MODEL-DRIVEN DEVELOPMENT OF REAL-TIME EMBEDDED SYSTEMS

András BALOGH
Advisor: András PATARICZA

I. Introduction

Model-Driven Architecture (MDA) [1] has become a main trend in software development. It defines a development methodology, which highly relies on modeling and model-transformations. Using these techniques, the designer can concentrate on the functional aspects of the system, while all implementation-related information and the implementation of the system itself is automatically generated by software tools. The traditional application area of MDA is the domain of enterprise information systems.

However, there are several other domains that need support for handling the growing complexity of systems. The application of model-driven development methodology in embedded systems raises several challenges to tool developers. In contrast with the traditional EIS domain, where the main focus was on the functional requirements, embedded systems domain also requires the specification of non-functional (or Quality-of-Service) properties of systems. These properties have to be collected, and maintained during the development cycle, and must be enforced by the tools during code generation.

These additional requirements lead to a new, more complex development methodology and tool chain. We propose a model-centric approach for this problem in this paper that is based on the VIATRA2 [3] model transformation framework. We apply the basic approach to a safety-critical distributed domain, that is currently under development in project DECOS (Dependable Embedded Components and Systems – an EU Framework 6 Integrated Project) [4].

II. The Target Domain

Currently, in the automotive and avionics industry, the various functionalities of the on-board electronics are provided by separate, federated subsystems. This approach results in huge number of electrical control units (ECUs) that leads to increased costs and power consumption. These systems include both safety critical (SC) (e.g. x-by-wire) and non safety critical (NSC) (e.g. entertainment subsystem) applications.

To decrease the cost and complexity of on-board electronics, the most suitable solution is to integrate the subsystems into a cluster built from a relatively small set of computing nodes. This results in a mixed set of SC and NSC subsystems running in a common environment. To allow this mixed structure, the runtime platform has to ensure the proper temporal and spatial separation of different subsystems, while supporting the communication between the specified components.

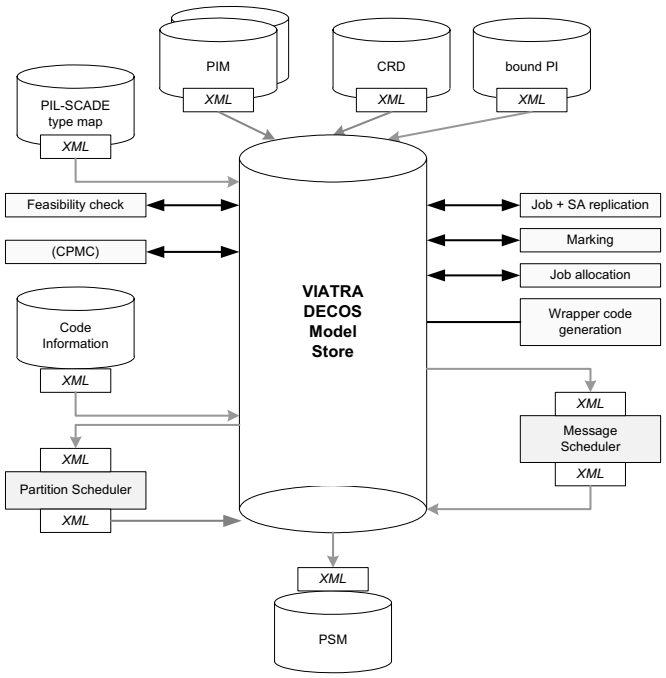
DECOS aims at integrating the various federated embedded computing subsystems into a single integrated system by a clear architecture that defines the required set of middleware services (e.g. messaging, group membership, global time) and the properties of platform operating systems (guaranteed separation of subsystems) [2].

III. Model-Driven Development in DECOS

The DECOS architecture supports complex applications; therefore we have to support the development in order to let the developers deal with this complexity. We have defined a domain-

specific platform-independent meta model (PIM) for distributed embedded systems that can be used to describe the functional, performance, and dependability aspects of the system, and we have also developed a platform-specific meta model (PSM) for the DECOS reference platform that also specifies all implementation-related properties of the systems.

The most important point of the development workflow is the mapping between the PIM and PSM, because we have to (semi-)automatically map the functional and also the non-functional requirements contained by the PIM to the actual hardware configuration (contained by the Cluster Resource Definition - CRD). We also have to take into account the actual Platform Interface parameters (timings, offered services, etc.).



The process requires a set of models be present simultaneously; therefore we have to use a generic model store. VIATRA2 [3] has been selected for implementation of the model store. The mapping itself cannot be full-automatic, because the developer may want to make some special restrictions (such as assigning specific jobs to specific HW nodes) called markings. These markings are integrated into the model and used by the subsequent steps.

The PSM generation starts with the job allocation, then the communication and task scheduling of nodes follows. The schedulers are third-party legacy tools. To allow the customization in each step, the user can review the results and make new markings, if needed. The consistency and feasibility of the PSM is continuously monitored by several special model transformations that search possible problems in the system model.

After the PSM generation is completed, the PSM can either be exported from the model store, or can be used directly to generate deployment configuration and wrapper code for the configuration of node operating system and middleware.

IV. Conclusions

MDA in Real-Time systems introduces new challenges for tool developers. Using the generic model store and model transformation infrastructure of VIATRA2 and some legacy tools we have succeeded to develop a PIM to PSM mapping that supports the specification of both functional and non-functional aspects of the systems. In contrast of the traditional full-automatic mapping, we also introduced a marking mechanism to support the manual, fine-grained customization of the system.

References

[1] The Object Management Group, *Model-Driven Architecture Information portal* http://www.omg.org/mda

[2] H. Kopetz, R. Obermaisser, P. Peti, N. Suri: *From a Federated to an Integrated Architecture fo Real-Time Embedded Systems*, - DECOS Technology paper

[3] *VIATRA2* – An Eclipse GMT subproject http://www.eclipse.org/gmt

[4] DECOS – An EU FW 6 IP http://www.decos.at/