# Towards Cycle-Accurate Performance Predictions for Real-Time Embedded Systems

Konstantinos Triantafyllidis, Egor Bondarev, Peter H.N. de With
Eindhoven University of Technology
5600 MB, Eindhoven, The Netherlands
{k.triantafyllidis, e.bondarev, p.h.n.de.with}@tue.nl

***Abstract*— In this paper we present a model-based performance analysis method for component-based real-time systems, featuring cycle-accurate predictions of latencies and enhanced system robustness. The method incorporates the following phases: (a) instruction-level profiling of SW components, (b) modeling the obtained performance metrics in MARTE-compatible models, (c) generation, schedulability analysis and simulation of a system model, (d) architecture improvement based on the analysis results. Our proposed method incorporates both the schedulability analysis and the simulation technique, complementing the advantages and eliminating the limitations of the individual steps. Moreover, the cycle-accurate performance metrics initiated by our method lead to accurate performance predictions for an autonomous navigation robot system, with only 6% deviation (or less) from the actual performance metrics.**

Component-based development has become an adopted practice in the real-time systems domain, since it enables rapid system prototyping and development of a system from existing blocks. Real-time systems are normally characterized by hard performance requirements, such as throughput, latency, etc. Therefore, at the early composition phases, reliable assessment methods are required to accurately evaluate and predict the performance of a designed system. Such analysis should consider the complete set of influencing factors, starting with intrinsic properties of hardware blocks (e.g. cache hierarchy) and ending with behavior of system tasks over the SW/HW topology and parameter-dependent workload. Another challenge comes from the limitations of analysis mechanisms, which are normally classified into two categories: analytical methods and simulation techniques. The former does not provide a detailed execution timeline, while the latter cannot guarantee a proper prediction of worst-case situations.

In the past decade, several methods addressing the problems of SW/HW component modeling, predictable assembly and evaluation of real-time systems have been proposed by the research community. Cortellessa *et al.* [2] have proposed a comprehensive approach for SW/HW component modeling, composition and consequent simulation of an assembly behavior. Klobedanz *et al.* [3] have discussed a performance analysis approach based on the AUTOSAR model. Both approaches do not provide platform-independent models with cycle-level accuracy. Bondarev *et al.* [4] have proposed a solution for design and performance analysis of conventional CBSE embedded real-time systems based on ROBOCOP components. This approach does not support detailed modeling and simulation of network-related primitives. Finally, Thiele *et al.* [6] presented an analytical method targeting worst-case latencies without predictions on detailed execution behavior.

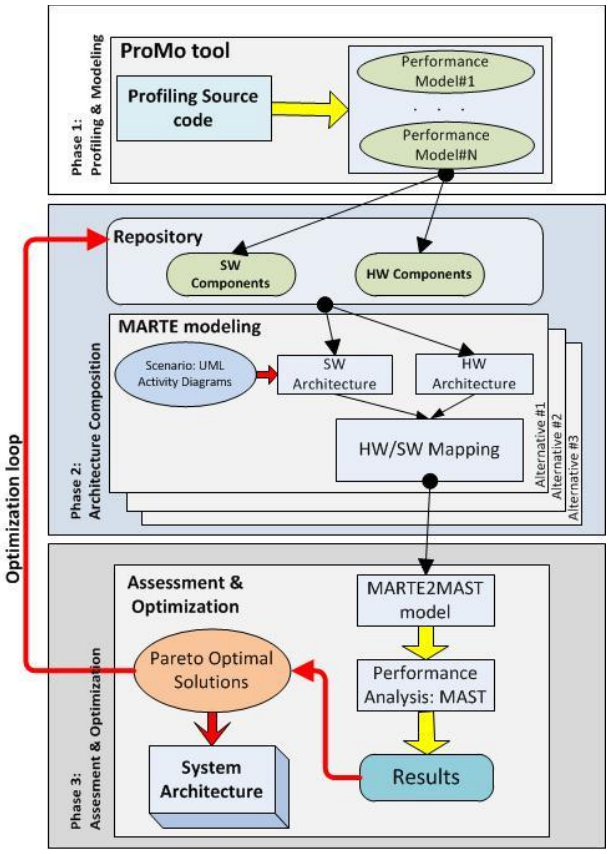


Fig.1: Analysis and design-space exploration method for RT systems

In this paper, we present our ongoing work on the ProMARTES method for analysis and design space exploration of real-time component assembly, see Fig. 1. The method consists of the following three phases. The *Profiling and Modeling* phase aims at profiling and automated generation of cycle-accurate performance models (MARTE compatible) for individual components at component development time. The *Architecture Composition* phase includes component selection, composition, SW/HW mapping and automated generation of a system model based

on defined workload scenarios. The composition can be performed for a number of architectural alternatives. The *Analysis and Optimization* phase enables prediction of system performance properties (latency, resource use, throughput, robustness, etc.) by schedulability analysis and simulation of the system model. The results are validated against the requirements, leading to follow-up design iterations. Each iteration searches for an optimal architecture by tuning the allowed *factors of freedom* (hardware topology, SW/HW mapping, scheduling policies, etc.).

The proposed ProMARTES method features a number of *benefits.* Firstly, the involved component profiling technique provides cycle-accurate performance metrics [1]. Our tooling chain offers automated generation of component performance models compliant with the UML-MARTE profile. Secondly, the established pipeline, generating models at different analysis phases, automates the analysis process and carries the profiled low-level metrics of the components through all phases, until the overall system performance is predicted. Thirdly, the method incorporates both the schedulability analysis and the simulation techniques. The schedulability analysis enables rapid identification of the best- and worst-case response latencies. However, it does not provide detailed behavior timeline data, average resource usage and latencies. In contrast, the simulation technique provides detailed behavior/execution timeline for all simulated system tasks, which enables identification of performance bottlenecks. Unfortunately, it requires a substantial time span to obtain converging prediction results. By combining these two analysis techniques, we complement the advantages and eliminate the limitations that each individual technique imposes. In conclusion, the worst-case predictions obtained at the early design phase by the schedulability analysis can be further used for a detailed simulation-based exploration of execution architecture problems (buffering, task interleaving, etc). Finally, the tool set for our method is encapsulated into the Eclipse Papyrus IDE environment, so that an architect can easily design the HW/SW architectures graphically and convert them into design models in an automated way.

Our method has a number of *limitations* which require further research. Firstly, the performance models can be obtained only for Linux-based operating systems and require the actual presence of the HW platforms. Secondly, the generation of the behavior models of the components is not yet automated and this task is supposed to be performed by the component developer. Thirdly, the method does not fully take into account the influence of the memory-, bus- and cache behavior on the performance of the system. For more accurate performance prediction, a cycle-accurate platform simulator needs to be integrated into the method. Moreover, due to the increasing popularity of applications that can be executed on a GPU, it would be valuable to support the modeling and the performance analysis of GPU-based systems. For analysis of network-related activities, ProMARTES does not incorporate the delays at the low OSI layers (transport, data link, physical), which reduces the accuracy of predictions on communication delays. We plan to integrate a more sophisticated network simulator for most of the OSI layers. Finally, manual composition of the architecture alternatives during the design space exploration is time-consuming and limits the space of possible alternatives. We are developing an engine for automated generation of architecture alternatives, which enables faster and broader exploration of possible design choices.

To *validate* our method, we have applied it to the real-world problem of an autonomous navigation robot system [5]. The system is composed of a robot and a remote processing node which communicate through a wireless network. The SW of the system is delivered by ROS, and it is based on four SW components. The navigation task is performed by 7 parallel tasks which characterize the behavior of the system. The most critical tasks of the navigation process are the `GM:Map` (composes the map of the environment) and the `MB:Nav` (transmits the control commands to the robot). Both tasks are periodic and characterized by hard real-time deadlines. We have composed the system and measured the actual latencies of the two critical tasks. Subsequently, we have compared these actual latencies to the predicted latencies, obtained by schedulability analysis and simulation techniques. The simulation predictions have shown a deviation of 1-2% compared to the actual response-time delay for the worst-case execution time (WCET) and 6-8% for the average-case execution time (ACET) of the two tasks. The predictions from schedulability analysis have shown that the predicted WCET is 8% higher than the actual WCET of the two tasks. The latter, increased, deviation can be explained by the fact that it cannot be ensured whether the system has reached the worst-case scenario during the actual execution. Moreover, we have applied a robustness test to check if the proposed architecture is still schedulable under overload conditions. To this end, we have increased the frequency of the robot's control loop by 10%. The system simulation has shown that the system still satisfies the hard real-time requirements with 3% increase of the WCET for the `MB:Nav` task. By examining the actual response-time delays, we have proven that also the actual system implementation satisfies the real-time requirements of the autonomous navigation robot.

The improved prediction accuracy of our framework is that our proposed method incorporates both the schedulability analysis and the simulation technique, which are complementary to each other in strength and eliminating the individual limitations.

## REFERENCES


[1] K. Triantafyllidis *et al.*, "Low-Level Profiling and MARTE-Compatible Modeling of Software Components for Real-Time Systems".

[2] V. Cortellessa, *et al.*, "Integrating Software Models and Platform Models for Performance Analysis".

[3] R K. Klobedanz *et al.*, "Timind Modeling and Analysis for AUTOSAR-Based Software Development - A Case Study" .

[4] Bondarev et al., "CARAT: a toolkit for design and performance analysis of component-based embedded systems".

[5] K. Triantafyllidis *et al.*, "Performance Analysis Method for RT Systems: ProMARTES for Autonomous Robot", submitted to FDL.

[6] L. Thiele, "Performance analysis of distributed embedded systems".