Integrating Model-based Formal Timing Analysis in the Industrial Development Process of Satellite On-Board Software

Rafik Henia[1], Laurent Rioux[1], Nicolas Sordon[1], Gérald-Emmanuel Garcia[2] and Marco Panunzio[2]
[1]*Thales Research & Technology, 1 Avenue Augustin Fresenl, 91767, Palaiseau Cedex, France*
[2] *Thales Alenia Space, 5 Allée des Gabians BP 99, 06156 Cannes la Bocca Cedex, France*

Keywords: Real-time Embedded Systems, Satellite On-Board Software Architecture, Model-based Formal Timing Analysis, Industrial Development Process, Component-based Design, Model-based Engineering.

Abstract: As a consequence of the increasing complexity of modern real-time applications, the need for an efficient, reliable and automated performance estimation method throughout the whole development cycle becomes essential. Model-based formal timing analysis appears at first sight to be the adequate candidate for this purpose. However, its use in the industry is conditioned by a smooth and seamless integration in the development process. This is not an easy task due to the semantic mismatches between the design and timing analysis models but also due to the missing links to the testing phase after code implementation. In this paper, we present a model-based timing analysis framework we developed in the industrial context of satellite on-board software. The framework enables overcoming the above mentioned problems, thus allowing a fully integration and automation of model-based timing verification activities in the development process, as a mean to shorten the design time and reduce risks of timing failures.

1 INTRODUCTION

The growing complexity of software applications combined with constant quality and shorter time-to-market constraints creates new challenges for the timing performance engineering practices in the development process of real-time embedded systems: it is expected that delivered products implement more and more complex features, while respecting stringent real-time requirements. When developing real-time systems according to a traditional application of the "V"-cycle, the timing verification activities start only when development and integration are completed. As a consequence, timing issues are not detected until the verification stage starts. At this time, they are more difficult and expensive to fix. Thus, a reliable timing performance prediction at early design stages is essential to guarantee that the designed system meets its timing requirements before time and resources are invested for the system implementation.

Formal timing analysis techniques are in theory well adapted for this purpose, since their applicability starts with the conceptual design phase and continues throughout all following development process phases. Furthermore, as recent development trends are based on a joint application of component-based software engineering (CBSE) and model-driven engineering (MDE), formal timing analysis can be applied directly on the design models: we would then refer to it as model-based timing analysis. Such analysis provides proofs for the timing behavior based on a mathematical model of the system timing behavior. These proofs allow calculating safe lower and upper bounds for performance values over a range of scenarios, thus guaranteeing corner-case coverage.

If model-based timing analysis seems to be so attractive in theory, what hinders its use in the industry?

A major reason is the lack of engineering methods allowing the integration of the model-based timing analysis in the different phases of the development process of real-time systems. Formal timing analysis is often not directly applicable to conceptual design models due to the semantic gap between the latter and the analysis timing model. In addition, after code implementation, the applicability of model-based timing analysis is hindered by the missing link to the testing phase of the implemented

Henia R., Rioux L., Sordon N., Garcia G. and Panunzio M..
Integrating Model-based Formal Timing Analysis in the Industrial Development Process of Satellite On-Board Software.
DOI: 10.5220/0004874306190625
In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development* (MBAT-2014), pages 619-625
ISBN: 978-989-758-007-9

software. Solving the above mentioned issues is therefore essential to break the walls separating the model-based timing analysis from the development process of real-time systems, in order to enable its use in the industry.

In this work, we present our model-based timing analysis framework allowing the application, the automation and the consolidation of formal timing analysis in the development process of real-time embedded systems starting from the early conceptual design phase until the integration phase, as a mean to reduce the design time and avoid costly timing failures detected after system implementation. The framework was developed in the context of the industrial component-based design and development of the Sentinel-3 satellite on-board software.

A description of the Sentinel-3 satellite on-board software use-case and the currently employed component-based design approach are given in the next section. Section 3 describes in detail the overall model-based timing analysis framework structure including the extension of the component-based design model with a timing performance model, the model transformation from design to timing analysis via a pivot analysis model, the worst-case timing analysis, the results translation process and the link to the system execution. In Section 4 we present the approach evaluation. Finally we draw the conclusions.

2 INDUSTRIAL CONTEXT

It has always been a challenge to introduce formal timing analysis into the industrial development process as the inputs required for the analysis, in particular the worst-case execution time (WCET) and the system behaviour description, are moving target all across the different development process phases. Starting from very high level system architecture and rough timing allocations, the formal timing analysis has to be refined at each step of the project (architectural design, detailed design, coding, unit test and software validation phases) down to concrete timing measurements on the final system.

Maintaining a representative timing analysis taking into account all the architectural, design (both static and dynamic) and timing changes across a complete development process is time consuming and error prone. The timing analysis shall be rather integrated into the development process. Its application shall be also automated in order to have the capability to continuously perform timing analysis during the process.

Thanks to the recent introduction of model based methods (in particular multi-view points) in the development process, this goal seems to be reachable.

The Sentinel-3 satellite on-board software use-case based on which we have developed our model-based timing analysis framework is an on-going project at Thales Alenia Space. The use-case as well as the employed component-based design approach are briefly described in the next two sections.

2.1 GMES Sentinel-3 Satellite On-Board Software

Sentinel-3 is an Earth Observation mission primarily devoted to support services related to the marine environment. It is one of the satellites of Copernicus (formerly known as GMES), an ambitious Earth Observation program to provide timely and accurate information for environment management, improve knowledge on climate change and help in civil security. The first Sentinel-3 satellite is expected to be launched between 2014 and 2015, followed by a second one so that they work together to provide maximum coverage. The mission's main objective is to determine parameters such as sea-surface topography, sea- and land-surface temperature as well as ocean- and land-surface color with high-end accuracy and reliability. Near-real time data processing and delivery will allow a broad range of Copernicus services for both the marine and land environment to continuously take advantage of the mission results. These services include, for example, maritime safety services that need ocean-surface wave information, surface temperature and data to improve ocean current forecasting systems; sea-water quality and pollution monitoring requiring advanced ocean color products in both open ocean and coastal oceanographic application areas; sea-ice charting services requiring sea-ice extent and iceberg detection; services to monitor land-use change, forest cover, photosynthetic activity, soil quality and fire detection.

Thales Alenia Space is the prime contractor of the Sentinel-3 mission and in particular it is the prime contractor of both the avionics and the platform on-board software of the satellite. The platform on-board software (OBSW) implements all major functions of the satellite: the Attitude and Orbit Control System (AOCS), the Thermal Control System (TCS), Mode Management, management of the ground/board interface, etc. A subset of those

OBSW functions is characterized by stringent real-time requirements (for example the commanding of the thrusters to perform an attitude modification or an orbital maneuver). The development process shall then be capable of statically ascertain that those real-time requirements are always fulfilled, by supporting a suitable and reliable form of timing estimation technique.

2.2 Component-based Design Approach

Thales created a software framework family, named MyCCM (Make your Component Container Model) (E. Borde, 2009), to support the implementation of real-time embedded software. MyCCM is a tailorable component-based design approach that takes inspiration from the Lightweight Component Container Model standard (LwCCM) defined by the OMG. MyCCM applications range from critical systems (e.g. on-board satellite software) to near real-time systems (e.g. image processing).

MyCCM implements the concept of functional components that encapsulate algorithms. MyCCM components correspond to passive code controlled by an underlying runtime, and are connected through communication ports to create a complete application. This allows the construction of applications by assembling independent functional components. It also enforces the separation of concerns between the functional aspects described by the component ports and the non-functional elements that stay outside the components (message chaining across the whole component architecture, FIFO sizes, task priorities, communications mechanisms, execution times etc.). MyCCM components can be seen as black boxes that contain the intelligence of the applications and are controlled by a runtime that is domain specific.

The MyCCM design process involves several aspects: data modeling, specification of the functional contracts of the components, specification of component implementations (i.e. the implementation of the algorithms), the connections between component ports, and possibly the allocation of execution resources (threads and mutexes) if the underlying runtime can be configured accordingly. MyCCM architectures can be described in Thales internal modeling tools or in UML modelers with plain UML. From such architecture models, it is possible to perform documentation and code generation.

3 MODEL-BASED TIMING ANALYSIS FRAMEWORK

The model-based timing analysis framework structure is illustrated in Figure 1. It is composed of five major steps. The first step consists in extending the design model with a performance model describing the timing characteristics, the behavior abstraction of the application and the execution properties of the platform. In the second step, the extended design model is transformed into a scheduling analysis model based on formal transformation rules preserving both the design model semantic and temporal behavior. The transformation is performed via an intermediate pivot analysis model. In the next step, formal timing verification is performed using a timing analysis tool. The fourth step consists in translating the timing analysis results to be compliant with the original design model and injecting them in the modeling tool. Finally, after code implementation, the formal timing analysis is refined based on concrete timing measurements extracted from test executions. All the mentioned steps in the model-based timing analysis framework are explained in detail in the following sections.

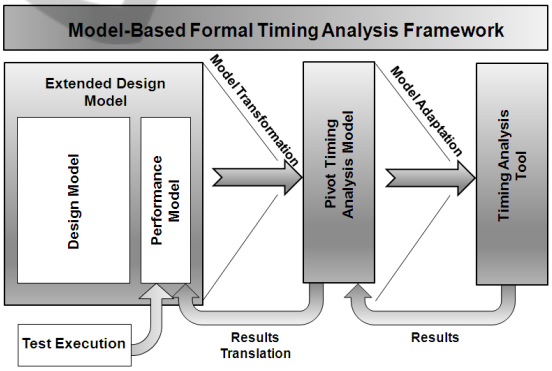

Figure 1: Model-based timing analysis framework structure.

3.1 Extending the Design Model with a Performance Model

MyCCM design models were not designed to take into account timing concerns. Since model-based timing analysis, calculates performance estimates based on the timing characteristics of the software application, the MyCCM design model has to be extended with a performance model. The performance model has to provide information describing the timing characteristics of the software application such as the activation frequency for the

tasks and the core execution times for the operations. Since execution times of operations are not yet available at early design stages, i.e. before code generation and implementation, the designer has to provide timing estimates instead (based on his experience of previous projects) and budgets (to be intended as an upper-bound requirement on the execution time).

As model-based analysis calculates performance estimates related to a specific execution platform, the performance model has also to provide information about scheduling and execution such as the processor speed, the scheduling policy (e.g. priority preemptive scheduling, EDF), the scheduling parameters (e.g. priorities), the tasks set and the mapping from the operations to task executors, etc.

And last but not least, MyCCM design models are static. Since model-based analysis calculates performance estimates related to the dynamic behavior of the application, the performance model has also to provide behavioral information such as the data dependencies between tasks and the communication protocols.

We selected specific concepts from the MARTE standard (MARTE, 2011) to build a performance model capable of expressing the needs described above. The MARTE standard is key technology for this purpose. It allows extending UML design models (UML, 2011) with concepts modeling the real-time constraints and the target platform, e.g. the SwSchedulableResource concept of the SRM (Software Resources Modeling) and the HwProcessor and HwBus concepts of the HRM (Hardware Resource Modeling).

In order to describe the dynamic behavior of the application and to annotate the MARTE-based performance model with the above mentioned timing and execution characteristics, we have used a so-called “Abstract Action Language” (AAL). AAL was defined at Thales and was originally intended to describe model-based functional test scenarios. We extended AAL with timing and execution elements to be able to annotate the performance model. Using AAL for both model-based timing analysis and functional testing allows increasing the design efficiency and robustness by avoiding the duplication of work and excluding the risks of consistency failures.

3.2 Filling the Semantic Gap between Design and Timing Analysis Models

Model-based timing analysis techniques are well adapted for the performance estimation at early design stages, since they rely on an abstraction of the timing relevant characteristics and behaviors such as execution scenarios and tasks activation and communication. From these characteristics and behaviors, the model-based timing analysis systematically derives worst-case scheduling scenarios and timing equations that provide safe bounds on the worst-case response times for each task.

However, we faced the problem that model-based timing analysis is not directly applicable to the extended MyCCM models due to the semantic mismatch between the latter and the variety of timing analysis models known from the classical real time systems research (MAST) (MPA) (CHEDDAR) and from the commercial timing analysis tools (SymTA/S).

For instance, in the common timing analysis models, a standard assumption is that a task writes its output data at the end of its execution. This is not the case in MyCCM. Operation calls in MyCCM are namely either synchronous (blocking) or asynchronous (non-blocking). As a consequence, the task, to which the caller operation is mapped, may write data into the input of a connected task, to which the called operation is mapped, at any instant during its execution and not necessarily at the end.

For the Sentinel-3 satellite on-board software use-case, the semantic mismatches between the design model and the different timing analysis models were basically related to the operation calls, accesses to semaphores and tasks activation behavior (data dependent activation).

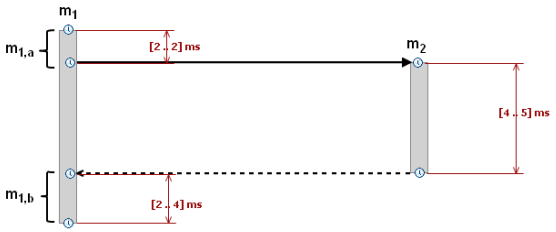

Figure 2: Synchronous call between operations in the MyCCM design model.

In order to overcome the semantic mismatch between design and timing analysis, we have defined a set of rules transforming extended MyCCM models into equivalent timing analysis models. This step is essential for the applicability of the model-based timing analysis to design models. In the following, we present an example of such transformation rules.

An example of a synchronous call between two operations in the extended MyCCM design model is illustrated in Figure 2. Let us assume that the operation m_1 is mapped to a task called T_1, while the operation m_2 is mapped to a task called T_2. Let us assume static priority preemptive scheduling for the tasks (Lehoczky, 1990). Regardless of the priority assignment for the tasks, the execution order of the operations will always be the following: after its activation, task T_1 will first execute the operation fragment $m_{1,a}$. Then, it calls task T_2. Since the call is blocking, task T_1 is suspended until task T_2 finishes executing the operation m_2 and sends data back. Then, task T_1 executes the operation fragment $m_{1,b}$.

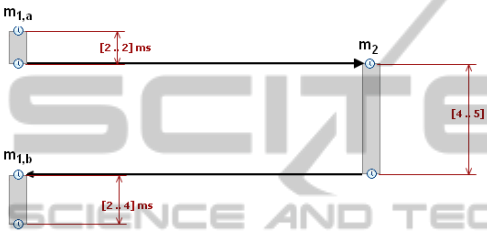

Figure 3: Transformed synchronous call between operations in the timing analysis model.

In order to keep the synchronous call behavior of the operations and tasks while respecting the timing analysis model semantic, we have to split the operation m_1 in two distinct operations corresponding to the operation fragments $m_{1,a}$ and $m_{1,b}$. This can be observed in Figure 3. We also have to split task T_1 in two tasks $T_{1,a}$ and $T_{1,b}$ that inherit its priority. Then, we have to map the operations $m_{1,a}$ and $m_{1,b}$ respectively to the tasks $T_{1,a}$ and $T_{1,b}$. Obviously, this transformation preserves the same execution order and thus, the synchronous call behavior of the original operations and tasks in the extended MyCCM model. On the other side, it is compliant with the above mentioned timing analysis standard assumption, since task $T_{1,a}$ calls task T_2 at the end of its execution and not before as task T_1 does.

3.3 Pivot Timing Analysis Model

Based on the set of transformation rules mentioned in the previous section, we can in theory directly translate an extended MyCCM design model into the selected timing analysis tool specific model. In practice, as illustrated in the framework structure in Figure 1, we have decided to introduce a pivot timing analysis model in-between, in order to ensure a minimum of independence from modeling and analysis tools.

The pivot timing analysis model is a sort of "standard" timing analysis model free from any specificity of the selected timing analysis tool. This makes the replacement of the analysis tool, if required, easier. In this case, the transformation from the extended MyCCM design model to the pivot timing analysis model remains unchanged, while only the interface between the pivot timing analysis model and the new selected timing analysis tool has to be implemented. The use of a pivot timing analysis model allows as well hiding the timing analysis tool complexity to the designer, since the latter does not need to pay attention to the tool specificities.

Generally speaking, the use of a pivot timing analysis model avoids the combinatorial explosion of transformations across every combination of modeling tool and timing analysis tool. Direct connections from N modeling tools to M timing analysis tools require N*M transformations. Connections from N modeling tools to M timing analysis tools via a pivot timing analysis model require only N+M transformations.

Note that in our developed model-based timing analysis framework, if required by the user, the pivot timing analysis model could be run in background thus being completely transparent to the designer.

3.4 Model-based Formal Timing Analysis

The next step in the model-based timing analysis framework is performing the timing analysis. The scheduling policy for tasks execution in the Sentinel-3 satellite on-board software use case is the fixed priority preemptive scheduling. The analysis was performed using the SymTA/S timing analysis tool (SymTA/S). The analysis algorithm is derived from the one presented by Tindell in (K.W. Tindell, 1994). Basically, based on mathematical proofs, the timing analysis algorithm calculates safe lower and upper bounds for the tasks response time values for each task, thus guaranteeing corner-case coverage. In addition to the response times, we use SymTA/S for the processor load, tasks output jitter and the buffer size calculation.

3.5 Timing Analysis Results Adaptation to Design Model Semantic

The worst-case responses times calculated by the

analysis tool SymTA/S are specific to the timing analysis model obtained after transformation of the MyCCM design model. Therefore, some calculated response times may be related to tasks and operations resulting from the splitting process explained in Section 3.2. Such tasks and operations are only notional and do not have a correspondence in the MyCCM design model. Thus, injecting the analysis results as calculated by SymTA/S in the MyCCM modeling tool may be confusing for the designer. A translation process for the calculated response times is therefore required.

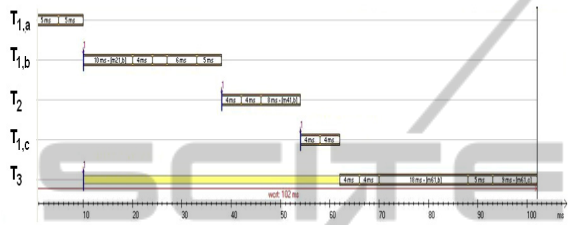

Figure 4: Gantt charts in the timing analysis tool SymTA/S.

Figure 4 shows a Gantt chart example produced by SymTA/S after worst-case timing analysis. It illustrates the worst-case execution scenario for a lower priority task T_3. As can be noticed, the worst-case scenario involves executions of the higher priority tasks $T_{1,a}$, $T_{1,b}$, $T_{1,c}$ and T_2. As their names suggest, the first three tasks were obtained after splitting a task called T_1 from the MyCCM design model. In order to provide the designer a comprehensive worst-case execution scenario, we have developed a translation process allowing merging executions of split tasks and operations and their illustration in dedicated Gantt charts. We implemented these dedicated Gantt charts using the TimingAnalyzer tool (TimingAnalyzer). The transformed Gantt charts are directly displayed in the modeling tool. Figure 5 illustrates the transformed Gantt chart obtained from the SymTA/S Gantt chart represented in Figure 4.

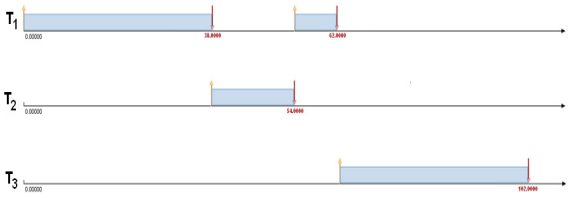

Figure 5: Translated Gantt charts injected in the design model.

3.6 Model-based Timing Analysis Consolidation through Functional Testing

The model-based timing analysis process presented above is based on timing budgets and estimates for the operations and tasks execution times. After code implementation, the defined timing budgets need to be consolidated. As illustrated in the model-based timing analysis framework in Figure 1, this is done with the help of model-based functional testing. Functional tests are namely generated from the design model and executed on the hardware platform. From the functional tests execution traces, lower and upper bounds for the core execution times of the operations and tasks are extracted. The obtained execution time intervals are then compared with the predefined timing budgets. If the execution time intervals fit within the timing budgets, there is no need to rerun the model-based timing analysis. Otherwise, the timing budget bounds are adjusted and the model-based timing analysis described in the previous sections is rerun using the new obtained core execution time values.

4 APPROACH EVALUATION

The model-based timing analysis framework was deployed on a demonstration platform where a representative part of an operational project (Sentinel 3) has been analyzed. To demonstrate the capability to use operational models, real MyCCM components of Sentinel 3 were used as starting point for the timing analysis framework. Then, a performance model was developed in order to describe the real-time architecture and the different temporal characteristics (WCET, deadlines, etc.). Then, model transformation is executed and the resulting pivot model is sent to the analysis tool without manual intervention from the user. This is a very important point for the acceptance of the solution as the pivot model does not need to be manipulated by the user, the user staying at the level of the MyCCM architectural model which is the right level of abstraction for him. After Analysis, the calculated results are translated then sent back to the design tool to offer a view at the right level.

The integration of the timing analysis results into the MyCCM design tool is a key enabler for a seamless integration of the model-based timing analysis into the development process. This integration moves the timing analysis from a parallel

loosely correlated to the main stream of the development process to a completely integrated process.

The main advantage of the approach is the elimination of the redundancies in the development process (avoiding describing the design for the timing analysis tool and thus avoiding eventual errors and non-synchronization). As the dynamic aspects are modeled in the same design tool than the architecture, they are always synchronized enabling an execution of the analysis very often during the development process (even eventually thought continuous integration process). Finally as the dynamic architecture is fully described in the design tool, generative techniques ensure that the model will always represent the final system (indeed all the code related to real-time entities declarations can be fully generated).

5 CONCLUSIONS

In this work, we present a model-based timing analysis framework developed in the industrial context of satellite on-board software, allowing the full automation of timing verification activities, their application at early design stages and their transparent integration in the development process of real-time embedded systems.

The model-based timing analysis framework allows bridging the existing semantic gap between the design models and the common timing analysis models through the application of dedicated model transformation rules, thus permitting the integration of the variety of existing timing analysis tools in the design process.

Through the use of a pivot timing analysis model, the developed framework ensures an independence from the selected timing analysis tool specificities and facilitates its replacement.

Other benefits of the model-based timing analysis framework are related to the increase of the design efficiency and robustness by respectively avoiding the duplication of work and excluding the risks of consistency failures through its applicability to the functional design model.

We believe that the model-based timing analysis framework we have developed represents an important step toward the full acceptance of the model-based formal timing analysis techniques in the industry.

ACKNOWLEDGEMENTS

The work described in this paper is partially funded by the ARTEMIS project MBAT (https://www.mbat-artemis.eu/).

REFERENCES

E. Borde, F.Gilliers, G. Haik, T. Vergnaud, J. Hugues, L. Pautet, 2009. *MyCCM-HI, a component-based Framework Implementing a Model Driven Engineering Approach*. Neptune 2009.

MARTE 1.1, June 2011. OMG publication, 2011-06-02.

UML superstructure 2.4.1, August 2011, OMG publication, formal/2011-08-06.

Lehoczky, 1990. *Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadline*. IEEE Real-Time System Symposium (RTSS 1990).

SymTA/S https://www.symtavision.com/symtas.html

K.W. Tindell, 1994. *An extendible approach for analysing fixed priority hard real-time systems*. Journal of Real-Time Systems, 6(2):133–152, Mar 1994.

TimingAnalyzer http://www.timing-diagrams.com/

MAST http://mast.unican.es/

MPA http://www.mpa.ethz.ch/Rtctoolbox/Overview

CHEDDAR http://beru.univ-brest.fr/~singhoff/cheddar/