What are the needs for components in vehicular systems? – An industrial perspective –

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

Joakin Fröberg
Mälardalen University
Volvo Construction Equipment
joakim.froberg@mdh.se

Mikael Sjödin
Mälardalen University
mikael.sjodin@mdh.se

Abstract

During the last few years, component based software engineering for embedded systems has received a large amount of attention. Often, however, the approach has been to adopt existing component technologies to be more suited for embedded systems. These "embeddified" versions of desktop/Internet technologies seldom find their way into the embedded systems market, for instance, due to timing unpredictability and high resource demands.

Our hypothesis is that there is no one-component technology suitable for all segments of the embedded systems market. Instead, we believe that different segments of the embedded systems market may be best served by different component technologies. In this paper, we focus on the market for heavy vehicles, such as wheel loaders, dumpers, and forest harvesters. Our approach is to study companies within this market segment and find their requirements on a component technology. In this paper we present initial findings from interviews with senior technical staff at two Swedish companies within the studied segment.

1 Introduction

This paper presents initial results of an ongoing project called "Component Technology for Heavy Vehicles" (HEAVE). The goal of the project is to identify, define and evaluate a software component technology within the business segment of heavy vehicles.

A component technology consists of a component model, an infrastructure (so called middleware), and tools for creating, composing and analysing components. During the last decade, the PC-/web-oriented software engineering community has achieved tremendous progress in component oriented software construction. Today, it is possible to download components on the fly and have them executed within the context of another program such as a web browser or a word processor. Software developing companies can purchase off-the-shelf components and embed them into their own software products. Technologies like CORBA [9], Java Beans [14], .NET [6], and other component models are used on a day-to-day basis in systems software development. However, existing component technologies are not applicable to most embedded computer systems. They do not consider aspects such as safety, timing, and memory consumption that are crucial for many embedded control systems. Some attempts have been made to adapt component technologies to embedded systems, like Minimum Corba [10] for example. However, these adaptations have not been generally accepted in the embedded systems segments. The reason for this failure is mainly due to the diversified nature of the embedded systems domain. Different market segments have different requirements on a component technology, and often these requirements are not fulfilled simply by stripping down existing component technologies.

2 Project Outline

Instead of starting from an existing technology and trying to "embeddify" it, this project take a different approach in that it will start unbiased and identify the specific requirements on a component technology for the heavy vehicles segment.

It is important to keep in mind that the embedded systems market is extremely diversified in terms of requirements placed on the software. For instance, it is obvious that software requirements for consumer products, telephony switches and avionics are quite different. Hence, our focus on one single market segment. It is important to realise that the development of a component technology is substantially simplified by focusing on a specific market segment. Within this market segment, the conditions for software development should be similar enough to allow a lightweight and efficient component technology to be established.

When the requirements are understood, we will study to what extent existing technologies fulfill those requirements.

We will also assess to what extent existing technologies can be adapted in order to fulfil the requirements, or whether selected parts like tools, middleware, message-formats and file-formats can be reused if a new component technology needs to be developed.

Once we have a better understanding of the requirements and understand to what extent existing technologies fulfil those requirements, we will make a specification of a suitable component technology. The specification will cover issues like component representation, interface descriptions, middleware functionality, component interoperability, message formats, etc.

From the specification, we will build a test-bed implementation of the component technology, reusing as much as possible of the existing technologies. The test-bed will be used to evaluate the component technology, and for a pilot project implementing some functions in a real vehicle environment.

3 Requirements Capture

To better understand the needs in the business area of heavy vehicles, this project has started with interviews with senior technical staff at two Swedish companies, Volvo Construction Equipment [15] and CC Systems [2]. Both companies develop on-board electronics and software systems for heavy vehicles, like dumpers and combat vehicles. They experience similar problems, related to the development of software for embedded real-time systems. By cooperating in this research project, in enabling a component technology for heavy vehicles, their joint desire is to improve the development process of on-board software systems.

3.1 Technical background

Distributed systems in mobile applications become more and more complex; vehicles are equipped with advanced electronics both as means of improving their capacity and functionality, and as a mean to decrease production costs (i.e. replacing mechanics with electronics). The electronic systems developer faces challenges of shorter development time and keeping the electronics part of the product cost to a minimum.

Companies often develop new systems in an evolutionary way, i.e. new systems are partially based upon previously developed systems. Typically, a company also develops a product line, i.e. a variety of related systems. A product-line approach to development is an effort to create an overall development process taking into account a whole product line. The aim is to avoid suboptimisation and lift the focus from single products. By focusing on the software architecture, developers want to get a high-level view of the system's properties. A product-line architecture approach is an overall strategy to address both the objectives of the product-line and its software architecture, in order to achieve system quality attributes, reduce development cost, shorten time-to-market, and reduce maintenance cost.

Not having a well specified development procedure to coordinate development in terms of processes, methods and technology, makes development expensive. One way of reducing time spent on development is to reuse software components and architectural solutions between products. There are different aspects of the advantages of using a component-based approach. These aspects can be divided into operational properties (e.g. reliability, safety, and real-timeliness) and development properties (e.g. reusability, scalability, configurability, and maintainability). One of the most important resulting effects of using a component-based approach is a shorter and more predictable development time.

Common desktop/Internet component technologies, such as COM, Java Beans and CORBA, are considered unfit for use in the on-vehicle control systems because of their excessive resource usage and unpredictable timing behaviour. However, selected parts (like tools and message formats) and ideas from these technologies can be reused.

To have companies actually using a component based software approach, the component models and middleware must be fit for their specific needs. Aiming to high, by trying to find the component technology, that can be used in all distributed embedded real-time systems, will most likely lead to yet another flexible but far to memory and time consuming model unfit for the companies needs.

3.2 CC Systems

CC Systems (CCS) is developing and supplying advanced distributed embedded real-time control systems with focus on mobile applications. Examples of control systems, including both hardware and software, developed by CCS are forest harvesters, rock drills, and combat vehicles.

3.2.1 Company background

Systems developed by CCS are characterised by rough environments, safety criticality, high functionality, and the requirements on robustness and availability are high. In the future, CCS focus on being a platform supplier (hardware, device drivers and middleware), as a complement to being a specialised application development company. The companies using the platform developed by CCS should develop their own applications, using market leading tools and methods (e.g. Rapsody, IEC 61131-3 [3]).

CCS´ goal is to use a component-based approach towards software construction, to enhance the ability to reuse

and analyse applications and because it reduces the degrees of freedom for application developers. This reduction in freedom, in turn, will minimise the risk for software errors, because component assembly can only be done in a predefined manner.

CCS expect that future systems will need a higher degree of configurability and greater ability to integrate third party software. Use of a component approach will facilitate the needed flexibility.

3.2.2 Techical Future

The component model used, preferably based on a standard modelling language like UML, UML-2 or RT-UML should be platform independent and provide support for integration with third party software.

The component model should preferably be based on passive components focusing on a pipe-and-filter solution. A component should be open source, i.e. no binaries, and should be platform independent. In order to support platform independency the components are not to use the operating system primitives or processor features directly. Components can, for example, be a gathering of objects using a common API. The components should be configured at compile-time to make them smaller and easier to analyse statically. All components must have a pre-calculated worst case execution time, making it possible to schedule tasks off-linc, to chcck if thc tasks mcct thcir dcadlincs, and to analyse the end-to-end timing behaviour of the complete system.

Some important questions need answers: What is the reason why existing component technologies for embedded systems are not used more frequently? Is it possible to use selected parts of the existing CBSE techniques (like tools and message formats) to develop a specialised heavy vehicle component technique? Is it possible to reach analysability of an end-to-end system using many different levels of safety integrity levels, i.e. safety critical, real-time critical, and information systems? Is it possible to use a subset of CORBA, by retaining selected parts like message formats for example, to provide an opportunity to communicate with other, more powerful, nodes running for example a full version of CORBA.

3.3 Volvo Construction Equipment

Volvo Construction Equipment (VCE) is one of the world's major manufacturers of construction equipment, with a product range encompassing wheel loaders, excavators, articulated haulers, motor graders, and more. What they all have in common is that they demand appropriate technical solutions and equipment that can help them to improve their performance. This project only focuses on the on-board electronics and software systems part of VCE´s business area.

3.3.1 Company background

VCE develops both on-board electronics and software. The systems are distributed embedded real-time systems with typically some five nodes, communicating via CAN. Another characteristic is that the software must perform in an environment with limited resources.

To accommodate reuse of software components and methodology between products, VCE has incorporated a component model for the real-time application domain. However, VCE wishes to strengthen its competence with component based software development in general. The resulting component technology will be used to extend their current practises within components engineering. They expect that achieving a higher competence and better approach to component based software engineering will significantly reduce software related costs.

Today VCE uses the real-time operating systems Rubus (from Arcticus Systems). Rubus encompasses, besides an operating system, a component model and tools to compose components into distributed real-time systems. All Rubus components have a known worst case execution time, and Rubus provides a tool to off-line produce static schedules for component execution. Hence, Rubus provides a good foundation for building component-based distributed real-time systems.

The design of VCE's current software architecture was done with the intent of using it for a relatively long time. The architecture was to be a base for development of several products over time. In order to be successful in the development effort, the desired properties (such as timeliness and memory consumption) of the system were considered already on the architecture stage.

3.3.2 Technical Future

VCE uses software component both to develop the infrastructure that applications execute upon (this infrastructure can be viewed as a kind of middleware), as well as for the actual applications executing on the infrastructure. Today, a typical software component is a rather big entity (e.g. a component for transmission control).

Analysing the software components source code using tools like Lint is important to reduce the risk of software failures in components. Memory usage and worst-case execution-time are other important non-functional properties which needs to be statically analysed.

All in all, analysability and testing is a very important to VCE. The component model used at VCE today gives some support for analysing timing related issues. However, some support, like end-to-end timing analysis is not supported by

tools (and is hence performed manually). VCE are also interested in functional testing of the components and the complete systems. Automatic generation of test cases are considered as an highly desirable tool for the future. Also tools for on-line monitoring the software and debug tools are considered important.

The only model of computation supported by the Rubus component model is the "pipes-and-filter" model. However, VCE has seen little (or no) need for other models of computation. This reflects the fact that most software in VCE's system is control related

There are no urgent needs for communication with third party software components. However, VCE is interested in having the opportunity to use third party software inside the components and also, taking the long view, to communicate with other (third-party) hardware nodes running, e.g., CORBA.

Today, VCE uses a component-based approach for development of their product line. One significant problem that has emerged due to the component-based approach is version and variant management. Shifting to component-based software-engineering causes novel problems for administration of the components' life-cycles. That is, new processes are needed to administer variant development and feedback of component modifications to projects using the components.

4 Related Work

Technologies like COM/DCOM [5], CORBA [9, 8], Java Beans/Enterprise Java Beans [14, 13], .NET [6] readily available and used by developers on a day-to-day basis. However, these technologies have all been developed for the PC-/Internet-market, and are usually not applicable for embedded control systems.

The IEC 61131-3 [3] standard allows component based development of simple control-applications. The port-based objects [12] approach goes one step further and allow reusable components to be interconnected by means of input and output ports. The Rubus operating system (by Arcticus [1]) provides an implementation of port-based objects that can be executed in a distributed environment.

The European research project PECOS (PErvasive COmponent Systems) [11, 7] focuses on the architectural issues of component-based software construction for embedded systems. PECOS has an architectural focus and its ambition to target the complete embedded systems market. This will probably cause their results not to be directly applicable to our studied market segment. However, PECOS will serve as one important source of information for this project.

Within the software engineering community, AOP has recently received a large amount of attention [4]. AOP provides a mechanism to configure software (or components) at compile-time. Since this configuration is done at compile-time, unnecessary functionality may be stripped away; yielding a smaller run-time version of the components than would configuration during run-time.

5 Contribution

The scientific contributions of this project are mainly the study of actual requirements from an industrial perspective and the survey of to what extent those requirements are fulfilled by existing component technologies. In addition, the implementation of a test-bed and a pilot project will have a scientific value, illustrating how a technology based on industrial requirements can be used to solve problems that are not solved by commodity technologies.

For the participating companies the specification of a component technology that is suitable for the considered market segment will be the main contribution. The test-bed implementation and pilot project will also provide valuable insight into how the new component technology can be used at the participating companies. An indirect contribution for the participating companies is also the increased competence within component-based software construction gained.

6 Summary

In this paper we have described work in progress within a project called "Component Technology for Heavy Vehicles". The project goal is to investigate the need and requirements for component based software-engineering for software for heavy vehicles.

The approach of this project is to start from actual requirement on a component technology and focus on a single market segment (in this case the segment for heavy vehicles). After requirement has been documented the next step is to define a component technology that satisfy those requirements. The goal in this second phase is to reuse as much as possible for existing component technologies. Examples that will be considered for reuse are message formats, middlewares, interface description languages, etc.

We have presented two companies that develop on-board electronics and software for heavy vehicles. Examples of the type heave vehicles considered by these companies are wheel loaders, dumpers, forest harvesters and bandwagons. The companies respective technical background was described are we could see that for a component technology to be applicable the whole systems development context needs to be considered. That is, not only the specific properties of components and middleware needs to be addressed, also issues like component life cycles, variant handling, static

(i.e. off-line) analysis, and testing needs to be considered to some extent. Its however important to keep in mind that a component technology alone cannot be expected to solve all these issues. Nevertheless, for a component technology to be accepted in this market segment it cannot introduce difficulties in these important (non-functional) domains of systems development.

References

[1] Arcticus Systems. The Rubus Operating System. http://-www.arcticus.se.

[2] CC Systems. http://www.ccsystems.se.

[3] IEC. Application and Implementation of IEC 61131-3, 1995.

[4] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. L., J.-M. Loingtier, and J. Irwin. Aspect oriented programming. In *11^{th} European Conference on Object-Oriented Programming*, volume 1241 of *LNCS*, pages 220–242. Springer Verlag, 1997.

[5] Microsoft. Microsoft COM Technologies. http://www.-microsoft.com/com/.

[6] Microsoft. .NET Home Page. http://www.microsoft.com/-net/.

[7] P. O. Müller, C. M. Stich, and C. Zeidler. *Building Reliable Component-Based Software Systems*, chapter Component Based Embedded Systems, pages 303–323. Artech House publisher, 2002. ISBN 1-58053-327-2.

[8] Object Management Group. CORBA Component Model 3.0, June 2002. http://www.omg.org/technology/-documents/formal/components.htm.

[9] Object Managment Group. CORBA Home Page. http://-www.omg.org/corba/.

[10] Object Managment Group. Minimum CORBA 1.0, August 2002. http://www.omg.org/technology/documents/formal/-minimum_CORBA.htm.

[11] PECOS Project Web Site. http://www.pecos-project.org.

[12] D. B. Stewart, R. A. Volpe, and P. K. Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. *IEEE Transactions on Software Engineering*, 23(12), 1997.

[13] SUN Microsystems. Enterprise Javabeans Technology. http://java.sun.com/products/ejb/.

[14] SUN Microsystems. Introducing Java Beans. http://-developer.java.sun.com/developer/onlineTraining/-Beans/Beans1/index.html.

[15] Volvo Construction Equipment. http://www.volvoce.com.