Industrial Grading of Quality Requirements for Automotive Software Component Technologies

Anders Möller[1,2], Mikael Åkerholm[1,2], Joakim Fröberg[1,3], Mikael Nolin[1,2] Robert Larsson[3]
[1]Mälardalen Real-Time Research Centre (MRTC), Västerås
[2]CC Systems, www.cc-systems.com
[3]Volvo Construction Equipment, www.volvoce.com

E-mail: anders.moller@mdh.se

Abstract

Software component technologies for automotive applications are desired due to the envisioned benefits in reuse, variant handling, and porting; thus, facilitating both efficient development and increased quality of software products. Component based software development has had success in the PC application domain, but requirements are different in the embedded domain and existing technologies does not match.

One challenge in devising a specially tailored component technology for the automotive embedded domain is that industrial requirements on such a technology are poorly understood. In this paper we present a set of graded industrial requirements on such a component technology.

The results can be used to guide modifications and/or extensions to existing component technologies in order to make them better suited for industrial deployment in the automotive domain. The results can also serve to guide other software engineering research by showing the most desired areas within component-based software engineering.

1 Introduction

During the last decade, Component-Based Software Engineering (CBSE) for embedded systems has received large attention, due to the envisioned benefits, e.g., improved quality and lowered development cost [1]. In the PC application area CBSE has already had large impact, supported by component technologies software development is today faster and products have higher quality. In the embedded system industry however, the component technologies for PC software are not viable and CBSE is still seen as an immature, but promising technique.

The reason for not adopting the component technologies from the PC domain is that requirements are often very different for embedded automotive applications. One challenge in devising a specially tailored component technology for the automotive domain is that the requirements on such a technology are poorly understood. Industry is facing significant risks and costs associated with the adoption of a new technology. Although benefits include reuse with shorter time-to-market and potential quality improvements, there are both technical concerns for increased complexity, but also concerns for development process changes. In order to device a new component technology the requirements from industry should be elicited and risks should be evaluated.

We have focused our industrial research project on finding the most important requirements - both technical and development process related - in order to find the critical issues to focus on when moving onto component-based techniques.

In [5] we describe requirements on a component technology as elicited from two companies in the business segment of heavy vehicles (e.g. construction, forestry and combat vehicles). The case-study focused on the question: What do you consider are the most important aspects on component-based development for vehicular control systems? The question was answered by senior technical specialists at two Swedish companies within the business segment of heavy vehicles. The main contribution from the initial study was that it straightened out some of the question-marks regarding actual industrial requirements. Another interesting note from this work was that a major part of the requirements was non-technical.

In this paper we extend our previous work by presenting the results from a second industrial study focusing on grading the requirements described in [5]. The motivation of grading requirements is that the results can be used to guide researchers and tool vendors to put focus on the most relevant industrial requirements, and to resolve conflicts between requirements.

This paper is outlined as follows: Section 2 introduces heavy vehicle systems, and in section 3 we describe the research method used when assembling and grading the requirements. In section 4 we describe the requirements, and in section 5 we present the requirements grading and discuss the results. In Section 6 we conclude the paper and discuss openings for future work.

2 Heavy Vehicles

What distinguishes the segment of heavy vehicles from the rest of the automotive industry (e.g., trucks and cars) is that the product volumes are typically lower. Also the customers tend to be more demanding with respect to technical specifications such as engine torque and payload, while they are less demanding with respect to style. This causes a lower emphasis on product cost and optimisation of hardware than in the automotive industry in general. The lower volumes also make the manufacturers more willing to design variants to meet the requests of a small number of customers.

However, the segment of heavy vehicles is not homogeneous with respect to software and electronics development practices. For instance, the industrial partners in this paper face quite different market situations and hence employ different development techniques:

- CC Systems (CCS) is developing and supplying advanced distributed embedded real-time control systems with focus on mobile applications. CCS develops both hardware and software for forest harvesters, rock drilling equipment and combat vehicles. The systems are built to endure rough environments and are characterised by safety criticality, high functionality, and high requirements on robustness and availability.

- 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 these products have in common is that they demand high reliability control systems that are maintainable and still cheap to produce. The systems are classified as distributed embedded real-time systems which must perform in an environment with limited hardware resources.

3 Research Method

The purpose of this study is to find out what heavy vehicle developers require from a component technology. This includes what is required in terms of a component technology to improve quality, cost or function.

Our approach has been to cooperate with our industrial partners very closely, both by performing interviews and by participating in projects. In doing so, we have extracted the most important requirements on a component-based technique from the developers of heavy vehicles point of view.

3.1 Case-Study 1: Requirements Capturing

The goal of this study was to extract all challenges of relevance when introducing a component technology, and find important requirements. It seems natural to seek answers where the requirements are defined, at the automotive software developing organisations. Secondly, the answers are likely qualitative with a context full of details from development setting, products, organisation etc. These two facts led us to perform a case study [9] for the two cases represented by two developing organizations.

According to [9] a case study is an empirical inquiry that investigates a contemporary phenomenon in its real life context and copes with situations where there are more variables of interest than data points. In this study the phenomenon is the reluctance to adopt a component technology in automotive development and thereby the requirements put on such a technology. It is clearly a contemporary phenomenon and the situation in a development organisation comprises many variables with no hope of sampling enough data points to map relations.

The case-study was performed at Volvo Construction Equipment and at CC Systems. The respondents were senior technical staff from different parts of the organisation like project managers, development process specialists, programmers, and testing specialists. The case-study protocol questions were open ended to encourage respondents to report on any issues they might attribute to component technologies.

We base most of our results on interviews with senior technical staff at the two companies involved in this paper, but we have also conducted interviews with technical staff at other companies. Furthermore, since the embedded systems market is so diversified we have limited our study to applications for distributed embedded real-time control in safety-critical environments, specifically studying companies within the heavy vehicles market segment.

3.2 Study 2: Requirements Grading

The first case study identified many areas of interests and many were closely related to the development process. Open ended discussions gave us the elicitation of the most important requirements but no notion of relative importance can be analysed based on these results. In order to grade requirements according to importance we performed a second study.

The requirement grading was performed in a workshop with a short presentation, definition of terms, questions and a numerical grading of requirements where the average sum was bounded. Thus, respondents could not grade all requirements high in order to get a sum average in the predefined range. The procedure were the following:

1. The workshop started with a short presentation of the study and of component technologies basics. A very brief background was presented with PC software benefits while automotive software engineers are still reluctant. Furthermore the development process of working with components in a component repository rather than developing in a normal V model was described. The terms; Tool, Components, Platform, Component Framework and Repository was explained. Finally the results from the earlier study were presented.
2. Secondly, the definitions of all the requirements that were to be graded were presented and respondents were given handouts with the definitions. Respondents were allowed to ask questions on the definitions.
3. The data collection was made by the respondents filling in a spreadsheet form on a laptop computer where all the twelve listed requirements were to be graded with a number 1-4 indicating from "interesting" to "absolutely decisive". The respondents were to make sure that the sum average of all their grades was in the range 2.4 - 2.6. The sum, average of grades, was shown and recalculated throughout the grading.

4 Requirements

The requirements presented in this section are the result from the first case-study. The requirements are divided in two main groups, the technical requirements (Section 4.1) and the development process related requirements (Section 4.2). Also, in Section 4.3 we present some implied (or derived) requirements, i.e. requirements that we have synthesised from the requirements in sections 4.1 and 4.2, but that are not explicit requirements from industry.

4.1 Technical Requirements

The technical requirements describe the needs and desires that our industrial partners have regarding the technically related aspects and properties of a component technology.

4.1.1 Analysable

The vehicular industry strives for better analyses of computer system behaviour in general. This striving naturally affects requirements placed on a component model. System analysis is considered important with respect to extra-functional system properties, such as the timing behaviour and the memory consumption.

When analysing a system, built from well-tested components the main issue is associated with composability. It must be possible to reason about the systems functionality and extra-functional properties such as reliability and timing characteristics with a compositional strategy, i.e., predict the system properties based on component properties and the components connection logics [1].

4.1.2 Testable and Debuggable

Testing and debugging is by far the most commonly used technique to verify software systems functionality. Testing is a very important complement to analysis and it should not be compromised when introducing a component technology.

In fact, the ability to test embedded-system software at component level can be improved when using CBSE. This is possible because the component functionality can be tested in isolation and this property is a desired functionality according to our industrial partners. Component test should be used before the system tests, and this approach can help finding functional errors and source code bugs at the earliest possible opportunity.

4.1.3 Portable

The components and the infrastructure surrounding them should be platform independent to the highest degree possible. In this context platform independent means hardware independent, OS independent and communication technology independent.

Components are kept portable by minimising the number of dependencies to system specific resources and design decisions. Such dependencies are off course necessary to construct an executable system; however the dependencies should be kept to a minimum.

4.1.4 Resource Constrained

The components should be small and light-weighted, the components infrastructure and framework should be minimised. Ideally there should be no run-time overhead compared to not using a CBSE approach.

Systems are resource constrained to lower the production cost and thereby increased profit. When companies design new ECUs future profit is the main concern. Therefore the hardware is dimensioned for anticipated use but not more.

4.1.5 Component Modelling

A component technology should be based on a standard modelling language like UML [7] or UML 2.0 [6]. The main reason for choosing UML is that it is a well known and thoroughly tested modelling technique with tools and formats supported by third-party developers.

The reason for our industrial partners to have specific demands in these details is that it is believed that the business segment of heavy vehicles does not have the possibility do develop their own standards and practices. Instead they preferably relay on the use of simple and mature techniques supported by a wealth of third party suppliers.

4.1.6 Computational Model

Components should preferably be passive, i.e. they should not contain their own threads of execution. A view where components are allocated to threads during component assembly is preferred. This is believed to enhance reusability and to limit resource consumption. The computational model should be focused on a pipe-and-filter model, partly due to the well known ability to schedule and analyse this model off-line. Also, the pipes-and-filters model is a good conceptual model for control applications.

4.2 Development Requirements

When discussing CBSE requirements the research community often overlooks requirements related to the development process. For software developing companies these requirements are at least as important as the technical requirements. When talking to industry earning money is the main focus. This cannot be done without having an efficient development processes deployed. Hence, to obtain industrial reliance, the development requirements need to be considered and addressed by the component technology.

4.2.1 Introducible

It should be possible for companies to gradually migrate into a new development technology. It is important to make the change in technology as safe and inexpensive as possible.

Revolutionary changes in the development technique used at a company are associated with high risks and costs. Therefore a new technology should be possible to divide into smaller parts, which can be introduced separately. For instance, if a layered software architecture (e.g. as described in [5]) is used, the components can be used for application development only and independently of the real-time operating system. Or, the infrastructure can be developed using components, while the application is still monolithic.

4.2.2 Reusable

Components should be reusable for use in new applications or environments than those for which they where originally designed [2]. The requirement of reusability can be considered both a technical and a development process related requirement. Development process related since it has to deal with aspects like version and variant management, initial risks and cost when building up a component repository, etc. Technical since it is related to aspects such as, how to design the components with respect to the RTOS and HW communication, etc.

Experiences from trying to reuse software components show that reuse is very hard and initially related with high risks and large overheads, and even more complex to build reusable real-time components for embedded systems [1].

4.2.3 Maintainable

The components should be easy to change and maintain, meaning that developers that are about to change a component need to understand the full impact of the proposed change. Thus, not only knowledge about component interfaces and their expected behaviour is needed but also information about current deployment contexts may be needed in order not to break existing systems where the component is used.

In essence, this requirement is a product of the previous requirement on reusability. The flip-side of reusability is that the ability to reuse and reconfigure the components using parameters leads to an abundance of different configurations used in different vehicles. The same type of vehicle may use different software settings and even different component or software versions. So, by introducing reuse we introduce more administrative work and configuration management.

The maintainability requirement also includes sufficient tools supporting the service of the delivered vehicles. These tools need to be component aware and handle error diagnostics from components and support for updating software components.

4.2.4 Understandable

The component technology and the systems constructed using it should be easy to understand. This should also include making the technology easy and intuitive to use in a development project.

The reason for this requirement is to simplify evaluation and verification both on the system level and on the component level. Also, focusing on an understandable model makes the development process faster and it is likely that there will be fewer bugs.

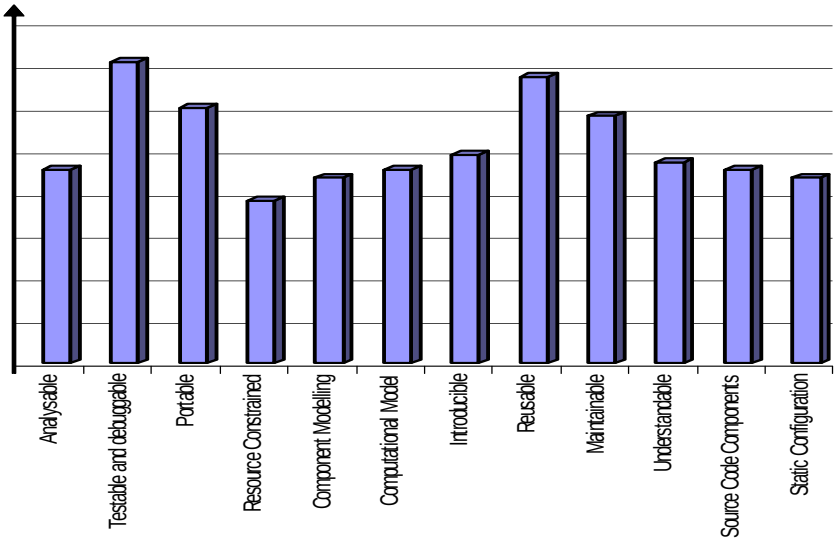

Figure 1. Requirements grades

4.3 Derived Requirements

Here, we present two implied requirements, i.e. requirements that we have synthesised from the requirements in Section 4.1 and 4.2, but that are not explicit requirements from the vehicular industry.

4.3.1 Source Code Components

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 (Section 4.1.2). Since source code debugging is demanded, even if a component technology is used, black box components is undesirable.

Using black-box components would, regarding to our industrial partners, lead to a feeling of not having control over the system behaviour. However, the possibility to look into the components does not necessary mean that you are allowed to modify them. In that sense, a glass-box component model is sufficient.

Source code components also leaves room for compile-time optimisations of components, e.g., stripping away functionality of a component that is not used in a particular application. Hence, source code components will contribute to lower resource consumption (Section 4.1.4).

4.3.2 Static Configuration

For a component model to better support the technical requirements of analysability (Section 4.1.1), testability (Section 4.1.2), and light-weightiness (Section 4.1.4), the component model should be configured pre-runtime, i.e. at compile time. Component technologies for use in the PC domain usually focus on a dynamic behaviour [4, 8]. This is of course appropriate in this specific domain, where powerful computers are used. Embedded systems, however, face another reality - with resource constrained ECU's running complex, dependable, and safety critical control applications. Static configuration should also improve the development process related requirement of understandability (Section 4.2.4), since there will be no complex run-time reconfigurations.

5 Requirement Grades

In this section we present the results (see Figure 1) from the second study, i.e. the industry grading of the requirements in section 4. We present the result by first discussing the requirements separately, and then in section 5.1 we draw same general conclusions from our work.

Analysable

Analysability is in general considered to be important, but the results from our case-study expose that it is not amongst the most important issues of component-based development. For example, it is worth noticing that our partners consider testability and the means to debug the application as much more important. Reasons for this might be that the business segment of heavy vehicles has low series (compared to, e.g., trucks or passenger cars) and that is cheaper to add extra processing power (faster CPU and more memory) in order to avoid timing or memory problems. It may be that a common view amongst industrial developers that analysability is complex and that it leads to a lot of manual information managing. Perhaps timing and memory consumption is not a problem in today's applications whereas testability gives direct feedback to the software developer

and might hence be seen as more important. Yet another reason might be that analysability is not believed to be feasible or practical for distributed and complex industrial systems.

Test and Debug

Test and debug is the most important quality attribute seen in the requirement grades (see figure 1). This is most likely due to the fact that testing of embedded systems is extremely time consuming today. Hence, from a company perspective - there is a huge amount of time (and money) to save if a component technology could decrease the time it takes to verify software functionality.

Another important issue is the rising requirement from Original Equipment Manufacturers (OEMs) that subcontractors deliver "error-free" software. Late or erroneous deliveries are typically punished by an OEM fine. This entail that testing of software (typically not complete systems but rather components) of the system gets more and more important.

It is also worth noticing that both CCS and VCE have spent huge amounts of money on developing test and debug equipment for their respective systems. Hence, the results might be a bit biased, i.e., that these companies consider it more important than the typical embedded software developer.

Portability

Portability is considered very important, mainly due to the fact that it is desired to keep hardware upgrading costs to an absolute minimum. But it is of course also important to be flexible in the choice of software platform.

For CCS, working with many different OEMs (and many different platforms), the requirements of portability is obvious - but it is striking to see that also VCE consider portability as being very important (see Figure 2). The reason for this is essentially that it is very important not to be too dependent on tool vendors and hardware platforms.

Resource Constrained

Surprisingly, and in quite contrary to what one could expect from developers of resource constrained embedded systems, this requirements is considered to be the least important in this study. The reason for this might be the fact that current state-of-practise development methods used by the vehicular industry are rather resource constrained. Hence, there is not much focus on this requirement in the daily work. It might be the case that developers take things they have for granted, and see things they do not have.

Another reason is Moore's law , it is cheaper to by more processing power than it is to spend money on analysing timing and memory consumption. This is also dependent on the product volumes, for low series products it might be worth spending some extra money on hardware in order to facilitate the use of more advanced development methods.

Component Modelling

This requirement is not considered to be very important; meaning that other aspects of modelling is more important than using business standards. For example, simplicity is more important than using a standard modelling language. However, it is interesting to notice that the requirement on using a standardised modelling language is more important relative to the requirement on resource usage.

Computational Model

The requirement on the computational model, meaning that the components should be passive (not having their own threads of execution) and that pipe-and-filter should be used as an architectural pattern, is the most deviating requirement (see Figure 2). This might be because VCE is currently using the Rubus Component Model [3] using a pipe-and-filter architecture, whilst CCS use different architectural patterns in different applications.

Introducible

It is considered relatively important that the component technology is easy to introduce in new and existing projects/products. This requirement also includes the possibility to use parts of a component technology, e.g., together with various operating systems depending on customer needs.

One would expect to see a certain difference between a sub-contractor and an OEM - but as can be seen in Figure 2 both companies agree on the relative importance of this requirement.

Reusable

It is very interesting to see that reusability which is one of the fundamental reasons for moving towards CBSE is considered to be the second most important overall requirement. The reason for this is likely the large potential of software reuse in terms of development time and cost.

Reusability is typically considered to be very demanding, so it is worth noticing that the companies are willing to spend the extra money on more processing power (low emphasis on the requirement of resource usage) in order to facilitate reusability.

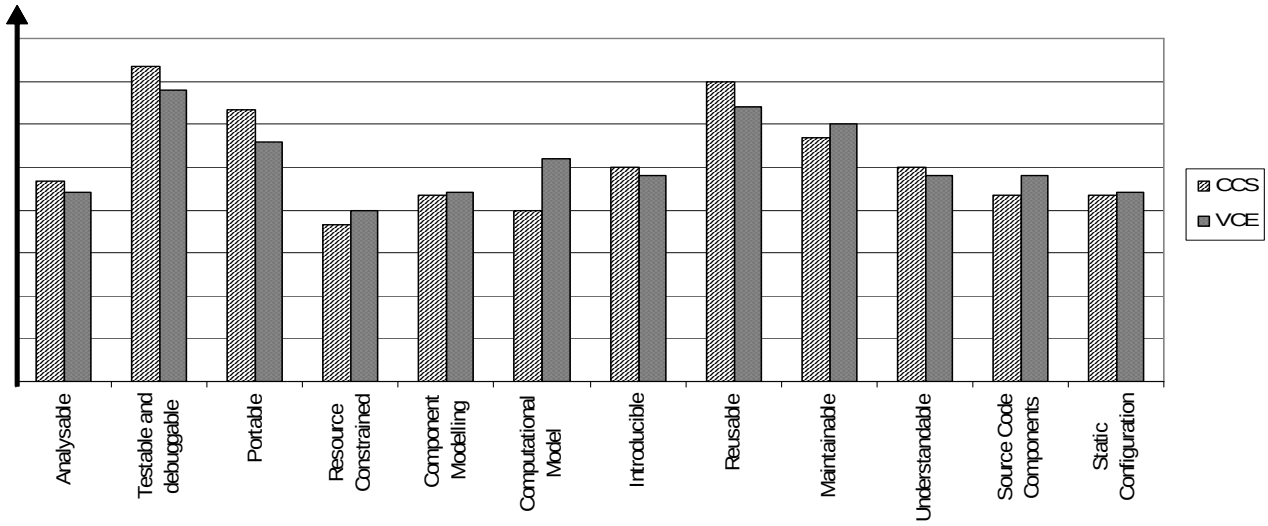

Figure 2. Requirements from the two companies

Maintainability

Maintainability is ranked as the third most important requirement. The reason for this is most likely the high costs that arise when upgrading or updating software. Support for software configuration management is considered a prerequisite in order to facilitate cross platform and product reuse, and hence these requirements are tightly coupled. Also, updating existing software by replacing erroneous software components requires efficient tool support.

Understandable

Understandability is not a primary requirement. This means that the companies are willing to spend some money on training personnel in software development in order to reach primary goals like reusability, portability and testability.

Source Code and Static Configuration

Not much focus is spent on the derived requirements. These requirements should perhaps not be compared with the other requirement since they are tightly coupled to primary requirements. This is rather to be seen as means to reach other requirements. For example, it is not possible to debug the application source code if the software components are delivered in a binary format.

This might be considered a weakness of the study, but we include the results for consistency reasons.

5.1 Discussion

It is interesting to see that the basic properties of CBSE (e.g. reusability, maintainability, and portability) are highly valued by industry. This might be biased due to the fact that this case-study deals with component-based development. However, the relative importances between the listed requirements are obvious and should be seen as a driver for component-based software.

Also, it is interesting to see that the results from the two companies (see Figure 2) correspond with each other very well. Bearing in mind that the two companies represent two different types of control system developers, OEM and subcontractor, these similarities are even more striking.

Another interesting conclusion from this case-study is that the development process related requirements (i.e. introducible, reusable, maintainable, and understandable) is considered to be substantially more important then the technical requirements. Hence, the research community should not overlook these problems but rather spend more focus on issues like, e.g., support for software configuration management.

6 Conclusions

We conclude that using software components and component-based development is seen as a promising to address challenges in product development, including integration, flexible configuration as well as support for software reuse.

The main contribution is that we show the relative importance of industrial requirements, in addition to the industrial requirements on a component technology for use in automotive applications. We describe and grade requirements on a component technology as elicited from two Swedish control-system developers. The requirements are divided into two main groups, the technical requirements and the development process related requirements. The reason for this is to clarify that the industrial actors are not only in-

terested in technical solutions, but also in improvements regarding their development process.

The result can be used to guide modifications and/or extensions to existing component technologies in order to make them better suited for industrial deployment. The results can also serve as a platform for software engineering research, since researchers can be guided to put focus on the most desired areas within component-based software engineering.

In future work we plan a continuation of this study with more companies involved within the domain of heavy vehicles. Another possibility would be to involve other segments within the automotive domain, e.g., car manufacturers, and explore differences.

Acknowledgements

We would like to thank CC Systems and Volvo Construction Equipment for their support and interest in this case-study.

References

[1] I. Crnkovic and M. Larsson. *Building Reliable Component-Based Software Systems*. Artech House publisher, 2002. ISBN 1-58053-327-2.

[2] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In *Proceedings of the 17^{th} International Conference on Software Engineering*, April 1995. Seattle, USA.

[3] K.L. Lundbäck, J. Lundbäck and M. Lindberg. Component-Based Development of Dependable Real-Time Applications. In *Real-Time in Sweden – Presentation of Component-Based Software Development Based on the Rubus concept, Arcticus Systems: http://www.arcticus.se*, August 2003. Västerås, Sweden.

[4] Microsoft Component Technologies. COM/DCOM/.NET. http://www.microsoft.com.

[5] A. Möller, J. Fröberg, and M. Nolin. Industrial Requirements on Component Technologies for Embedded Systems. In *Proceedings of the 7^{th} International Symposium on Component-Based Software Engineering*, May 2004. Edinburgh, Scotland.

[6] Object Management Group. UML 2.0 Superstructure Specification, The OMG Final Adopted Specification, 2003. http://-www.omg.com/uml/.

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

[8] Sun Microsystems. Enterprise Java Beans Technology. http://java.sun.com/products/ejb/.

[9] R. Yin. *Case Study Research – Design and Methods*. Applied Social Research Methods Series, Volume 5, SAGE Publications, 2003. ISBN 0-7619-2553-8.