Software Component Technologies for Real-Time Systems
- An Industrial Perspective -

Anders Möller
MRTC
Mälardalen University
CC Systems
anders.moller@mdh.se

Mikael Åkerholm
MRTC
Mälardalen University
mikael.akerholm@mdh.se

Johan Fredriksson
MRTC
Mälardalen University
johan.fredriksson@mdh.se

Mikael Nolin
MRTC
Mälardalen University
mikael.nolin@mdh.se

Abstract

In this paper, we compare existing component technologies for embedded systems with respect to requirements captured from the vehicular industry.

The vehicular industry wants to make use of the advantages with component based design; however they also need to address non-functional properties of their products, such as reliability and timeliness. Several component technologies addressing such properties have recently been proposed. In this paper, we present initial findings from an ongoing evaluation concerning some of these technologies with respect to the requirements stated by industrial actors.

We conclude that none of the studied technologies is a perfect match for the industrial requirements. Furthermore, no single technology stands out as being a significantly better choice than the others; each technology has its own pros and cons.

1 Introduction

During the last few years, component-based software engineering for embedded real-time systems has received a large amount of attention in the research community. However, industrial software developers are still, to a large extent, using monolithic and platform dependent software technologies.

Often companies can achieve considerable business benefits in terms of reduced costs, shortened time-to-market and increased software quality by applying a suitable component technology. There is however significant risks and costs associated with the adoption of a new development technique. These must be carefully evaluated before introduced in the development process.

Our approach in this paper is to study some of the existing component technologies suitable for distributed embedded real-time systems and to compare these technologies with industrial requirements [16]. The main purpose of this Work in Progress paper is to disclose our initial findings and to solicit feedback on which techniques to study and what requirements are of interest.

2 Industrial Requirements

The benefits of using a component based technique can be divided into two different aspects, the operational benefits (e.g. reliability and safety) and the development benefits (e.g. reusability and maintainability). The requirements on such a component based technique can, in the same way, be divided into technical- and development requirements.

Apart from the requirements addressed later in the paper, safety and robustness are evident requirements on a vehicular system. The system should function correctly in stressful environmental conditions and perform its required functions under stated conditions for a specified period of time without any catastrophic consequences to the environment. However, safety and robustness are not easy for a component technology to consider, since these requirements are mainly related to system design and implementation.

The requirements are obtained from interviews with senior technical staff at two Swedish companies, CC Systems [1] and Volvo Construction Equipment [2]. These companies develop control software for large, low-series vehicles (e.g. wheel loaders and forest harvesters) and their systems are characterised as safety critical distributed embedded real-time systems with limited hardware resources.

Our definitions of the elicited requirements, listed below, include important aspects of the introduction of a component-based development technique. These definitions, including both technical merits and demerits, are somewhat different, or should be seen as an extension, of the generally used definitions.

2.1 Technical Requirements

Analysable – the chosen technique should be easy to analyse with respect to non-functional properties, such as the timing behaviour and the memory consumption. It is important to be able to both verify if the tasks meet their deadlines and to be able to analyse the end-to-end timing behaviour of the complete system.

The components should be configured at compile-time, to make them smaller and easier to analyse statically.

Modelling and Computation – based on information extracted during the interviews, the technique should be based on a standard modelling language like UML [3]. The components should preferably be passive, focusing on a pipe-and-filter computation model [4]. The reason to be that restrictive in the choices concerning the modelling and computations is related to simplicity and the use of mature techniques.

Open - a component should be source code, i.e., no binaries. The reasons for this include that companies are used to have access to the source code, to find functional errors and enable support for white box testing. The possibility to look into the components does not necessary mean that you are allowed to modify them.

Portable – the components, and the infrastructure surrounding them, should be platform independent to the highest degree possible. In order to support platform independency, the components should not use the operating system primitives or the processor features directly. This is an important requirement because of the frequently shifting hardware and operating system needs.

Resource Constrained – the systems considered (distributed embedded real-time systems) are usually resource constrained, when it comes to the CPU and the memories. Therefore, the software systems should be light-weighted and the components infrastructure should be minimised.

2.2 Development Requirements

Maintainable - the component should be easy to change and maintain, e.g., for use in new applications or environments than those for which it was originally designed.

Introducable - the possibility for companies to gradually migrate into the chosen technique, not jumping in to the new technique to fast, is important, to make the change in technique as safe and inexpensive as possible.

Reusable - the components should be easy to reuse and the technique, and its supporting tools, should offer support for component version management. To have good support for version and variant management is a very important requirement, because it reduces the risk of reinventing components – after all, software reuse is one of the most important aspects when introducing a component based development technique.

Understandable - the system should be easy to understand, to simplify evaluation, and verification both on the system level and on the component level. This should also include making the technology easy and intuitive to use in a development project.

3 Existing Component Technologies

In this section, existing component technologies for embedded systems are described. The technologies considered originate both from academia and industry. The selection criterion for a component technology has firstly been that there is enough information available, secondly that the authors claim that the technology is suitable for embedded systems, and finally we have tried to achieve a combination of both research and industry examples. The technologies described and evaluated are PECT, Koala, Rubus Component Model, PBO, PECOS and CORBA.

3.1 PECT

Prediction-Enabled Component Technology (PECT) [5] is a development infrastructure that incorporates development tools and analysis techniques. PECT is ongoing research project at the Software Engineering Institute (SEI) at Carnegie Mellon University.

PECT defines that any component technology can be used if composition rules guarantee runtime properties, by enforcing that predictable construction patterns are used. What is allowed by a user, and what is required by the underlying component technology, is determined by the available analysis methods and prediction goals.

PECT focuses mainly on analysis; assumed that the prediction framework contain prediction techniques for the desired properties; a high grade is motivated on this requirement. PECT is also portable and introducable, because of its independence of the underlying technology.

As PECT is highly analysable, portable and introducable, it is not very understandable. In order to understand the model, the mapping to the underlying component technology must be understood as well.

3.2 Koala

The Koala component technology [6] is tailored for development of software in consumer electronics, and it is developed and used by Philips [7].

Consumer electronics are often resource constrained since they use cheap hardware to keep development costs low. Koala pays special attention to resource usage through a thread sharing technique. The thread sharing technique keeps the number of threads low, which in turn keep the memory utilisation low. The implementation is realised with message queues which have a function to process messages in the context of a thread.

All components in Koala are source code components and are therefore totally open for inspection. This makes it easier for companies to find functional errors and enables white-box testing. The technology is also understandable; it builds on simple and mature techniques.

An obvious problem with Koala, compared to the requirements is that it seems hard to gradually introduce the technology. Koala components are tightly coupled to the Koala compiler, and the underlying operating system. The components use the same interaction mechanisms in between each others as towards the operating system.

3.3 Rubus Component Model

Rubus is developed by Arcticus systems [8], with support from the research community, and is, e.g., used by Volvo Construction Equipment.

The Rubus component model is tailored for resource constrained systems with real-time requirements. Rubus has a red and a blue part for hard and soft real-time respectively. The red kernel is used for time-critical applications and is therefore time-triggered. The blue kernel is event-triggered, and used for less time-critical applications.

The computation model provided by Rubus is the desired pipe and filter model, very simple and suitable for control applications. Like Koala, Rubus also has source-code components. The components are hence open for inspection and white-box testing.

A requirement that is not met is the constraint of portability. The Rubus component model is too tightly coupled to the Rubus operating system since it is shipped with, and developed on top of, the Rubus operating system.

3.4 PBO

Port Based Objects (PBO) [9] combines object oriented design, with port automaton theory. PBO was developed as a part of the Chimera RTOS project [10] at the Advanced Manipulators Laboratory at Carnegie Mellon University. Together with Chimera, PBO forms a framework aimed for development of sensor-based control systems, with specialisation in reconfigurable robotics applications.

An explicit design goal for a system based on PBO is to minimise communication and synchronisation, thus facilitating reuse. PBO is a simple and intuitive model which is highly understandable, both at system level and within the components themselves; hence the requirement of understandability is satisfied.

While PBO is very intuitive, it is also tightly coupled with its RTOS, Chimera. Therefore it is hard to introduce parts of PBO in present system configurations. Because of the dependencies on the RTOS, PBO can not be considered very portable.

3.5 PECOS

PECOS [11] is a collaborative project between industrial and research partners. The goal for the PECOS project is to enable component-based technology for embedded systems, especially for field devices, i.e. embedded reactive systems. The project tries to consider non-functional properties very thoroughly in order to enable assessment of the properties during construction time.

There is no special run-time environment developed in the PECOS project. Instead there are requirements on platform independence, or at least on portability.

The PECOS project has incorporated the Unified Modelling Language (UML) for modelling the system. This makes the model attractive considering the requirement of model and computation.

Furthermore, PECOS is a research project and much focus has been put on non-functional properties such as memory consumption, timeliness etc. which makes PECOS analysable.

The requirement of openness is not considered fulfilled, due to the fact that PECOS uses black-box components. In later releases, the PECOS project is considering to use a more open component model [12].

3.6 CORBA Based Technologies

The Common Object Request Broker Architecture (CORBA) is a standard that provides a set of rules for writing platform independent applications. The CORBA standard is developed by the Object Management Group (OMG) [13].

A major drawback with CORBA is that it requires a lot of functionality in order to connect diverse platforms within a heterogonous system. Because of this, variants of CORBA exist, two major are Minimum CORBA [14] for resource constrains systems, and RT-CORBA [15] for time-critical systems.

OMG has also defined a CORBA Component Model (CCM) [17]. CCM extend the CORBA object model by defining features and services that enable application developers to implement, mange, configure and deploy components that integrate commonly used CORBA services.

Because CORBA is a middleware architecture that defines communication between nodes, it becomes highly portable. While CORBA is portable, and powerful, it is also very run-time demanding. In CORBA, bindings are performed during run-time. Therefore the requirement of analysability can not be considered fulfilled. Dynamic binding is very computation intense, hence CORBA is not suitable for resource constrains systems. CORBA is using binary components, i.e. the components are closed, and inspection or white-box testing is out of the question.

4 Summary of Evaluation

Table 1, shows a summary of the initial evaluation of component technologies for embedded vehicular systems presented in the paper. The evaluation of the different technologies is based on the requirements defined in section 2.

3 = *Good, the requirements are very well fulfilled.*
2 = *Satisfactory, the requirements are to some extent fulfilled*
1 = *Bad, the requirements are not or very little fulfilled*
NA= *Not Available, requirement is not adressed*
IN = *Inconclusive, not determined*

Require. \ Model	Koala	Rubus	PECT	PBO	Corba	PECOS
Analysable	1	2	3	2	1	3
Model and computation	2	2	NA	2	1	3
Open	3	3	NA	IN	1	1
Portable	1	1	3	1	3	IN
Resource constrains	3	2	NA	2	1	2
Maintainable	3	2	2	2	1	1
Introducable	1	2	3	1	3	1
Reusable	3	2	1	2	1	2
Understandable	3	2	1	3	1	2

Table 1: A summary showing how well existing component technologies fulfil industrial requirements.

5 Conclusion and Future Work

Our conclusion, based on the industrial requirements, is that there is no one-component technology available that fulfil all the requirements listed in section 2. However, some of the technologies are based on interesting techniques and concepts.

We have noticed that, for a component technology to be fully accepted by industry, the whole systems development context needs to be considered. It is not only the technical properties, such as modelling, computation model, and openness, that needs to be addressed, but also development requirements like maintainability, reusability, and to which extent it is possible to gradually introduce the technology. It is however important to keep in mind that a component technology alone cannot be expected to solve all these issues.

We will continue to investigate the industrial requirements in more detail, and also continue to capture requirements by cooperating with other industrial partners. We will also assess to what extent existing technologies can be adapted in order to fulfil the requirements, or whether selected parts of existing technologies can be reused if a new component technology needs to be developed.

6 References

[1] CC Systems homepage, http://www.cc-systems.com

[2] Volvo Construction Equipment homepage, http://volvoce.com

[3] Selic, B., Rumbaugh, J., Using UML for modelling complex real-time systems, Rational Software Corporation 1998

[4] M Shaw, D. Garlan, Software Architecture: Perspectives on an Emerging Discipline. PrenticeHall 1996

[5] K. C. Wallnau. Volume III: A Technology for Predictable Assembly from Certifiable Components, Technical report, Software Engineering Institute, Carnegie Mellon University, April 2003, Pittsburgh, USA

[6] R. van Ommering, F. van der Linden, and J. Kramer. The Koala component model for consumer electronics software. IEEE Computer, 33(3):78–85, March 2000.

[7] Arcticus Systems Home Page. http://www.-arcticus.se.

[8] Philips, Home Page http://www.philips.com

[9] D. B. Stewart, R. A. Volpe, P. K. Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects, IEEE Transactions on Software Engineering, December 1997, pages 759-776.

[10] P. K. Khosla et al., The Chimera II Real-Time Operating System for Advanced Sensor- Based Control Applications, IEEE Transactions on Systems, Man and Cybernetics, 1992

[11] O. Nierstrasz, G. Arévalo, S. Ducasse, R. Wuyts, A. Black, P. Müller, C. Zeidler, T. Genssler, R. van den Born, A Component Model for Field Devices Proceedings of the First International IFIP/ACM Working Conference on Component Deployment, Germany, June 2002.

[12] R. Wuyts, S. Ducasse. Non-Functional Requirements in a Component Model for Embedded Systems, In International Workshop on Specification and Verification of Component-Based Systems, OOPSLA 2001.

[13] Object Management Group. CORBA Home Page. http://www.omg.org/corba/

[14] Object Managment Group. Minimum CORBA 1.0, http://www.omg.org/technology/-documents/formal/minimum_CORBA.htm

[15] D.C. Schmidt, D.L. Levine, and S. Mungee. The Design of the tao real-time object request broker. *Computer Communications Journal,* Summer 1997

[16] Möller A., Fröberg J., Nolin M., What are the needs for components in vehicular systems? - An industrial perspective -, In Proceedings of the WiP Session of the 15th Euromicro Conference on Real-Time Systems 2003.

[17] OMG, CORBA Component Model 3.0, June 2002, http://www.omg.org/technology/documents/formal/components.htm