Achieve consistent mappings between component models and real-time models

– Licentiate Thesis Proposal –

Johan Fredriksson

Mälardalen Real-Time Research Centre
Department of Computer Science and Engineering
Mälardalen University, Västerås, Sweden

johan.fredriksson@mdh.se

Abstract. *To manage the increasing complexity of embedded systems, the industry is constantly looking for new software development strategies. Today, component-based development is a hot area within both industrial and academic domains. Although the abstraction level of component models increases, they do not express the timing properties that are used in timing analysis. For this reason a mapping between component models and real-time execution models are necessary. This report is a licentiate proposal that discusses, foremost, the issues of transforming components and component architectures into run-time tasks.*

1. Introduction

When software systems are built there are many paradigms that can be used, regular imperative programming, object oriented programming, declarative programming etc. One paradigm that has shown to be useful for large desktop and high-end system is component based development. *Component-Based Development (CBD)* is the idea of reusing existing components, much in the same way as standard components are used in other engineering disciplines[14,15], and has been around since at least 1968 [16]. CBSE has become a widespread practice not least through de facto standards like COM, CORBA, Enterprise Java Beans, and .NET. These de facto standards however are very flexible and computation demanding, and are hence not very suitable for a large range of software systems, e.g., resource constrains systems or real-time systems. Component models for resource constrains systems must deal with more requirements besides the *functional*. These additional requirements are often referred to as *non-functional* or *extra-functional* properties. To manage the increasing complexity of embedded systems, the industry is constantly looking for new software development strategies.

This report is a licentiate proposal that discusses, foremost, the issues of transforming components and component architectures into run-time tasks within the frames of the SAVE project [10]. The main goal of the SAVE project is to establish an engineering discipline for systematic development of component based software for safety critical embedded systems. This will be vital to the Swedish industry, and paves the way for establishing an industry for safety-critical and other components. The main innovation of SAVE is the interdisciplinary combination of architectural and component based design with analysis, and verification, in the specific context of safety and real-time.

The licentiate thesis is a part of the PhD education, and the purpose is to investigate interesting and relevant research areas for an upcoming PhD dissertation. The goal of the licentiate thesis is to chart a specific area in which succeeding research is to be made. This licentiate thesis proposal focuses on the specific area of component to task mappings and discusses adjacent areas.

2. Background and Motivation

Historically, the development of embedded systems is being done using only low level programming languages, to guarantee full control over the system behaviour. As the complexity and the amount of functionality implemented by software increase, so does the cost for software development. Also, since product lines are common within the domain, issues of commonality and reuse are central for reducing cost as well as increasing reliability. Hence component-based development can be an efficient and promising approach for software development, enabling well defined software architectures as well as reuse.

The automotive domain is an example of industry that is using component-based development (CBD) in practice. The components are not software components, but rather electronic control units (ECUs). A modern car have between ten and hundred ECUs, each responsible for a function in the vehicle. The ECUs are developed both in house and by third parties, thus vehicle manufacturer tend to become system integrators, integrating in house and third party ECUs. Because of scares resources regarding, e.g., computation power, memory consumption and energy consumption the software has requirements on timeliness, memory consumption, energy consumption etc. Besides this, the software has requirements on safety and reliability.

CBSE has become a widespread practice not least through de facto standards like COM, CORBA, Enterprise Java Beans and .NET. These de facto standards are very flexible and computation demanding and are hence not very suitable for resource constrains systems. Software components often have properties that make them hard to analyze due to complexity and black box properties. However, there are component technologies that have considered different properties to better conform to resource constrains systems with timing requirements. For instance [1,7] and [9,18] are examples of such industrial component technologies, while [4,5] and [12] are academic.

3. Research Description

Although component based development is a well established engineering practice it is relatively young and unexplored in the context of computer software. There are still no real standards for Component-Based Software Engineering (CBSE). However, a few de-facto standards are present; still, they are aimed at desktop application development. The component-based approach has shown to be effective both in means of lowering development times, and reducing the number of errors [13]. However, embedded systems tackle even more problems in form of scares resources, timing requirements, memory requirements, etc [17]. In the real-time and embedded systems domains there are many theories, methods and tools that provides solutions to these *extra-functional requirements*. These methods use a number of system properties, such as worst-case execution time (wcet), execution period, deadlines, jitter, etc., and terms such as tasks and threads, to reason about timing and other requirements. However, the notion of components is rarely used. On the other hand component technologies usually do not include extra-functional properties. To be able to use component approach, which improves the development process, and at the same time guarantee system behaviour it is important how components map to run-time tasks. This problem is briefly discussed in e.g. [19].

The goal of this work is to develop methods for mapping component models to run-time execution models, regarding extra-functional properties.

3.1 Methodology

There are few companies in the embedded and real-time domain that use component-based development, and even fewer companies that use components to tasks allocation in an ordered fashion. Hence it is hard to find a current industrial setting of the stated research description. Moreover, little research has been performed on the specific area of component to task allocation, thus this research is mainly built on experience of senior researchers and previous related work. The industrial relevance of the research is investigated with an *inquiry* on a set of companies within the vehicular domain, and component mappings will be *simulated* to evaluate the prospects. The research methodology is depicted in Figure 1.

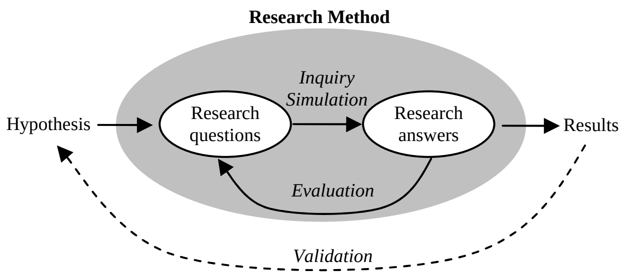

Figure 1 Research method used for validating the hypothesis

3.2 Hypothesis

Given that CBSE has improved software development of desktop applications, we make the assumption that CBSE can be used for embedded and real-time systems as well. From this assumption we form a hypothesis that says, firstly, that component based development is useful for managing the increasing complexity of embedded systems. Developers of embedded systems must handle the problems with scares resources and performance requirements. Hence, the hypothesis says, secondly, that an efficient mapping of components to run-time tasks can provide an efficient resource allocation.

The concepts of component mapping may be used for achieving consistent mappings between component models and real-time models. *(H)*

The hypothesis is very vague and general, which makes it almost is impossible to falsify; on the other hand, it is therefore also very hard to confirm it. Although, it is possible to create a number of more specific questions that, if answered, will support the general hypothesis. The questions do not necessary prove the hypothesis, but they will suggest that the research is aimed at an interesting and real problem. The answers to the question will then be validated regarding the general hypothesis. In the next section, three specific research questions are formulated.

3.3 Research Questions

In current component based embedded software development practices the mapping of component services to run-time threads (tasks) is a problem [19]. Most embedded systems has real-time requirements, hence it is vital that the mapping considers temporal attributes, such as *worst case execution time* (WCET), deadline (D) and period time (T). In a system with many small component services, the overhead from context switches will be quite high. Embedded real-time systems consist of periodic and sporadic events and they usually have end-to-end timing requirements. Periodic events can often be coordinated and executed by the same task, while preserving temporal constraints. Another problem is that embedded systems often have very scares resources. Hence, there are many trade-offs to be made when allocating component services to tasks. From this we state the following question:

How can components in embedded component-based systems be allocated to operating systems tasks for high utilization of hardware while preserving timeliness? *(Q1)*

Components can be allocated to tasks in a variety of ways. One of the more intuitive ways of allocating is a one-to-one mapping where each component is allocated to a separate task. However, this allocatin will give high overhead regarding task switches and memory usage. Hence an allocation where several components are allocated to the same task is desired. However, often different restrictions and requirements on different components, such as period time, jitter or response times, prevents components to be allocated in some ways. There may be one or several ways that can be found to allocate tasks with respect to different requirements. In our work we suggest a framework that utilizes genetic algorithms to find and evaluate allocations.

Except functional behaviour, technologies for embedded and real-time systems usually define a set of extra-functional properties. The extra-functional properties are properties that define, e.g., timing behaviour or memory behaviour. These properties are for analysis purposes only and have no functional value. There may be properties required for performing and analyzing a component mapping, thus a second question is raised:

What properties must be managed by a component technology in order to analyze mappings from components to run-time tasks? *(Q2)*

To be able to perform evaluations with our proposed framework, a component needs some properties on which the evaluation can be made. There are also a set of properties needed to fulfil basic requirements on timeliness and security. What properties are needed is highly dependent on system type. In this research we are focusing on embedded safety-critical real-time systems. In order to find the extra-functional properties needed in this domain we have performed an extensive survey together with five companies that operate in the specified domain. These properties are then validated, and methods for validating them regarding a component mapping are described.

Component technologies have domain specific extra-functional properties that may be adjacent or contradictory. Naturally, when properties are contradictory, trade-offs have to be made. These properties have to be carefully considered when mapping several components with different properties to the same run-time task

How are system properties affected by mapping several components to the same run-time task *(Q3)*

When mapping several components to one task, several system properties are affected. Some properties may be contradictory, hence trade-offs may be necessary. This means that mapping that is optimized regarding one property may be invalid, or bad regarding another property. It is important to understand these trade-offs in order to consider as many properties as possible.

3.4 Discussion & Contributions

The answers to the research questions will be based on publications, six papers of which two are published, paper A and paper C. Preliminary answers to the research questions posed above are presented briefly below.

How the concepts of component-based development, can be, and is used in the domain of embedded systems is addressed in papers A, B, and C, and in report A. Existing technologies are evaluated in an industrial setting. General requirements on component models for embedded systems are discussed, and new a technology for embedded systems is proposed. In paper D a task allocation strategy is proposed and simulation and evaluation of the mapping theories presented.

The contributions of the work will be:

1. *Evaluation of how the concepts of Component-Based Software Engineering are used for developing embedded systems, and a proposal of how a new software component technology can be used for developing embedded systems.*
2. *Evaluation of how component to task allocation can be used for increasing performance and system utilization, and a proposition for an evaluation framework for such allocations.*
3. *Concrete proposal of how component to task mappings can be integrated in a component technology and used to support development of component-based embedded systems.*

4. Research Focus

The research questions comprise summaries of the research activities that we have focused on in our work. This section briefly discusses this work and related areas.

4.1 Component model requirements for real-time mapping

In order to perform mapping from components to run-time tasks, there must be a notion of measurement to evaluate each mapping. Therefore, a component model should exploit non-functional properties. A run-time mapping can be performed considering one or several properties with different importance. Dependent on how the mapping is performed, system properties are affected differently. There may be undesired or unpredicted side effects from a mapping. Hence, trade-offs may be required to minimize undesirable effects, while maximizing the desired effects. We address which properties are needed in order to perform a mapping from components to real-time tasks. Further we discuss trade-offs to be considered. Moreover, we are analyzing the requirements from industry, within the vehicular domain, to be considered in a component model.

We propose basic characteristics of a component technology that can be used for expressing, e.g., real-time properties. High level timing requirements are set on transactions, and tasks must be assigned attributes in a way that these requirements are kept. Attributes that are assigned tasks are worst case execution time, period for periodic tasks and minimum inter-arrival time for event triggered tasks. From these attributes, different priority assignment approached can be used, e.g., Bate and Burns [20]. When priorities are assigned, ordinary real-time analysis can be performed to evaluate schedulability of the systems. An approach for synthesising code for the run-time system is proposed. This includes mapping the tasks to operating system specific tasks, mapping data connections to an OS, generating glue code, compiling and linking; all in an automated fashion.

The appropriateness of the proposals is evaluated with an inquiry to industry within the vehicular domain.

4.2 Mapping Embedded Component Based Systems to Target Platforms

Component based systems often have very simple strategies for allocating components to run-time entities. In many component models there is no clear strategy for providing this mapping. Very often this part is left outside the component technology which causes a gap between high-level design and concrete implementation. A common approach is to map each component to a run-time task. Another common approach is to map all components to one task. Both these strategies have drawbacks in form of memory consumption or flexibility. By analyzing a composition of components it is possible to find a relation between components and run-time tasks that fulfils a set of requirements on, e.g., memory consumption and CPU overhead.

We address strategies for allocating components to run-time tasks and how the mappings affect the system. We have developed an evaluation framework for evaluating mappings between component services and tasks. The evaluation framework is a set of models for calculating properties, together with schedulability analysis, which are used with an optimization algorithm to find a feasible allocation that fulfils given requirements on, e.g., memory and performance. The framework is implemented and evaluated, with genetic algorithms, on a proposed component model. The proposed component model is applicable to a large set of commercial or research embedded component models, e.g., koala[9], pecos[7] and autocomp[21].

As future work we will investigate the effect of more complex real-time properties such as jitter and blocking. To evaluate the work the current implement is extended and used for evaluating the work. Further we will investigate more optimization properties and extend the component model to be feasible for a larger set of commercial and research component models.

5. Proposal to outline of the thesis

The preliminary outline is listed below. Section 1 is

1 Introduction ~ 5 pages

1.1 Research description

1.2 Contribution

2 Background ~20-25 pages

2.1 Component based Software Engineering

2.2 Report A, A Sample of Component Technologies for Embedded Systems. Technical Report, Mälardalen Real-Time Research Centre Department of Computer Science and Engineering, Mälardalen University, Västerås, Sweden. Mikael Åkerholm and Johan Fredriksson

2.3 Component to Task Mapping

3 Task allocation for Embedded Component Based Systems

Paper A, Evaluation of Component Technologies with Respect to Industrial Requirements. In Euromicro Conference, Component Based Software Engineering Track (ECBSE 2004), 2004. Anders Möller, Mikael Åkerholm, Johan Fredriksson, Mikael Nolin

Paper B, Importance of Quality Attributes for Software Embedded in Vehicles, and Implications for a Component Technology, ongoing work

Paper C, Introducing a Component Technology for Safety Critical Embedded Real-Time Systems, In International Symposium on Component-based Software Engineering (CBSE7) Edinburgh, Scotland, May 2004. Springer Verlag, Kristian Sandström, Johan Fredriksson, Mikael Åkerholm
Paper D, Calculating Resource Trade-offs when Mapping Component Services to Real-Time Tasks. Ongoing work. Johan Fredriksson, Kristian Sandström, Mikael Åkerholm

Paper E, System properties and trade-offs when mapping component services to real-time tasks. Starting-up work. Johan Fredriksson, Kristian Sandström, Mikael Åkerholm

4 Conclusions and Future Work

5.1 Abstracts and my contributions:

Report A

In this technical report we briefly describe several component technologies for embedded systems. The focus is broad; we are addressing all from development tools to run-time mechanisms and for each case study we will focus on the most distinguishing parts.

Title: A Sample of Component Technologies for Embedded Systems

Author(s): Johan Fredriksson, Mikael Åkerholm

Published: Technical report

My Part: The work has been evenly distributed among the two authors.

Paper A

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.

Title: Evaluation of Component Technologies with Respect to Industrial Requirements

Published: In Euromicro Conference, Component Based Software Engineering Track (ECBSE 2004), 2004.
Author(s): Anders Möller, Mikael Åkerholm, Johan Fredriksson, Mikael Nolin

My part: The evaluation work and documentation of the results have been equally distributed between all the authors

Paper B

In this research, we provide a vehicular domain specific classification of the importance of different quality attributes for software, and a discussion of how they could be facilitated by a component technology. The basis is a list of quality attributes, where the importance of each of the attributes has been ranked by representatives for different companies within the vehicular domain.

Title: Importance of Quality Attributes for Software Embedded in Vehicles, and Implications for a Component Technology

Published: Ongoing work

Author(s): Mikael Åkerholm, Johan Fredriksson, Kristian Sandström, Ivica Cronkovic

My Part: I intend to participate in all parts of the work, with focus on the classification of attributes considering my research focus.

Paper C

Safety critical embedded real-time systems represent a class of systems that has attracted relatively little attention in research addressing component based software engineering. Hence, the most widely spread component technologies are not used for resource constrained safety critical real-time systems. They are simply to resource demanding, to complex and to unpredictable. In this paper we show how to use component based software engineering for low footprint systems with very high demands on safe and reliable behaviour. The key concept is to provide expressive design time models and yet resource effective runtime models by statically resolve resource usage and timing by powerful compile time techniques. This results in a component technology for resource effective and temporally verified mapping of a component model to a commercial real-time operating system.

Title: Introducing a Component Technology for Safety Critical Embedded Real-Time Systems.

Published: In International Symposium on Component-based Software Engineering (CBSE7) Edinburgh, Scotland, May 2004. Springer Verlag.

Author(s): Kristian Sandström, Johan Fredriksson, Mikael Åkerholm

My Part: I have been author of the parts concerning attribute assignment and synthesis. I have also taken part in the overall shaping of the method and paper.

Paper D

The issue of allocating component services to schedulable task entities has gained little focus, even though component based development has attracted an increasingly interest, also in the real-time community. Trade-offs when allocating component services to tasks, are, e.g., cpu-overhead, footprint and integrity. In this paper we introduce general framework for calculating properties, such as memory consumption and cpu-overhead, of a given mapping of component services to tasks. We show the effect of the framework by considering both memory consumption and cpu overhead on several allocations.

Title: Calculating Resource Trade-offs when Mapping Component Services to Real-Time Tasks.

Published: Submitted to EMSOFT 2004

Author(s): Johan Fredriksson, Kristian Sandström, Mikael Åkerholm

My Part: I have been initiator and author of all parts of the paper. I have been engaged in the shaping of the method and the paper.

Paper E

A run-time mapping can be performed considering one or several properties with different importance. Dependent on how the mapping is performed, system properties are affected differently. There may be undesired or unpredicted side effects from a mapping. Hence, trade-offs may be required to minimize undesirable effects, while maximizing the desired effects.

Title: System properties and trade-offs when mapping component services to real-time tasks

Published: Staring-up work

Author(s): Johan Fredriksson, Kristian Sandström, Mikael Åkerholm

My Part: I am the initiator and author of all parts of the paper.

5.2 Time Plan

To complete the thesis, the main remaining and finished activities are:

Publications

Report A	**95% finished**
Paper A	**published**
Paper B	**20% finished**
Paper C	**published**
Paper D	**90% finished**
Paper E	**0% finished**
Thesis	**0% finished**

A time plan for the remaining work is presented in Figure 2.

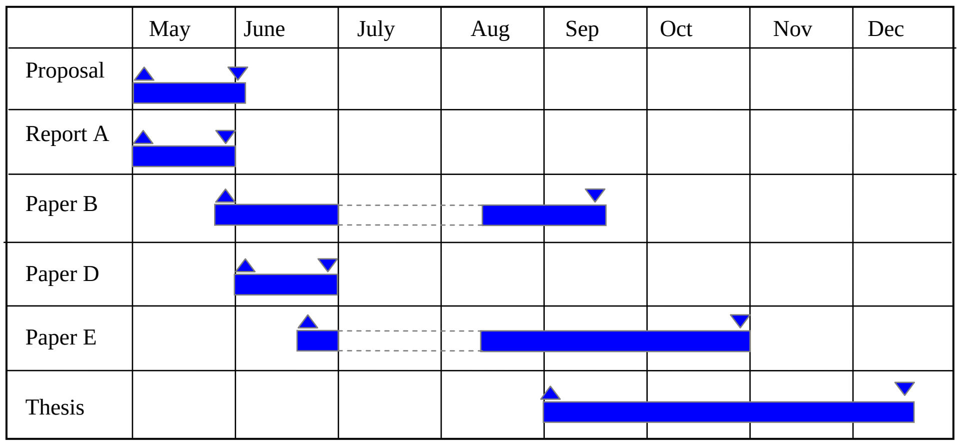

Figure 2 Time plan for papers and thesis

6. Other results

Results not published in the thesis are presented below:

6.1 Courses

The requirement for the licentiate degree is at least 30 points of courses. This requirement is fulfilled, and a summary of finished and pending courses is shown below

Table 1 Credited, finished and pending courses

Software Engineering	5p
SAVE – Component Based Software Engineering for Safety-Critical Embedded Systems	5p
Distributed Development	3p
Component Based Technologies	3p
Component Based Software Engineering	5p
Real-Time Systems- advanced course	5p
Parallel Systems I	2p
Research Methodology	5p
Distributed Real-Time Systems	2p
Safety- Critical Systems	5p
Total completed	**40p**
In progress:	
Embedded Control Systems	5p
Total	**45p**

6.2 Publications not included in the thesis

Conferences and workshops:

- “Software Component Technologies for Real-Time Systems - An Industrial Perspective”, In WiP Session of Real-Time Systems Symposium (RTSS)

Cancun, Mexico, December 2003. Author(s): Anders Möller, Mikael Åkerholm, Johan Fredriksson, Mikael Nolin
- Attaining Flexible Real-Time Systems by Bringing Together Component Technologies and Real-Time Systems Theory, In Proceedings of the 29th Euromicro Conference, Component Based Software Engineering Track Belek, Turkey , September 2003. IEEE, Author(s): Johan Fredriksson, Mikael Åkerholm, Kristian Sandström, Radu Dobrin
- On the Teaching of Distributed Software Development, In 25th International Conference INFORMATION TECHNOLOGY INTERFACES Dubrovnik, Croatia , June 2003. IEEE. Author(s): Ivica Crnkovic, Igor Cavrak, Johan Fredriksson, Rikard Land, Mario Zagar, Mikael Åkerholm

MRTC Reports:
- An Industrial Evaluation of Component Technologies for Embedded-Systems, MRTC Report ISSN 1404-3041 ISRN MDH-MRTC-155/2004-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, February 2004. Author(s): Anders Möller, Mikael Åkerholm, Johan Fredriksson, Mikael Nolin
- Component Based Software Engineering for Embedded Systems - A literature survey, MRTC Report ISSN 1404-3041 ISRN MDH-MRTC-102/2003- 1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, June 2003. Author(s): Mikael Nolin, Johan Fredriksson, Jerker Hammarberg, Joel G Huselius, John Håkansson, Annika Karlsson, Ola Larses, Markus Lindgren, Goran Mustapic, Anders Möller, Thomas Nolte, Jonas Norberg, Dag Nyström, Aleksandra Tesanovic, Mikael Åkerholm

7. References

1. Arcticus Systems Homepage: http://www.arcticus.se/
2. Autosar homepage: http://www.autosar.org/
3. East/EAA homepage: http://www.east-eea.net/
4. H. Hansson, M. Åkerholm, I. Crnkovic, M. Törngren, SaveCCM – a component model for safety-critical real-time systems, Submitted to CMDS
5. M. de Jonge, J. Muskens and M. Chaudron, Scenario-Based Prediction of Run-time Resource Consumption in Component-Based Software Systems, Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction, May, 2003.
6. Microsoft Homepage: http://www.microsoft.com
7. 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
8. OMG Homepage: http://www.omg.org
9. 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

10 SAVE homepage: http://www.mrtc.mdh.se/SAVE

11. Sun Homepage: http://www.sun.com
12. 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.

13. Parastoo Mohagheghi, Reidar Conradi, Ole M. Killi, Henrik Schwarz: An empirical Study of Software Reuse vs. Reliability and Stability, Proceedings of the 26th international conference on software engineering (ICSE 2004), 23-28 May, 2004, Scotland.

14 Szyperski C., Component Software - Beyond Object-Oriented Programming, Addison-Wesley, 1998.

15 Crnkovic I. and Larsson M., Building Reliable Component-Based Software Systems, Artech House, 2002.

16 McIlroy M. D., “Mass Produced Software Components”, In Proceedings of Nato Software Engineering Conference, 1968.

17 Hammer, D. K., Chaudron, M. R. V.: Component-Based Software Engineering for Resource-Constraint Systems: What are the Needs?. 6th Workshop on Object-Oriented Real-Time Dependable Systems, January, Rome, Italy, (2001)

18 Stewart, D. B., Volpe, R. A., Khosla, P. K.: Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects, IEEE Transactions on Software Engineering, December (1997), 759-776

19 Kodase S., Wang S., and Shin K. G., "Transforming structural model to runtime model of embedded software with real-time constraints", In proceeding of Design, Automation and Test in Europe Conference and Exhibition pp. 170-175, 2003.

20 Bate, I., Burns, A.: An Approach to Task Attribute Assignment for Uniprocessor Systems. Proceedings of the 26th Annual International Computer Software and Applications Con-ference, IEEE, (2002)

21 Sandström K., Fredriksson J., Åkerholm M., “Introducing a Component Technology for Safety Critical Embedded Real-Time Systems”, In proceedings of International Symposium on Component-based Software Engineering (CBSE7) Edinburgh, Scotland, May 2004.