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

Anders Möller
Mälardalen University
CC Systems AB
Anders.Moller@mdh.se

Joakim 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 unpredictable timing behaviour 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 and forest harvesters. Our approach is to study companies within this market segment and find their requirements on a component technology. This paper presents work in progress, based on 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 project is in its initial phase, and has started with interviews of senior technical staff at two Swedish companies. 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 (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 [1], Java Beans [2], .NET [3], and other component models are used on a daily basis in systems software development. However, existing component technologies are not applicable to most embedded computer systems, since 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 [4] for example. However, these adaptations have not been generally accepted in the embedded systems segments. The reason for this 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 takes 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 fulfil 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 [5] and CC Systems [6]. 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 ennobling 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. 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 [7] to development is an effort to create an overall development process taking into account a whole series of similar products. The aim is to avoid sub-optimisation 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.

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 too 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 too memory- and time consuming model.

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 and combat vehicles.

3.2.1. Company background. Systems developed by CCS are built to manage rough environments, and are characterised by safety criticality, high functionality, and the requirements on robustness and availability are high. In the future, CCS will 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. Rhapsody [8], IEC 61131-3 [9]).

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 of freedom, in turn, will minimise the risk for software errors, since component assembly can only be done in a predefined manner.

CCS expects 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. Technical Future. The component model used, preferably based on a standard modelling language like UML [10], UML-2 [11] or UML-RT [12] 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 [15] solution. A component should be open source, i.e. no

binaries, and must be platform independent. In order to support platform independency, the components are not to use the operating system primitives or the 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-line, to check if the tasks meet their deadlines, and to analyse the end-to-end timing behaviour of the complete system.

Some important questions need answers: Which are the reasons 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 use a subset of an existing technology, by retaining selected parts like message formats for example, and still provide an opportunity to communicate with other, more powerful, nodes running the full version of that technology?

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, 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. The focus of this project will only be on the on-board electronics and software systems part of the VCE business area.

3.3.1. Company background. VCE develops both on-board electronics and software. The systems are distributed embedded real-time systems, which must perform in an environment with limited hardware resources.

To accommodate reuse of software components and methodology between products, VCE has incorporated a component model for the real-time application domain. However, they 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 component engineering. VCE 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 [13]. Besides an operating system, Rubus encompasses a component model, and tools for assembling components. All Rubus components have a known worst-case execution time. Rubus also provides a tool to produce off-line static schedules for the 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 at the architecturing stage.

3.3.2. Technical Future. VCE uses software components 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. 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 [14] is important to reduce the risk of software failures in components. Memory usage and worst-case execution time are important non-functional properties that need to be statically analysed.

Overall, analysability and testing is important to VCE. The component model used by VCE today, gives some support for analysing timing related issues, but lack support for end-to-end timing analysis. VCE are also interested in functional testing of the components and the complete systems. Automatic generation of test cases are considered a highly desirable tool for the future. In addition, tools for on-line monitoring of the software and debug tools are considered important.

The only model of computation supported by the Rubus component model is the pipe-and-filter [15] 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, for example, 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 the projects using the component.

4. Related Work

Technologies like COM/DCOM [3], CORBA [1, 4], Java Beans/Enterprise Java Beans [2], .NET [20] are readily available and used by developers on a daily basis.

However, these technologies have all been developed for the PC-/Internet-market, and are usually not suitable for embedded control systems.

The IEC 61131-3 standard [9] allows component-based development of simple control applications. The port-based objects [16] approach goes one-step further and allows reusable components to be interconnected by means of input and output ports. The Rubus operating system (by Arcticus [13]) provides an implementation of port-based objects that can be executed in a distributed environment.

The European research project PECOS (PErvasive COmponent Systems) [17, 18] 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.

Within the software engineering community, Aspect Oriented Programming (AOP) [19] has recently received a large amount of attention. AOP provides a mechanism to configure software at compile-time.

5. Contribution

The scientific contributions of the HEAVE 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 by the participating companies. An indirect contribution for the participating companies is also the increased competence within component-based software construction gained during the project work.

6. Summary

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

The approach of this project is to start from actual requirements on a component technology, and focus on a single market segment (in this case the segment for heavy vehicles). After the requirements have been documented, the next step is to define a component technology that satisfies 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, such as wheel loaders, dumpers, forest harvesters, and combat vehicles. The companies' technical background has been described. We have noticed that, for a component technology to be applicable, the whole systems development context needs to be considered, i.e. not only the specific properties of components and middleware needs to be addressed but also issues like the component life cycles, variant handling, static (i.e. off-line) analysis, and testing needs to be considered to some extent. It is 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.

7. References

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

[2] SUN Microsystems. Java homepage. http://java.sun.com/

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

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

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

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

[7] Clements, P., Northrop, L., Software Product Lines, Addison-Wesley, August 2001, ISBN 0-201-70332-7

[8] I-Logix homepage, Rhapsody, http://www.ilogix.com/

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

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

[11] J.Warmer, T.Clark, UML 2.0 The future of OCL, Report of the workshop held on October 2, 2000, York

[12] Douglas, B.P., Real-Time UML – Developing efficient objects for embedded systems, Addison Wesley Longman, Inc, 1998

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

[14] S. C. Johnsson, Lint, a C Program Checker, Comp. Sci. Tech Rep. No 65 (D), updated version TM 78-1273-3

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

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

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

[18] 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.

[19] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira L., J.-M.Loingtier, and J. Irwin. Aspect oriented programming. In 11th European Conference on Object-Oriented Programming, volume 1241 of LNCS, pages 220_242. Springer Verlag, 1997.

[20] Microsoft. .NET home page. http://www.microsoft.com/net/