A Modeling Approach for the Timing Verification of COTS Components-based Distributed Hard Real-Time Systems

Julio L. Medina[1,2], Patricia López[1], José M. Drake[1], François Terrier[2], and Sèbastien Gerard[2]

[1] *Universidad de Cantabria, Computers and Real-Time Group, Av Los Castros s/n, 39005-Santander, SPAIN*

[2] *Commissariat al'Energie Atomique - Saclay, DRT/DTSI/SOL/LLSP, F-91191, Gif-sur-Yvette Cedex, FRANCE*
{medinajl, lopezpa, drakej}@unican.es, {julio.medina, sebastien.gerard, francois.terrier}@cea.fr

Abstract

The specification data sheet given by the suppliers of Commercial Off-the-Shelf (COTS) components that are planned to be used in distributed real-time systems, like those intended by the AUTOSAR initiative, must include information about their timing behaviour. A first version of this information must be initially stated by the car manufacturers at the specification of future components, but its final accurate model is indispensable at integration time in order to evaluate the schedulability of the complete applications that may use them. This position paper suggests a prismatic modeling approach that allows the timing behavior specification of the services and resources offered by the real-time components to be expressed independently of the models of the other components and the platform they rely upon. This opacity is strictly required by the automotive industry to preserve the intellectual properties on the implementation of the delivered components. The analysis model used follows the transactional approach implemented in the MAST component-based real-time modeling methodology. It satisfies the opacity and composability properties required for the automatic assembly of models that hold the temporal behavior of component based applications. Some configuration-time services are required to suppliers to tune the analysis models at integration and deployment time.

1. Introduction

Along the last decades, a significant part of the evolution in functionality and novelty in the automotive domain has relied on the advances in microelectronics, networks and software that have been introduced on it. The number of microprocessor based controllers, known in this domain as electronic control units or ECUs, has raised in at least an order of magnitude, and most importantly, the structure of their relationships has changed from stand-alone to largely distributed systems.

This technological trend represents a big change in the automotive industry, not only for the development costs structure, whose bigger part is shortly about to shift to the electronics and embedded software side, but mostly because of the new role of the car manufacturers in the development process.

Traditionally the car builders, known as the Original Equipment Manufacturers or OEMs, outsource the development of parts to external suppliers, who specialize and serve parts as black boxes to different OEMs. This policy extends from mechanical parts to complete ECU-based subsystems. OEMs give to suppliers precise specifications, and take the responsibility of verifying and integrating them in their final products. The verification of each ECU and its corresponding wiring subsystem can be made in isolation since the interactions between subsystems are limited and known by the OEM, and furthermore they are able to be stated in the part specifications as maximum ratings, typically temperature, vibration or electromagnetic interferences. The temporal properties can also be tested in isolation by the supplier prior to delivery since there is not timing expected interference from the rest of the system. This model works fine with a reduced number of isolated components, but quickly collapse when the number of them rises at the current trend, particularly because for cost-efficiency reasons they are forced to share the processing and communication platforms. In a component based approach for development, like the one proposed by the AUTOSAR initiative [1], the OEM must not integrate and verify complete loosely coupled subsystems, but fine grain third party black-box software components delivered by the suppliers, whose operation may be easily affected by other components in the system. This becomes even more challenging when the timing properties are to be evaluated, specially when the components are part of larger control loops, for which the timing requirements can only be expressed as maximum end-to-end deadlines and jitters. These challenges require from the OEMs more than regular software engineering capabilities, they are now being invited to be full members of the selective club of companies developing hard real-time distributed systems.

2. A new OEM-Supplier relationship

Besides the real-time constraints, other aspects like the important influence of the final user in the overall design, the high dependability expected, the heterogeneity of the models of computation involved in the distributed control algorithms, or the constraint resources in the embedded platforms used, post to the OEM a significant effort to define an architectural description of the prospective software components involved. This implies a richer definition of components, which is made not only at the functional assembly level, it requires also a description of the minimum internal aspects of the components that will serve to infere the expected Non Functional Properties (NFPs) of the entire system. In particular the models to evaluate its timing behaviour. The timing budgets and resource allocation validated in this primary stage must be able to be verified after. The models used to describe NFPs at this stage may be used as part of the specifications in the contracts with suppliers.

This intra components models will be of course much complete and close to the final ones as experience is gained by the OEM, but at least for the complex software components a certain degree of interaction supplier-OEM is envisioned along the development process. This interaction claims at the minimum for the confidence that the final verifiable timing models of a component returned by the supplier match the binaries delivered. Nevertheless, for new and/or risky projects, it is advisable to consider a slightly closer collaboration with suppliers, in such a way that any unveiled architectural interaction that may be discovered by them shall be reported to the OEM soon in the development process. These refined views may be formalized by concrete deliverables that report about the feasibility of the specification at concrete stages of the development. In particular, the models used to describe the timing, reliability/safety, dynamic memory consumption, or any other relevant NFP must be kept up to date as soon as possible.

Considering the complexity and variability of execution platforms, the final delivered components must include deployment oriented specific services. These services will be used to measure the execution times over the concrete platform on which it is deployed for all (or the critical set of) the sections of code used in the characterization of the timing behaviour of the functional component services. These validation specific services are to be programmed by suppliers and included in the binary code, in such a way that they return the worst, the best, and the average values for the execution times to be used in the analysis models.

3. A breif overview of real-time CBSE

Component-based software engineering (CBSE) is one of the most promising technologies to be applied in the real-time domain, in order to rise software quality, reduce time to market and manage its ever growing complexity. Even though an increasing demand of real-time CBSE technologies is pointed out by the automation, power systems, avionics, or automotive industries [2][3], its actual usage is limited due to the lack of experiences in the effective implementation, analysis, and verification of predictable components.

In this work we understand a component as “A non trivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture, and conforms and provides the physical realization of a set of interfaces” [4]. This concept stresses components as coarse grain reusable modules, developed independently of its usage in a particular application and replaceable by others functionally equivalent. The components modeling methodology used here not only considers the modeling of real-time software applications, but also the middleware modules and software/hardware platforms, are conceived as components, since the main abstraction behind their models is the predictability of the temporal response of their offered software services. Some of the key issues to be addressed in the real-time modeling of services offered by components in a CBSE environment are:

- The real-time behavior of a service depends not only on the code that actually implements it, but also on the behavior of the other components’ services used for its instantiation. This forces to describe the collaboration and synchronization mechanisms at a level quite lower than the usual functional specification of the interfaces.
- Real-time applications are often embedded and limited in resources, hence general purpose technologies like EJB/J2EE, CCM or COM+ are not directly suitable, particularly without disks or secondary memory. Others like CIAO, Lightweight CCM or TimeWeaver are useful, and despite the huge complexity of the resulting hard real-time models, they are able to be analyzed.
- The Timing response of a component depends also on the platform in which it is deployed (hardware, operating system, communication resources, etc.). Real-time components require real-time platforms, whose services have predictable timing responses and offer the proper timing management primitives and scheduling services.
- Predictability of a real-time application is affected by the others running on the same platform. For embedded systems it is usual to apply offline schedulability and performance analysis techniques since the whole system load is known. However for complex component based distributed system, it is rather difficult to known in advance the actual total workload, so platforms must support run time services for the reservation of processing capacity at components instantiation time [5].

4. The transactional analysis approach

One of the scientific approaches that promise to be effective in the validation of component based models for timing analysis is the one based in transactional models [8]. A number of tools and techniques for schedulability analysis and response time calculation have been proposed for it [9] [10], and its fundaments are referenced by the synchronization protocols and scheduling policies used in the vast majority of real-time as well as general purposes operating systems [11]. The transactional model is used as a reference for analysis by the "UML profile for Schedulability, Performance and Time" (SPT), current OMG standard for modelling and analysis of real-time systems [12].

The transactional methodology models a real-time application by two complementary descriptions:

a) Control flow (transactional) model: It is a reactive model, which describes the application as a set of concurrent real-time transactions, which are sequences of activities that are triggered in response to external or timed events. A transaction is described by its causal flow of activities, the generation pattern of the triggering events, and the timing requirements that must be met. An activity describes the amount of processing capacity that is required to perform the duty that it has associated. There is no direct activation or execution flow dependency between activities in different transactions; they only interact by sharing the processing-resources and the mutually exclusive passive resources.

b) Resources contention model: It describes the active and passive resources that are used by the activities in a mutually exclusive way, showing characteristics like their capacity, overheads, access protocols or scheduling policies. It is used to evaluate the blocking time in the access to passive synchronization resources or while contending for active resources like processors or networks.

One of the major assets of this modelling strategy is that partial transactional models can be generated, stored, and characterized in a parametric way. These models represent the ways in which the relevant services of the components are used in the critical situations to be analyzed, and may be associated with the components at different stages of their development. This permits to formulate the real-time model of a system wide application with end-to-end constraints by composition of the individual real-time models of the software and hardware components that forms it. The modeling approach is said to be prismatic in the sense that components are not grey-boxes, they have a number of coloured reflective models, each for a particular purpose.

This approach allows for the schedulability analysis of both, hard real-time configurations, and soft real-time situations. The first are made using the worst and best case timing values, while the latter uses average and probability distribution values. These timing values are estimations or measurements of both, the CPU execution time of code segments, and the time used to transmit messages through the networks. The techniques available include holistic analysis, which take advantage of priority based networks (like CAN Bus) and make the calculus less pessimistic. Design tools are available to obtain feasible sets of locally optimized priorities for messages and threads in distributed systems, though unfortunately currently no absolute optimal solutions exist in this general case.

Among the limitations of this approach should be considered the fact that complete models are necessary to make any sort of prediction of the system behaviour. No independent or partial evaluation of components is possible if they require others to operate. Of course estimated models may be used, but there is not guarantee of a linear or "smooth" adjustment along the development process, in particular in presence of strong architectural changes like the introduction of mutually exclusive shared resources, or significant new functionality. For small adjustments in timing values instead, response time analysis techniques do allow to get some feed back, like the proportion in which timing values of a transaction may be enlarged or should be reduced to get schedulability.

But maybe the characteristic of this approach that requires more effort to apply it in a component technology framework like AUTOSAR, is the fact that one, both, or a combination of these two solutions are necessary: Whether the OEMs build very precise architectural and behavioural specifications of the components they need, and/or the suppliers are obliged to bring the precise timing behavioural models of the code corresponding to the binaries delivered. This may seem to be a partial infringement of the IPs, but there is still a significant margin for creativity between the timing abstraction model and the binary code. This is more a management than a technical challenge that it is worth to face, particularly considering the predictability gained.

The information in the transactional model of a real-time application tend to be complex, therefore, specific and commercial tool support shall be required for its processing and management. Standards for interoperability among tools are also useful to exploit it effectively for analysis and design. This standardization effort is currently under development [15], and a response is expected to the OMG request for proposals for the UML Profile for Modeling and Analysis of Real-Time and Embedded systems [13].

5. Relevant features of MAST

The aim of the MAST methodology [6][7][8] is to enable the modeling of the temporal behavior of a real-time situation, in order to evaluate its schedulability and predict its timing responses by means of automatic tools. A real-time situation represents a specific mode of operation of the system, characterized in the transactional approach by the workload that it is demanded, the set of transactions involved, and the platform resources used. Complete information, and download is at http://mast.unican.es

With the analysis and design tools that are available in the MAST environment, the model can be analyzed in order to generate results such as:

- Whether the set of transactions is schedulable or not.
- The worst-case response times, which can be compared to the respective deadlines.
- The system and per-transaction slacks. They are the percentage by which the execution time of all operations (of the system or of the transactions) in the rt-situation may be increased while still keeping the system schedulable.
- The processing resource slack, evaluated as the percentage by which the speed factor of each processing resource may be decreased while still keeping the real time situation schedulable.
- The utilization of the resources, evaluated as the relation, in percentage, between the time that the resource is being used, and the total elapsed time.
- The local optimal sets of priorities to assign to threads, and messages in order to have the system schedulable.

6. Conclusions and future work

This paper suggest the usage of the transactional approach for the analysis of temporal properties of component based models like the one in the AUTOSAR initiative. To accomplish this goal the expected real-time behaviour of each component must be stated by the OEM as part of its initial specification, but must be also re-stated posibly refined, by the component supplier at the delivery of the binaries. This implies the statement from the supplier that the given real-time models faithfully reflect the internal structure of the binary code. The supplier must also provide run-time services useful to measure the worst, best, and average execution times for the running platform, so that accurate analysis models could be created. This data will give the time they take all the operations for the real-time situations of interest.

The MAST modeling methodology and the set of tools that it brings are useful for the modeling, composition, and evaluation of component base distributed real-time systems, provided they are annotated with the necessary information for the analysis. The transactional models are a good modeling abstraction, that may preserve the IPs of the suppliers while giving to the OEMs the necessary information to evaluate the system performance at integration time.

Further work must be made to explore the feasibility of coarse granularity abstractions, like those proposed by the Accord Modeling Suite [14], so that a high level design methodological approach may drive the automatic obtention of the models.

References

[1] http://www.autosar.org/index.php

[2] Crnkovic I. and Larsson M.: "Building Reliable Component-Based Software Systems" Artech House Publishers, 2002.

[3] Norström C. et al.: "Experiences from Introducing State-of-the-art real-time Techniques in the Automotive Industry". Proc. Of 8th IEEE Int. Conf. On Engineering of Computer Based Systems (ECBS01) Washington, April, 2001.

[4] A. Brown and K. C. Wallnau: “The current state of CBSE”, IEEE Software, pp.37-46, September-October 1998.

[5] M. Aldea, G. Bernat, I. Broster, A. Burns, R. Dobrin, J. M. Drake, G. Fohler, P. Gai, M. González Harbour, G. Guidi, J.J. Gutiérrez, T. Lennvall, G. Lipari, J.M. Martínez, J.L. Medina, J.C. Palencia, M. Trimarchi. “FSF: A Real-Time Scheduling Architecture Framework”. RTAS'06 pp. 113-124, april 2006

[6] M. González Harbour, J.J. Gutiérrez, J.C. Palencia and J.M. Drake: “MAST: Modeling and Analysis Suite for Real-Time Applications”. Proceedings of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, June 2001

[7] J.L. Medina, M.González Harbour, J.M. Drake:" "MAST Real-time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems", RTSS, December, 2001.

[8] P. López, J.L. Medina y J.M. Drake. Real-Time Modelling of Distributed Component-based Applications. Procc of the EUROMICRO-SEAA'06. IEEE Press, august 2006.

[9] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, A Practitioner's Handbook for Real-Time Systems Analysis, Kluwer Academic Pub., 1993.

[10]A. Cheng, "Real-time Systems Scheduling, Analysis and Verification". ISBN 0-471-18406-3. J. Wiley & Sons, 2002

[11]IEEE Std 1003.13TM-2003: "IEEE Standard for Information Technology-Standardized Application Environment Profile (AEP)-POSIXR Realtime and Embedded Application Support", 2003.

[12]Object Management Group: "UML Profile for Schedulability, Performance and Time Specification", Version 1.1. OMG document formal/05-01-02, January, 2005.

[13]Object Management Group: UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), RFP. 2005. OMG document: realtime/05-02-06.

[14]Gerard S. and Terrier F. “Intensive use of UML model transformations: the ACCORD environment”. WTUML: Workshop on Transformations in UML, (ETAPS 2001)

[15]Medina J., Lopez P. and Drake J.M. “Towards a UML Profile for Real-Time Modelling of Component-Based Distributed Embedded Systems”. FDL'06, 2006, ISSN: 1636-9874