A component-based development framework for supporting functional and non-functional analysis in control system design

Johan Fredriksson
Mälardalen University
Mälardalen Real-Time Research Centre
Västerås, Sweden
johan.fredriksson@mdh.se

Abstract

This extended abstract reports on research that show how component-based software engineering may be used for realizing embedded systems; and how component technologies for use in embedded vehicular systems can reduce resource usage without compromising non-functional requirements, such as timeliness.

1 Introduction

During recent decades processors have become more powerful and more cost efficient. They are now of interest in areas where not used before, permitting the development of ever more complex system applications. To manage the increasing complexity of applications, Industry is constantly looking for new software development strategies. One paradigm found to be of use in desktop computing systems is Component-Based Software Engineering (CBSE). However, embedded systems have requirements that are not regarded in desktop applications, such as low memory utilization, low processor overhead and predictability. Many embedded systems are safety-critical because they control applications in our society. If these applications malfunction they can have disastrous consequences. Hence, non-functional characteristics (such as reliability) are very important in these types of applications. One important class of non-functional requirements are real-time requirements. These requirements define within what time a task must be performed.

The notion of components is rarely used in real-time systems. On the other hand, current component technologies usually do not include non-functional properties typical of real-time systems. To be able to use a component approach, which makes the system development process more efficient, and at the same time guarantees system behaviour, both how the component technology uses non-functional properties, and how components are allocated to the run-time systems are important.

The research presented has been focused on four related areas; non-functional properties and the relation to industry, component-based development considering real-time systems, mapping between components and real-time tasks, and run-time techniques for decreasing pessimism of real-time analysis. Section 2 gives a summary of our research. Sections 3 to 7 explains further each of the four parts of our research. For the full licentiate thesis please refer to [1].

2 Research Summary

Component-technologies available today have not been used extensively in the embedded systems domain. To understand why this is the case we have conducted a survey and performed evaluations of the requirements of the vehicular industry with respect to software and software development. The purpose of the evaluation is to provide a foundation for defining models, methods and tools for component-based software engineering.

We propose a component-model designed for resource constrained embedded real-time systems that use powerful compile-time techniques to realize the component-based approach and ensure predictable behaviour.

The main contribution of our research is the implementation and evaluation of a framework for resource-efficient mappings between component-models and real-time systems. We show how efficient mappings can reduce memory usage and CPU-overhead. An implemented framework utilizes genetic algorithms to find feasible, resource efficient mappings. Finally we propose a resource reclaiming strategy for component-based real-time systems in order to decrease the impact of pessimistic execution time predictions.

Along this line there is a growing interest towards standardizations through standard bodies such as the OSEK and AutoSAR comittess. The market for advanced development tools and technologies is still small, though this is indeed the place where much productivity can have gain.

3 Non-functional properties

In order to find out which properties are important for the vehicular industry, a survey was performed on different vehicular companies. Representatives of the companies were requested to prioritize quality attributes (extendibility, maintainability, usability, predictability, security, safety and reliability) regarding importance. An evaluation of existing component technologies with respect to industrial requirements was also conducted. The technologies evaluated were PECT [9], Koala [8], Rubus CM [2, 3], PBO [7], PECOS [10] and CORBA based technologies [4]. The technologies were chosen firstly on the basis that there is enough information available, and secondly that the authors claim that they are suitable for embedded systems. CORBA, however, was chosen as a reference technology to represent the desktop/Internet domain.

The evaluation points out which requirements that are partly fulfilled by a component technology, and which are not. The requirements were gathered from an industrial case-study performed at Volvo Construction Equipment and at CC-Systems.

4 Component models for embedded systems

As described in the previous section, our research has pointed out that there are few component models that support some of the important non-functional attributes. We show how CBSE can be used for embedded real-time systems with high requirements on analyz-

ability and low memory footprint. We consider the non-functional attributes predictability, reliability, safety and usability which are all expressed as important by the vehicular domain. Existing commercial component technologies often have powerful run-time mechanisms to realize the component-based approach, which is a disadvantage in terms of resource utilization for resource constrained systems. The defined component model is based on the pipe-and-filter interaction model and uses a Read-Execute-Write paradigm; all in-ports of a component are read, the component executes and finally writes all its out ports. This execution model has the advantage of being highly analyzable. Moreover, the control systems in vehicles are often suitable for the pipe-and-filter paradigm. End-to-end deadlines are imposed to the model and are augmented with start and completion jitters. A middleware is proposed to handles all communication between the component model and the underlying run-time system.

5 Allocating components to real-time tasks

Many component-based systems today use one-to-one allocations between design-time components and real-time tasks, or other rudimentary allocations. Finding allocations that co-allocate several components to one real-time task leads to better memory and CPU usage. However, the one-to-one allocations have the benefit of being highly analyzable, which is often a strong requirement in embedded systems, especially in embedded systems that handle time-critical functions such as engine control and breaking systems.

Due to the combinatorial explosion of possible allocations from components to tasks, the problem is complex by nature. An allocation from components to a task is evaluated considering schedulability (timeliness) and isolation, where isolation is defined as mutual exclusion of components regarding shared resources or other legitimate engineering reasons. Because the problem is inherently complex the strategy is to evaluate our allocation approach by implementing a framework that utilizes a meta-heuristic search technique

Each allocation is validated with respect to period-times, isolation, end-to-end deadlines and schedulability to ensure that an allocation is feasible. A proposed framework gives the possibility to optimize allocations regarding the properties memory consumption and CPU-overhead to find a resource efficient solution. The results from the evaluation were satisfactory, and we have found that for industrially representative systems memory consumption and CPU-overhead can be decreased by as much as 32% and 48% respectively compared to a one-to-one mapping.

6 Resource Reclaiming

In real-time systems there are often unused resources in terms of CPU-time due to pessimistic predictions. These resources can be used for executing tasks, e.g., more often, or with higher quality (longer time).

We show how component technologies can be extended with multiple services to provide different quality levels depending on residual time in real-time systems. We do this by combining the multiple versions paradigm [6], with the adaptive threshold algorithm [5]. The multiple versions paradigm allows us to have several versions (services) of the same component. In this research the multiple versions is used for the same functionality with different quality; consider, e.g., more or less iterations in a numerical approximation. Each quality level is associated with a value which is accumulated to the system as that quality level is chosen and executed. The Adaptive threshold algorithm allows us to provide a system that strives to maximize the total system value by choosing the appropriate quality level dependent on the residual time of the system. However, the multiple versions and adaptive threshold algorithm generates some extra overhead in the system, both in terms of memory and CPU-time. Thus this approach may not be appropriate for very small systems with extreme requirements on keeping the memory and CPU-overhead low.

7 Conclusion

Component-based development has proven itself to be a promising approach for effectively produce complex systems. In this extended abstract we have presented research that begin to take the component-based appraoch into the embedded by; investigating the needs of the indusrty and providing efficient pre-run-time and run-time mechanisms and tools for deploying component-based systems on resource limited platforms.

8 References

[1] J. Fredriksson. Transformation of component models to real-time models. Technical report, Technology Licentiate Thesis No.47, ISSN 1651-9256, ISBN 91-88834-55-7, Mälardalen Real-Time Reseach Centre, Mälardalen University, 4 2005.

[2] K. L. Lundbäck. Rubus os reference manual. general concepts. arcticus systems: http://www.arcticus.se.

[3] K. L. Lundbäck, J. Lundbäck, and M. Lindberg. Component-based development of dependable real-time applications, 2003.

[4] OMG. The common object request broker: Architecture and specification. Technical report, OMG Formal Documatation (formal/01-02-10), February 2001.

[5] N. A. R. Davis, S. Punnekkat and A. Burns. Flexible scheduling for adaptable real-time systems. *Proceedings of IEEE Real-Time Technology and Applications Symposium*, May 1995.

[6] J. A. Stankovic and K. Ramamritham. What is predictability for real-time systems? *Real-Time Systems*, 2(4):247–254, November 1990.

[7] D. B. Stewart, R. A. Volpe, and P. K. Khosla. Design of dynamically reconfigurable real-time software using port-based objects. In *IEEE Transactions on Software Engineering*, pages 759–776. IEEE, December 1997.

[8] R. van Ommering, F. van der Linden, and J. Kramer. The koala component model for consumer electronics software. In *IEEE Computer*, pages 78–85. IEEE, March 2000.

[9] K. C. Wallnau and J. Ivers. Snapshot of ccl: A language for predictable assembly. Technical report, Software Engineering Institute, Carnegie Mellon University, 2003. CMU/SEI-2003-TN-025.

[10] M. Winter, R. Genssler, A. Christoph, O. Nierstraszn, S. Ducasse, R. Wuyts, G. Arevalo, P. Muller, C. Stich, and B. Schönhage. Components for embedded software . the pecos apporach. In *Proceedings of Second International Workshop on Composition Languages In conjunction with 16th European Conference on Object-Oriented Programming (ECOOP) Malaga, Spain*, June 2002.