Study on the Performance Modeling Approach for Automotive Embedded Control Software

Xiaofeng Yin, Jingxing Tan, Xiuting Wu and Qichang Yang

Abstract With the ever increasing complexity of automotive E/E (Electrical and Electronic) systems, model-based development techniques have been more and more widely used in the current development process of automotive embedded control software. Regarding the safety–critical automotive control systems with hard real-time characteristics, modeling timing and resource related performance and carrying out timing analysis for the control software at an early design stage play a crucial role to guarantee the quality of software as well as improve the cost-efficiency.

Keywords Meta-model · Performance · Modeling language · Embedded real-time system · Automotive control software

The motivation of this study is to investigate an appropriate performance modeling and timing analysis approach that can be integrated into the currently used model-based development tool chain. A performance modeling language for automotive embedded control systems (PMOLACS) at high level of abstraction was put forward using meta-modeling technique, which consists of three different meta-models corresponding to software structure (SWS), target platform system (TPS), and run time system (RTS), respectively. The SWS meta-model defines the modeling paradigm of the constituent, interactive behaviors, timing characteristics and resource requirements of software components, the TPS meta-model defines the modeling

F2012-D03-004

X. Yin (✉) · X. Wu · Q. Yang
Institute of Automotive Engineering, Xihua University, Chengdu, China
e-mail: xiaofengyin@vip.sina.com

J. Tan
Department of Science and Technology, Xihua University, Chengdu, China

paradigm of the constituent, timing characteristics and resource constraint of the hardware and real-time operating system (RTOS) of the target platform, and the RTS meta-model defines the modeling paradigm of executable software system. The algorithms regarding translating existing functional model with timing and resource requirements together into the SWS model, building TPS model, mapping the components of the SWS model to the processors of the TPS model to form the tasks of the RTS model, and timing analysis, will also be discussed.

1 Introduction

To meet the increasing demands on vehicle performance, such as drivability, safety, power, fuel economy, emission, as well as comfort, etc., a large number of embedded control units have been applied to the vehicle and the architecture of automotive E/E (Electrical and Electronic) systems has being become more and more complex. How to manage these complex E/E systems and ensure their performance requirements such as timing and resource constraints while their functional requirements are implemented and verified is a big issue in the current model-based development process, since the current automotive embedded software development process pays little attention to the non-functional requirements, especially the timing constraints, until the end of the development process—testing code on the target platform, although the algorithms related to the function of the system under development can be tested early through rapid control prototyping (RCP). If the system's ability to meet timing constraints could be analyzed formally in the early design process, it is possible to avoid the hidden errors to be left in the final product software due to lacking of direct timing analysis that may hurt the driver and passengers and/or destroy the vehicle, and to avoid costly late-stage redesign of the software that may postpone the delivery of the product software.

With respect to the complexity management of automotive E/E systems, AUTOSAR (AUTomotive Open System ARchitecture) provides a set of software infrastructure to enable the reuse and exchangeability of software modules between OEMs and suppliers through standardization of the software architecture of ECUs (electronic control unit) [1]. However, the main attention of AUTOSAR at present is focused on the implementation of software function.

Some other model-based design tools are also widely used in the development of automotive embedded control software. For instance, MATLAB Simulink/Stateflow [2] is used to design the control algorithms and then corresponding source codes are generated by a specific code generator such as Real-Time Workshop (RTW). As mentioned above, since the control algorithms can be optimized using RCP technique, the functionality of the controller can be tested at an early design stage. However, the direct formal verification of timing-related performance for embedded control software still can not be conducted in the current model-based development tool chain.

In this investigation, the meta-modeling technique was used to construct a performance modeling language for automotive embedded control systems (PMOLACS) that could model the timing constraints and resource requirements of software components, the resource constraints of the hardware that will be the target platform for the product software, as well as the virtual runtime system that could be used as a basis to analyze the schedulability of each task. In addition, a number of software modules (also called interpreters) implementing the algorithms of functional model reuse, component assignment, task forming, priority assignment and timing analysis were integrated into the modeling environment configured by the PMOLACS paradigm.

2 Modeling Requirements Related to Timing and Resource

In order to model the performance of automotive embedded control software, two main factors must be taken into consideration, i.e., timing and resource. Since most automotive embedded control systems (especially safety–critical system) are hard real-time systems, which means the completion of each task must meet its deadline, otherwise disastrous accident may occurs. On the other hand, the hardware resource of embedded controller is usually limited for the purpose of cost reduction. Therefore, the timing properties, resource requirements, and resource constraints must be described by the PMOLACS paradigm.

3 Performance Modeling for Automotive Embedded Control System

3.1 Modeling Approach

In this study, the automotive embedded control domain specific performance modeling language PMOLACS has been defined by a UML-based meta meta-model which defines a set of generic meta-modeling concepts including *Folders*, *Models*, *Atoms*, *References*, *Connections*, *Sets*, etc. [3]. These generic modeling concepts have been used to define the PMOLACS paradigm which is specified by a set of meta-models that can be further used to configure the modeling environment for automotive embedded control systems. In other words, we use the UML-based generic meta meta-model to define the meta-models of PMOLACS language, and the latter is then used to configure the generic modeling environment, and finally the configured modeling environment can be used to construct the models of automotive embedded control systems.

3.2 Performance Modeling for Automotive Embedded Control System

PMOLACS paradigm defines three meta-models for automotive embedded control system, i.e., (1) the software structure (SWS) meta-model that defines the modeling paradigm for software components, interaction between software components, timing properties and resource requirements of software components, (2) the target platform (TP) meta-model that defines the modeling paradigm for the constituent of hardware environment and real-time operating system (RTOS), timing features, and resource constraints of the target system, and (3) the runtime system (RTS) meta-model.

As specified in the SWS meta-model, the software system consists of a number of sub-systems that further consist of a number of software components. The execution time, priority, and required memory are captured by the attributes of the modeling element of software component. While the system deadline and execution period are captured by the attributes of the modeling element of sub-system. And the connection between sub-systems, between software components, or between subsystem and software component are described by association classes that can be divided into data connection and event connection, which have attributes describing the size of data passed and the size of data communicated, respectively.

As specified in the TP meta-model, the target platform system consists of a number of real-time operating systems (RTOS), central processing units (CPU) and networks which are further classified into CAN, LIN and FlexRay. Each CPU only has a unique RTOS associated with it. The hardware resource constraints such as the maximum memory, the minimum size of assignable memory, and the upper bound of utilization are captured by the attributes of the modeling element of CPU, which the speed (baud rate) and utilization bound of network are captured by the attributes of the modeling element of CAN, LIN and FlexRay. And the timing features such as the context switching overhead, scheduling overhead, timer overhead, timer resolution, etc., are described by the attributes of the modeling element of RTOS.

As specified in the RTS meta-model, the runtime system consists of a number of logical tasks that are used to group a number of tasks together, and the task further consists of a number of actions which are corresponding to the software components defined in the SWS meta-model. The execution time and required memory of software component are captured by the attributes of the modeling element of action. And the scheduling policy that may be preemptive, non-preemptive, or mix-preemptive (a policy defined by OSEK specification [4]), the response delay, the deadline, and priority of each task are described by the attributes of the modeling element of task. In addition, the logical task may have one or more triggers (corresponding to timer) which are used to periodically invocating tasks. The timing properties such as deadline and minimum period are captured by the attributes of the modeling element of timer.

4 Integration Algorithms with PMOLACS Towards Timing Analysis

4.1 Functional Model Importation

After the PMOLACS paradigm is defined, it is used to configure the generic modeling environment to build automotive domain specific performance modeling environment (PMOLACS modeling environment). The modeler can either build the SWS model for a specific automotive embedded control application and input performance parameters manually or reuse the existing functional model built by Simulink and add performance parameters automatically.

To reuse the existing Simulink model, a software module has been developed, which translates the atomic level functions of Simulink model into an equivalent model by replacing the *mutex*, *busses*, and *goto* blocks in Simulink with their equivalent connections in PMOLACS. The timing constraints and resource requirements such as required memory, execution time, and execution rate for each function is also input simultaneously. And the models with a number of hierarchical levels in Simulink are translated into a flatten model in PMOLACS.

4.2 Component Assignment

Once the SWS model is built either manually or automatically, the modeler could construct the TP model using TP meta-model to define the architecture of the target platform, such as how many processors will be used for the specific application, what kinds of networks will be used for each processor, and what kind of RTOS will be worked on each processor. During the process of target platform modeling, the modeler also needs to define the parameters of resource constraints and timing features manually in the PMOLACS modeling environment.

A software module has been developed to implement software component assignment based on the built the SWS model and the TP model, which maps each software component in the SWS model to one of the processors defined in the TP model on condition that the resource constraints can all be satisfied. Two different algorithms have been implemented in the component assignment module: one is load balancing that tries to balance the loads of different processors, the other is communication minimizing that tries to minimize the amount of communication across different processors.

4.3 Task Forming

Regarding grouping the software components together to form RTOS tasks, there exist conflicting strategies. If the task contains many components, the overhead of context switches will be reduced. If the task contains few components, the overall response time may be reduced. In the implementation of task forming in this investigation, the components having same execution rate, being assigned on a same processor, and not forming dependency loop are grouped together to form a task, for the purpose of reducing the overhead of context switches.

4.4 Priority Assignment

Three different algorithms have been implemented to assign the priorities for each task including: (1) the deadline-monotonic (DM) policy that assigns the task with the shortest deadline the highest priority [5], (2) the rate-monotonic (RM) policy that the task with the shortest cycle duration the highest priority, and (3) a combined policy that first assigns priorities according to the RM policy and then the DM policy is used to assign priorities to the tasks that have the same priority assigned by the RM policy. In addition, the modeler still can define the priority for each task in the PMOLACS modeling environment manually. Once all tasks have priorities assigned, the process of transforming the SWS model into the RTS model is completed and then the timing analysis can be performed based on the resulting RTS model to determine if each task meets its timing constraint.

4.5 Timing Analysis

The timing analysis algorithm consists of the following three main steps: (1) constructing task timing graph according to the priority, scheduling policy, and interactive relation of the task; (2) calculating the response time for each task, with regard to a specific task, this is done by summing up the response time of the direct predecessor of that task, the execution time of all tasks that preempt that task, the execution time of that task, the overhead used by task scheduling and context switching from the commencement to the completion of that task, via traversing all of the input concurrent links of that task on the task timing graph; (3) evaluating the schedulability for each task, if the response time of each task is not greater than the deadline of that task, the RTS is schedulable, otherwise, the design of the specific automotive control software needs to be refined such as re-assigning priority, modifying the architecture of target platform, or adjusting component mapping algorithms.

4.6 Algorithms Integration

The algorithms of functional model importation, component assignment, task forming, priority assignment, and timing analysis have been developed and implemented using Visual C++. The software modules corresponding to these algorithms have been built as dynamic link library and then registered as components in the PMOLACS modeling environment to work together with the PMOLACS paradigm.

5 Conclusions and Future Work

Aiming at performance modeling and timing analysis for automotive embedded control software at an early design stage, a performance modeling language PMOLACS is constructed using meta-modeling technique via the generic modeling environment, which is further used to configure an automotive embedded control domain specific modeling environment. With the implementation of a number of algorithms related to model reuse, model transformation, and timing analysis, the resulting PMOLACS modeling environment can be integrated into the current mainstream development tool chain for automotive embedded control system as an performance modeling, timing analysis as well as design automation or recommendation tool.

As part of our future work, integration of the output of RTS model with the source code generated from Simulink model in compliance with the state-of-the-art standards such as AUTOSAR still needs to be further investigated.

Acknowledgments The work reported in this paper was supported in part by the National Natural Science Foundation of China (Grant No. 60970072), the Scientific Research Fund of Sichuan Provincial Education Department (Grant No. 10ZA100, KYTD201003), and the Research Fund of Key Laboratory of Automotive Engineering of Sichuan Province (Grant No. SGXZD9902-10-1).

References

1. Bunzel S (2010) Overview on AUTOSAR cooperation. In: 2nd AUTOSAR open conference, Tokyo, Japan, May 13
2. Mathworks Website: http://www.mathworks.com
3. Ledeczi A, Maroti M, Bakay A, Karsai G, Garrett J, Thomason C, Nordstrom G, Sprinkle J, Volgyesi P (2001) The generic modeling environment. In: IEEE international workshop on intelligent signal processing (WISP'2001)
4. OSEK/VDX. OS 2.2.3. 2005. http://www.osek-vdx.org
5. Burns A, Wellings A (2001) Real-time systems and programming languages, 3rd edn. Addison Wesley, New Jersey