Enabling Model-Driven Schedulability Analysis in the Development of Distributed Component-Based Real-Time Applications

Patricia López Martínez, José M. Drake and Julio L. Medina

Departamento de Electrónica y Computadores, Universidad de Cantabria, 39005-Santander, SPAIN
{lopezpa, drakej, medinajl}@unican.es

Abstract

This paper presents a strategy to include temporal behaviour metadata in the descriptors of software components in order to develop hard real-time component-based distributed applications, keeping the opacity and composability features that are inherent to the components paradigm. The Deployment and Configuration of Component-based Distributed Applications Specification of the OMG has been extended to include and manage the information that is required to design, analyse, and configure component-based applications with hard real-time requirements. The real-time data added to a component interface enable the application designers to validate scheduling and design decisions without any knowledge of the component's code. Besides, real-time reusable and composable analysis models, developed according to a concrete modelling methodology, are added to each implementation of the component interface. In the context of a concrete application, they are processed by tools to generate the complete real-time analysis model of the application, which is used to evaluate the configuration parameters that guarantee its schedulability.

1. Component-based real-time applications[1]

Component-based software development reveals as one of the most promising software engineering approaches to be applied in the industry nowadays [1]. In the case of applications with real-time requirements, a temporal behaviour model of the application must be built to evaluate the scheduling parameters configuration. With that purpose, the designer of a component must generate, together with the component's code, a parameterized model, which abstracts the timing and sequence of all the actions performed by the component, and includes all the scheduling, synchronization and resources information that is necessary to predict the real-time qualities of the applications in which the component is integrated. This model must be included as non-functional metadata in the package with which the component is delivered. Later, when an application is built as an assembly of components, the application designer, in analogy to the generation of the application's code as a composition of the code of its constituent components, can also compose the set of real-time models and build the real-time analysis model of the application.

Nowadays, there are several methodologies for the analysis and design of component-based real-time systems. Some of them [2,3] elaborates a model specific for the complete assembled application. Similarly to our approach, [4] addresses model composability, but the formulation of the real-time models that is proposed in this paper, improves the reusability and opacity inherent to components.

The models are handled by means of an extension of the Deployment and Configuration of Component-based Distributed Applications Specification of the OMG [5] (D&C), called RT-D&C [6]. The process followed for the design of real-time characteristics must be consistent with the general purpose development process of component-based applications, so this approach considers the process defined in D&C, which is shown in Figure 1 with some of the real-time extensions proposed. In order to design applications with real-time requirements, the components used to build them must offer predictable temporal behaviour, and the resources and services provided by the execution platform must also exhibit bounded timing responses. As shown in Figure 1, the actors involved in the development of real-time applications have access to the models that describe the temporal behaviour of the components, which are provided as part of the packages in which the components are delivered. They also can access the real-time models of the platform resources. The strategy used to formulate these models is introduced in Section 2.

The assembler builds the application as an assembly of component instances that implements the functional and non-functional requirements of the application specification. According to D&C, he takes his decisions based only on the metadata available in the component interfaces descriptors, which are independent of the component

1. This work has been funded by the EU under contracts FP6/2005/IST/5-034026 (FRESCOR), FP7/NoE/214373 (ArtistDesign), FP7/CSA/224330 (ADAMS); and by the Spanish Government under grants TIN2008-06766-C03-03 (RT-MODEL), and TFI-020400-2008-99 (EVOLVE). This work reflects only the author's views; the EU is not liable for any use that may be made of the information contained herein.

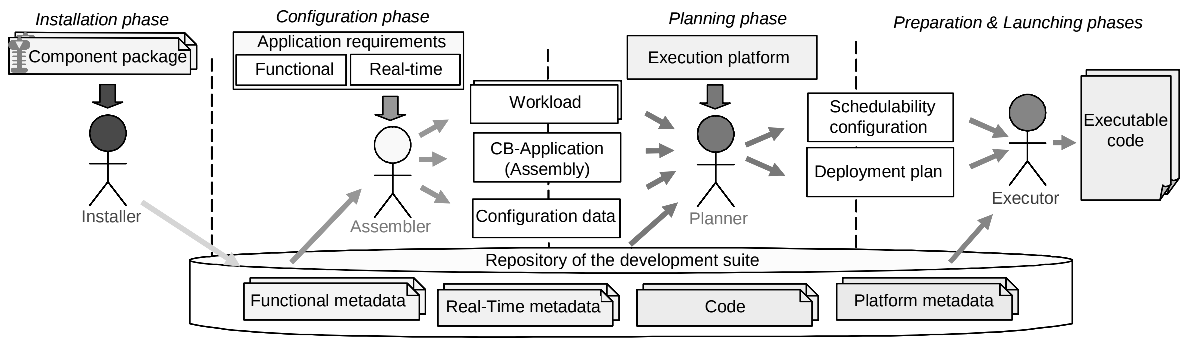

Fig. 1. Development process of component-based real-time applications

implementations that might be chosen later. If the application has real-time requirements, the assembler has two new responsibilities:

- He must select the components so that the application can fulfil its real-time requirements. A reactive model based on transactions [7][8] is considered for the real-time requirements specification. The activities executed in the application are formulated as a set of end-to-end flow transactions, each of them representing the sequence of activities executed in the application in response to an external or timed event. The timing requirements are formulated on them, as temporal constraints between their internal execution states. The RT-D&C descriptor of a component interface includes the declaration of the transactions that can have origin in the component. So, in a real-time application, the assembler chooses the components not only based on their provided functionality, but also because they implement the appropriate transactions. Together with the description of the application, the assembler must formulate, by means of a new descriptor defined in RT-D&C, its workload, i.e., the set of concurrent transactions executed on the application, with their corresponding timing requirements.
- He must select only components that lead to an application with predictable behaviour, i.e. for which an analyzable behaviour model can be obtained. With that aim, in RT-D&C the metadata related to the components ports include the enumeration of the operations offered by the component that has a real-time model defined, and the enumeration of the operations whose real-time models are needed by the component. Two components are composable from the real-time predictability point of view, i.e. its connection has predictable behaviour, when the server component provides real-time models of the services required by the client component.

The responsibility of the assembler is different regarding the functional and the real-time requirements. The assembler buils an assembly that guarantees the fulfilment of the functional requirements of the application. However, regarding real-time requirements, he can only guarantee that it is possible to generate an analyzable temporal behaviour model of the application, which can be used to verify if the timing requirements are met. This is because the information available in the component interfaces is not enough to evaluate their temporal behaviour. The assembler does not know the concrete implementation of each component, neither the processing capacity of the platform, both of which are required with that purpose.

The planner defines the concrete implementations used for each component instance, and the processing nodes in which they are installed. He does have all the information required to evaluate the application's temporal behaviour, so, he is responsible of designing the application to make it schedulable, i.e, to make it meet all its timing requirements. This real-time design process includes:

- Obtaining the values to assign to the schedulability configuration parameters of the components that make the application schedulable. These parameters can be priorities of threads, priority ceilings, EDF deadlines, etc.
- Obtaining the values to assign to the configuration parameters of the execution platform that guarantee the schedulability of the application. Example of this kind of parameters are the number of available threads or communication channels, the priority of transmission of messages through the network, etc.

Designing the schedulability of a real-time application is a very complex task, which can not be based on the designer intuition or expertise. To deal with it, the planner must be able to build the reactive model that describes the application temporal behaviour, and apply real-time analysis tools to it, in order to calculate the set of schedulability configuration values that leads to an schedulable application. This process is explained in Section 3.

2. Reactive model of an application

The description of the temporal behaviour of an application constitutes the basis for determining the configuration of the components and the platform that guarantees the application schedulability. It is described by means of a reactive model, which conceptually corresponds to the one introduced in the "UML Profile for Modelling and Analysis of Real-Time and Embedded systems" (MARTE) [7] of the OMG. The model is organized around the concept of *AnalysisContext*, which represents a specific operational mode of the system taken as the basis for the analysis. It is described by means of two complementary aspects:

- The *ResourcePlatform* comprises the resources available in the execution platform, like *ProcessingResources* (processors or networks), *SchedulableResources* (threads, processes, etc.), *SharedResources* (synchronization mechanisms, critical sections, etc.). The execution times of the components depend on their speed, and, being mutually exclusive, they cause interdependencies in the execution of the activities deployed on them.
- The *WorkloadBehavior* declares the load imposed over the activities performed by the system. This is organized according to a reactive model that describes the set of independent *EndToEndFlow* transactions that execute concurrently in response to external (coming from the environment) or timed (sent by the system timer) *WorkloadEvents*. The time instants in which the events are generated are defined by means of a deterministic or probabilistic pattern. The end-to-end flow transactions constitute also the scenarios in which the timing requirements (deadlines, jitters, miss-ratios, etc.) are expressed by means of *TimingObservers*. Each *EndToEndFlow* is described as an ordered set of *Steps* that take place according to the expected causal control flow.

In classic hard real-time systems design strategies, this reactive model is used to conceive the system timing behaviour and to devise its implementation. However, when the application is built by assembling reusable, opaque code components, the reactive model has to be

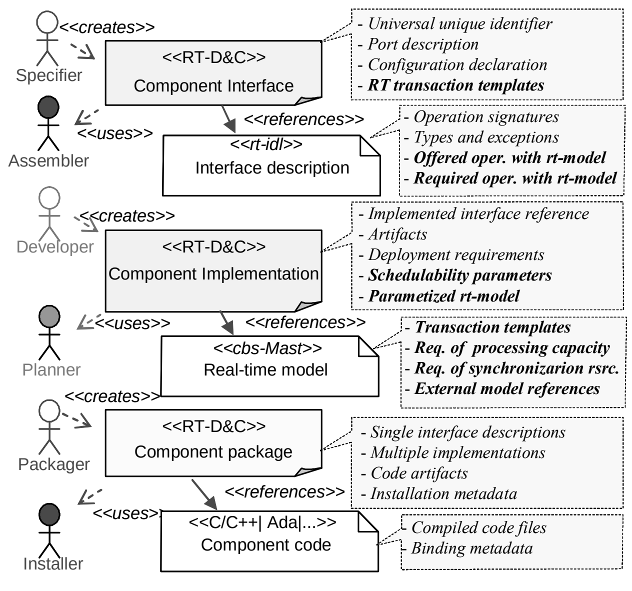

Fig. 2. Description of a real-time component.

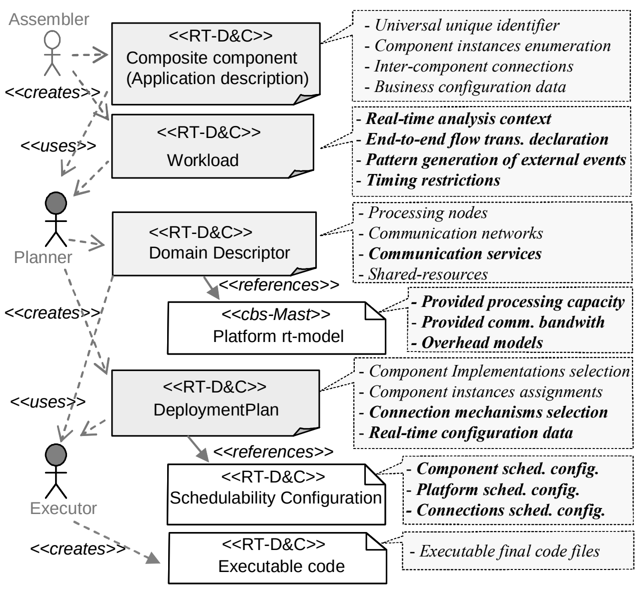
Assembler
<<creates>>
<<RT-D&C>>
Composite component
(Application description)
- Universal unique identifier
- Component instances enumeration
- Inter-component connections
- Business configuration data
<<uses>>
<<RT-D&C>>
Workload
- Real-time analysis context
- End-to-end flow trans. declaration
- Pattern generation of external events
- Timing restrictions
Planner
<<RT-D&C>>
Domain Descriptor
- Processing nodes
- Communication networks
- Communication services
- Shared-resources
<<references>>
<<creates>>
<<cbs-Mast>>
Platform rt-model
- Provided processing capacity
- Provided comm. bandwith
- Overhead models
<<uses>>
<<RT-D&C>>
DeploymentPlan
- Component Implementations selection
- Component instances assignments
- Connection mechanisms selection
- Real-time configuration data
<<references>>
Executor
<<RT-D&C>>
Schedulability Configuration
- Component sched. config.
- Platform sched. config.
- Connections sched. config.
<<creates>>
<<RT-D&C>>
Executable code
- Executable final code files

Fig. 3. Description of a real-time component-based application

obtained by the aggregation of various pieces of information: a) the description of the execution platform, (b) the deployment plan, that describes the application and the instances involved in it, and (c) the workload of the concrete mode of operation of the application to analyse. The extension to the OMG´s D&C specification on which this work relies defines and organizes as metadata the information provided by each component to build the application reactive model, in such a way that it is stored with it, and it can be reused in any application built with instances of the component. These models are composed with those of the platforms and the workload to build the full reactive model for each analysis context. Figure 2 resumes the functional and real-time metadata that are included in the description of a component when RT-D&C is used.

The model that describes the temporal behaviour of a component must include all the information related to its internal code that is required to predict the behaviour of any application in which the component may be used. It is not a complete "model" because the information it defines is not enough to predict the complete behaviour of the component services. Only in the context of an application it can be composed with the models of the rest of the components that form the application, and with the model of the platform in which the application is execute, to generate a complete real-time analysis model, from which the actual response times of the component services can be extracted. Figure 3 resumes the functional and real-time information that is included in the RT-D&C description of a real-time component-based application, from which its reactive model can be built.

The model of each component and platform resource must be formulated following a concrete real-time modelling methodology, which offers the composability features required to generate the model of the final application by composition of the models of the elements that forms it. The modelling methodology is independent of the D&C specification, although the usage of modelling standards, as MARTE, could ease the compatibility between components and platforms of independent vendors. In our case, the CBSE-MAST [9] modelling methodology is used. It implements the fundamental structure of the SAM (Schedulability Analysis Model) model in MARTE, brings parameterizable models to formulate the components behaviour, and facilitates their composition into full reactive models, able to be analized with the MAST [8] tools.

3. Schedulability design process

Evaluating the application schedulability configuration is the most specific and critical phase of the design of a real-time component-based application. As it is shown in Figure 4, this process starts from the real-time application description that is created by the assembler, that includes both the structural (Composite Component), and the reactive (Workload) points of view. The planner is the actor who deals with the schedulability design process, and as a result, he generates a deployment plan in which the schedulability parameters of the components and the platform resources have assigned values that guarantee that the timing requirements of the application are met during the execution. The design process is iterative, model-driven and assisted by tools. The planner starts each iteration with a deployment plan proposal. The *rtModelComposer* tool, based on the workload description and the information provided by the deployment plan, generates the real-time reactive model of the application by composing the partial models of the components and the platform resources, which are stored in the repository. This application real-time model is processed by real-time design tools, which evaluates the set of optimal values that must be assigned to the priorities (or other scheduling parameters) in order to make the application schedulable. The obtained schedulability configuration values must be validated by real-time analysis tools. If the application results schedulable, the process finalizes. If not, the planer must use the information obtained from the analysis tools about slacks or utilization levels of the platform resources, to propose a new deployment plan. The modelling methodology and the real-time analysis and design tools used in our approach are those available in the MAST suite, which offer a full set of techniques for event-driven distributed real-time systems.

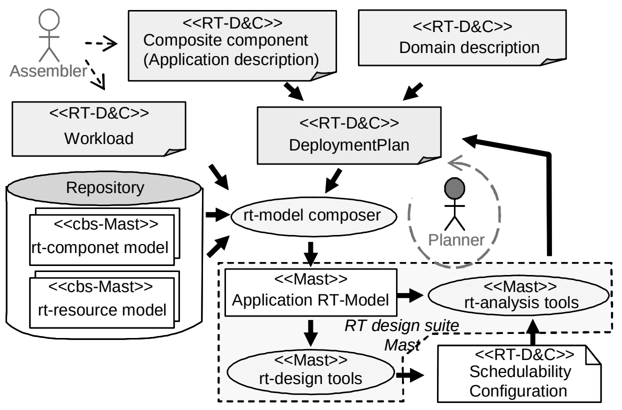

Fig. 4. Real-time design of component-based applications

4. Conclusions

D&C supports composition based on functional aspects. The development strategy and tools proposed in this work uses an extension of this specification that incorporates metadata representing the temporal behaviour of components and platforms. This allows the designers of real-time component-based applications to build their models and then analyse them using only the set of basic concepts included in the extension, without requiring expertise in the real-time modelling methodology used by the developers of the components to formulate their analysis models.

References

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

[2] A. Stankovic et al., VEST: An Aspect-Based Composition Tool for Real-Time Systems, in *Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symp.* Washington, DC, USA, 2003.

[3] G.A.Moreno, P. Merson, Model-driven performance analysis, in *Proc. of the 4th Intl. Conf. on Quality of Software Architectures, Germany,* 2008.

[4] E.Bondarev et al. CARAT: a toolkit for design and performance analysis of component-based embedded systems, Proc. Design, Automation and Test in Europe, 2007

[5] Object Management Group, Deployment and Configuration of Component-based Distributed Applications Specification, OMG doc. formal/06-04-02 (2006).

[6] P. López Martínez et al., Real-time Extensions to the OMG's Deployment and Configuration of Component-based Distributed Applications. OMG's 9th *Work.* Distributed Object Computing for Real-time and Embedded Systems, Arlington, VA, USA, 2008.

[7] Object Management Group, UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) RFP, OMG doc. ptc/07-08-04, 2007

[8] M. González Harbour et al., MAST: Modeling and Analysis Suite for Real-Time Applications, in Proc. of the *Euromicro Conference on Real-Time Systems*, June 2001.

[9] P. López, J.M. Drake, and J.L. Medina, Real-Time Modelling of Distributed Component-Based Applications, In *Proc. of 32h Euromicro Conference on Software Engineering and Advanced Applications*, Croatia, August 2006.