Research Planning Course – Assignment 2

Séverine Sentilles

Mälardalen University
Department of Computer Science and Electronics
Box 883, SE-721 23
Västerås, Sweden

severine.sentilles@mdh.se

1. Project overview

1.1. Context

During the 90s, the Object-Oriented technology has shown its limits in its ability to produce both reliable and low-cost software. The Component-Based Software Engineering (CBSE) is a new paradigm which has emerged to overcome these drawbacks. Thus, CBSE is a sub-field of software engineering, which first lean on the object-oriented well-tried features and second focuses more particularly on the development of independent pieces of software, called components. These latter can be connected together to produce a final output, which can be a bigger component offering more functionalities, a system or an entire application. In other words, component-based software engineering is a discipline which aims at the production of software, in a broader sense, by the use of engineering principles. Consequently, in CBSE, a particular accent is put on the notion of reusability for the same reason that industry adopted this concept years ago. Indeed, reusability can allow to reduce the production costs while increasing the development speed of new products. Moreover, it also implies the utilisation of a "same" component in different software. It can consequently be tested a number of times in several contexts. This may lead to improve the quality of the developed products since they can be built out of "certified" sub-components.

However, at contrary to industry, where a component can be understood and reuse quite easily, this concept is more difficult to capture in CBSE and even though this is a primordial notion, there is no general agreement about what a component is really. Several definitions coexist focusing all on different aspects. A definition has nevertheless emerged as the most commonly acknowledge one. It states that "*A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party*" and has been established by Szyperski [1]. This introduces some of the other concepts particularly used in CBSE, that is to say interfaces, contracts, independent deployment, composition and a so-called third-party (simply meaning that the user of a component is not necessarily its developer). Those

notions will not be developed any further in this document since CBSE is not the main subject here, but more information about these different concepts can be found in [2].

The component-based technology is particularly appreciated to develop information systems or service-oriented systems. However, another type of systems exists which is called real-time system and used in numerous domains, such as vehicular, automation, nuclear and so on. In fact, those systems are exploited in domains where the respect of timing-requirements are a condition "sine-qua-none" for maintaining the safety of the physical device which relies on it. In other words, real-time systems must react correctly to events in an appropriate amount of time (neither too fast nor too slow) and in well-specified ways. This means that a real-time system must be predictable. Indeed, any infraction to one of these timing requirements can lead to a catastrophe, such as for instance the crash of an aircraft or the explosion of a nuclear plant. In such systems, timing requirements can be expressed as deadlines, execution times or periods among other things. When missing a deadline for example is totally forbidden by the specification of the system, then the system is called a hard-real time system. On the other hand, some miss of deadline can be tolerated in some systems called soft real-time systems.

Moreover, real-time systems can also be embedded, as it is the case for example in the automotive field where an airbag is an example of such embedded real time systems. This leads to take into consideration new requirements due to the limited resources existing in embedded systems. However, the component-based approach has not been used a lot so far to produce embedded real-time systems. This lack of utilisation comes mainly from the fact that the well-known component models (Enterprise JavaBeans, COM/DCOM, .NET, etc) and their supporting technologies, (respectively J2EE and Microsoft .Net framework for the two latter) consume too much memory and do not take into consideration any timing requirements.

Therefore, there is a need to have a component model dedicated to embedded real-time systems allowing thus to bring the advantages of CBSE, such as reusability, to the embedded system world. Some propositions have already been elaborated and are briefly presented in Section 2. However, they first only focus on the constraints inherent in the targeted utilisation domain and second are most of the time only use "in-house" (i.e. within the enterprise which has developed this component model). This is the case, for instance, for AUTOSAR which is currently under development and aims exclusively at the vehicular domain or for the Koala component model used in all the Philips equipment. But up to now, there is no common structure to build embedded real-time software no matter what the targeted domain is.

In the same time, the complexity of those embedded systems augments more and more and since those systems are, most of time, safety-critical, there is also a need to produce software with a certain degree of confidence in their ability to face the constraints and requirements of the targeted domain and to act without

any fault. The PROGRESS project, which will be explained in the following section, is an attempt to provide a solution to those problems.

1.2. The PROGRESS project

PROGRESS, which is both a project and research centre for the development of predictable embedded software, was launched on the 1st January 2006 within the Mälardalen Real-Time Research Centre and is supported by the Swedish Foundation for Strategic Research. Its vision is to establish the component-based development of embedded software as a mature engineering discipline which should allow to manage the increasing complexity of embedded software, to strengthen their quality and reduce their overall production costs. That is to say, PROGRESS aims at the provision of all the theories, methods, and tools needed to achieve those objectives with respect to industrial criteria. The integration of those techniques and results should be done either within a model-base framework, which constitutes the core approach of the PROGRESS project, or a set of demonstrators to show the obtained benefits. Indeed, most of the research results is envisioned to be applicable in a real industrial context.

1.3. My research project

It is within this context that my research project takes place and more precisely, within the Component-Based Development (CBD) cluster, which is one of the research groups of the PROGRESS centre. The objective of this cluster is to increase the research excellence in the area of component-based software modelling and design for embedded real-time systems. As a consequence of the situation described above, the following goals can be identified in my research project:

1. To specify a model-based framework suitable for a predictable development of predictable component-based embedded software. This model-based framework is a generic component model, or metamodel, which should be “generic enough” to allow to enumerate all the specificities, constraints and requirements, inherent in a first time, in the vehicular, automation and telecommunication domains, before eventually be enhanced to match other embedded domains. This model-based framework will have the ability to instantiate a particular component model according to the characteristics of a class of embedded system domains and second will provide means for a predictable development of those systems. That is to say, this model should make possible to use different analysis and composition theories of functional and non-functional properties and provide means for static and dynamic specification of components and component-based systems.

In order to see whether it is possible to manage to specify a generic component model gathering all the different requirements needed by several embedded real-time domains (the vehicular, automation and telecommunication domains), I envisage to adopt the following method:

- read the literature about the existing component models for embedded systems
- Try to find similarities/differences features, requirements, constraints…
- Try to gather those observations into a model

2. Moreover, the results obtained within the previous step as well as the results provided by the other PROGRESS research groups, have to be presented with respect to some industrial settings. Consequently, there is a need for an integrated development environment (IDE), which is a software gathering all the tools and techniques needed during the development process of an embedded systems. At contrary to most of the existing IDE that focus mainly on the programming aspect, this one should highlight the verification steps. My secondary objective is thus to provide the specification of this toolset which should also allow to automatically generate an instance of a concrete component model corresponding to a specific domain with respect to all the requirements and properties specified in the generic component model. Finally, as development environments face in general perpetual changes, such as the apparition of new tools for instance, this IDE should also be endowed with means to facilitate the addition of new features.

In order to obtain an acceptable specification of this IDE, I envisage first to study the most largely used IDE at present time that is Eclipse to identify its key features and second to identify the missing elements, which should be added to create an environment allowing a predictable development process of embedded systems.

Finally, to validate all these expected results, I will rely on a validation by example approach. That is to say, to present first some use-cases to exemplify for instance how to get a concrete application from the generic model. Then, concerning the Integrated Development Environment, the tool could by used by an enterprise to demonstrate its capacities.

2. Related work

This section presents some works related to the above context. It contains a brief description of UML, of some of the existing component models for embedded systems, and of Eclipse.

2.1. UML

The Unified Modelling Language (UML) [14] specified by the Object Management Group is generally not considered as a component model. Indeed, it is more commonly used to describe systems, components and component models rather than define the fundamental characteristics of a particular component model. Thus, UML2.0, with its component diagram among others, provides a

basic notation to describe component-based systems. It involves the concepts of components, composite components, ports, interfaces, connectors, etc. However, some features existing in other component models are not native in UML2.0 but can be added thanks to the stereotype, tag and profile mechanisms that allow tuning the original model to fit these new requirements.

2.2. SaveCCM

SaveCCM [3, 4], developed within the SAVE project and several Swedish Universities, is a component model specifically designed for embedded control applications in the automotive domain with the main objective of providing predictable vehicular systems. SaveCCM is a simple model that constrains the flexibility of the system in order to improve the analysability of the dependability and of the real-time properties. The model takes into consideration the resource usage, and provides a lightweight run-time framework. For component and system specification, SaveCCM uses "SaveCCM language" which is based on a textual XML-syntax and on a subset of UML2.0 component diagrams.

2.3. AUTOSAR

The **AUTomotive Open System Architecture** (AUTOSAR) [8] is the result of the partnership between several manufacturers and suppliers from the automotive field. It envisions the conception of an open standardized architecture. Moreover, each element has to be fully exchangeable between vehicle platforms, manufacturer's applications and supplier's solutions. All those objectives can be attained by the utilisation of both a component-based approach for the application and standardized layered architecture. This allows separating the component-based application from the underlying platform. AUTOSAR support both the client-server and Sender-Receiver communication paradigms and each AUTOSAR Software Component instance from a vehicle platform is only assigned to one Electronic Control Unit (ECU). The AUTOSAR Software Components, as well as all the modules in an ECU, are implemented in C.

2.4. Koala

Koala [9] is a component model developed by Philips for building consumer electronics. Koala components are units of design, development and reuse. Semantically, components in Koala are defined in a ADL-like language. Koala IDL is used to specify Koala component interfaces, its Component Definition Language (CDL) is used to define Koala components, and Koala Data Definition Language (DDL) is used to specify local data of components. Koala components communicate with their environment or other components only through explicit interfaces statically connected at design time. Koala targets C as implementation language and uses source code components with simple interaction model. Koala pays special attention to resource usage.

2.5. Pecos

Pecos [12] is a joined project between ABB Corporate Research and academia. Their goal is to provide environment that supports specification, composition, configuration checking and deployment for reactive embedded systems built from software components. Component specification and component composition is done in ADL-like language called CoCo. There are two types of components, leaf components and composite components. Furthermore, the components can be passive, active and event triggered. The inputs and outputs of a component are represented as ports. At design phase composite components are made by linking their ports with connectors. Pecos targets C++ or Java as implementation language, so the run-time environment in the deployment phase is the one for Java or C++. Pecos enables specification of EFP properties such as timing and memory usage in order to investigate in prediction of the behaviour of embedded systems.

2.6. Rubus

Rubus [11] component was developed as a joint project between Arcticus Systems AB and Department of Computer Engineering at Mälardalen University. The Rubus component model runs on top of the Rubus real-time operating system. It focuses on the real-time properties and is intended for small resource constrained embedded systems. Components are implemented as C functions performed as tasks. A component specifies a set of input and output ports, behaviour and a persistent state, timing requirements such as release-time, deadline. Components can be combined to form a larger component which is a logical composition of one or more components.

2.7. Fractal

Fractal [10] is a component model developed by France Telecom R&D and INRIA. It intends to cover the whole development lifecycle (design, implementation, deployment and maintenance/management) of complex software systems. It comes up with several features, such as nesting, sharing of components and reflexivity in that sense that a component may respectively be created from other components, be shared between components and describes its own behaviour. The main purpose of Fractal is to provide an extensible, open and generic component model that can be tuned to fit a large variety of applications and domains. Consequently, nothing is fixed in Fractal; On the contrary, it even provides means to facilitate adaptation in notably having different implementations to fit the specific needs of a domain as for example its C-implementation called Think, which targets especially the embedded systems. A reference implementation, called Julia and written in Java, is also provided and can also be used in some embedded systems. Finally Fractal can also be seen as a metamodel which intends to encompass other component models.

2.8. Eclipse

Eclipse [14], is currently the most largely used Integrated Development Environment (IDE). An IDE is simply a framework containing a programming environment integrated into a software application that provides all the different tools which are needed during the development process of software application. Eclipse comes with many development tools (Graphical User Interface builder, a text or code editor, a compiler and/or interpreter and a debugger). One of the characteristics, which have made Eclipse so popular, is its plug-in mechanism. Indeed, Eclipse offers possibility to adapt the environment by adding any type of tools.

3. Hot topics & Leading researchers

This section includes some of the hot topics in the field of Component-Based Software Engineering as well as a list of the leading researchers.

3.1. Hot topics

- **Components and model-driven development**

Model-driven development is one of software engineering hot topics at the moment. It mainly focuses on the use of different models through the whole lifecycle of a development process as well as how to transform one model to another with as many loss of information as possible according to the limitations of the models in presence. One of the main known initiatives in this domain comes from the Object Management Group (OMG) with its Model-Driven Architecture (MDA).

- **Tools and Environments**

Tools and Environments of developments are a particularly attractive field due to the need to provide new environments of development. Indeed, at present time, most of Integrated Development Environment (IDE) focus only on the programming aspect involved during the development of software but don't take into consideration the testing and verification aspects which are primordial to a trustworthy software development.

- **Composability of non-functional properties**

Nowadays, component-based systems take mainly into consideration the functional aspects, such as for instance the communication mechanism used between components. Nevertheless as the complexity of the systems grows and their reliability becomes more and more a central aspect, the question of the place and composition of non-functional properties increases. Indeed, even though it seems possible to set some extra-.functional properties to components, there still exists an uncertainty about the results of their composition.

- **System quality and dynamic binding**

This topic focuses mainly on the reliability of a system at execution time. Indeed, a component-based system which works correctly may need an update to add for instance new functionalities. This modification can be done by the insertion of a new component into the system at run-time but in such a context, how to guaranty that this change would not bring any instability into the system, that the system would not crashed or would react as expected? In the same way, how to identify which component need to be removed adapted or linked to as a consequence of the addition of this component? These are some of the quality issues that can be raised here.

3.2. Leading Researchers

This section lists some of the leading researchers in Component-Based Software Engineering.

- **The ARTIST2 NoE group**,
- **CBSE Symposium organisers:**
  - **Ivica Crnkovic**,
  - **Kurt Wallnau,**
  - **Judith Stafford,**
  - **Heinz Schmidt,**
  - **George Heineman,**
- **Kung-Kiu Lau,**
- **Michel Chaudron**¸
- **Fractal research Group,** constituted by Eric Bruneton, Thierry Coupaye, Jean-Bernard Stephani who have designed and developed the Fractal specification and the implementation of reference Julia. Others people have joined this project to developed other aspects of the Fractal Component Model: Philippe Merle, Nicolas Senturier, Romain Rouvoy, Nicolas Pessemier.
- **Paola Inverardi**
- **Jean-Marc Jézéquel,**
- **Bertrand Meyer,**
- **Rob van Ommering,**
- **Ralf Reussner,**
- **Douglas Schmidt,**
- **SOFA group,** constituted among others by Frantisek Plasil who has designed the SOFA component model.
- **Ian Sommerville,**
- **Clemens Szyperski**,

4. Key Conferences

This section presents some of the key conferences in component-based software engineering:

1. The International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)
2. The European Software Engineering Conference (ESEC)
3. The ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)
4. EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (SEAA) Component-Based Software Engineering Track
5. International Conference on Software Engineering (ICSE)

5. References:

This section lists with no distinction both seminal papers and references used to write this article:

CBSE:

[1] C. Szyperski Component Software - *Beyond Object-Oriented Programming, Second Edition*. Addision-Wesley, 2002.

[2] Ivica Crnkovic, Magnus Larsson - *Building reliable component-based software systems*. Artech House, Inc. 2002.

[3] M. Åkerholm et al., *The SAVE approach to component-based development of vehicular systems, Journal of Systems and Software*, Elsevier, May, 2006

[4] Åkerholm, M. et al, *The SaveCCM Language Reference Manual*, Version 0.4, 2006

[5] B. Jonsson and E. Brinksma and G. Coulson and S. Graf et I. Crnkovic and S. Gérard and H. Hermanns and J.-M. Jezequel and A. Ravn and Ph. Schnoebelen and F. Terrier and A. Votintseva, *Embedded Systems Design: The ARTIST Roadmap for Component-based Design and Integration Platforms*, May 6th, 2003

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

[7] Crnkovic, M. Larsson, O. Preiss, Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes,), Architecting Dependable Systems III,, p pp. 257 – 278, Springer, LNCS 3549, 2005

[8] AUTOSAR Development Partnership, *AUTOSAR – Technical Overview v2.0.1*, 27/06/2006, http://www.autosar.org/download/AUTOSAR_TechnicalOverview.pdf

[9] R. van Ommering, F. van der Linden, and J. Kramer. "*The koala component model for consumer electronics software*", In IEEE Computer, pages 78–85. IEEE, March 2000.

[10] E. Bruneton, T. Coupaye & J.B. Stefani, *The Fractal Component Model*, February 5, 2004. http://fractal.objectweb.org/specification/index.html

[11] Maaskant; "*A Robust Component Model for Consumer Electronic Products*", Philips Research Book Series Volume3, p167-192

[12] M. Winter, C. Zeidler, C. Stich, "*The PECOS Software Process*", Workshop on Components-based Software Development Processes, ICSR 7 2002.

(Meta-)Modelisation:

[13] The Object Management Group, *UML Superstructure Specification v2.1*, April 2006.
Available at http://www.omg.org/docs/ptc/06-04-02.pdf.

Integrated Development Environment

[14] Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht and Philippe Vanderheyden. *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. IBM Redbooks.Durham, NC, USA: IBM, 2004.