replacement "on the fly" in a run-time environment. It is important to keep the requited level of the compatibility to avoid a possibility to interrupt the whole system.

3. Conclusion

Configuration Management becomes a more significant part in CBSE. The development is radically reduced, but more efforts on integration, thus CM activities, is required. The CM methods used in the conventional development can be taken as a starting point, but new methods have to be investigated and introduced in CM to efficiently use components. For this reason we propose a new item in the Handbook of CBSE, in chapter 3, Technology for supporting CBSE - Development support, or under Maintenance and Reengineering support.

We expect that the STINA project may contribute to such a work.

4. References

[AOY-98] M. Aoyama: New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development, 1998 International Workshop on CBSE

[BRO-98] Alan W. Brown, Kurt C. Wallnau: An Examination of the Current State of CBSE: A Report on the ICSE Workshop on Component-Based Software Engineering, 1998 International Workshop on CBSE

[LEB-94] David B. Leblang, The CM Challenge: Configuration Management that Works, Configuration Management, edited by Walter F. Tichy, Johnn Wiley & Sons, ISBN 0 471 94245-6

[NAD-98] Nader Nada, David C. Rine, A Validated Software Reuse Reference Model Supporting Component-Based Management, 1998 International Workshop on CBSE