

# RT-D&C: A real-time extension of the OMG's Deployment and Configuration Specification

Patricia López Martínez, César Cuevas and José M. Drake

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

## Abstract

*This paper proposes an extension to the Deployment and Configuration of Component-based Distributed Applications Specification of the OMG (D&C) to support the development of applications with hard real-time requirements. The deployment of this kind of applications must include the configuration of the parameters that manage the components schedulability, in order to guarantee that the execution of the application always satisfies the specified timing constraints. New metadata about resources usage have to be available with the components to schedule the applications which use them. The extension is formulated at platform independent model (PIM) level, based only on the model of component and application introduced in the D&C, and the requirements introduced by a reactive real-time paradigm. The extension is strictly compatible with the metamodel and the process defined in the current D&C, adding some new modelling elements to the metamodel and new optional phases specific of real-time applications to the design application process.*

## 1. Introduction<sup>1</sup>

This work describes an extension of the OMG's D&C Specification [1] defined with the purpose of providing it with support for the development of component-based applications with hard real-time requirements. The D&C specification defines a metamodel that allows designing applications based on components that satisfy the three main principles of the component-based development [2]: isolation, composability, opaqueness. Our objective is proposing an extension compatible with the D&C specification, which allows designing applications with hard real-time requirements imposed in their specifications. The targeted applications are those based on the traditional reactive model of real-time systems[3]:

- The application is conceived as a set of concurrent exe-

cutions of end-to-end flow transactions that are triggered in response to external or timed events.

- The timing requirements are defined in the context of a workload model of the application, in which the event generation patterns are known.
- The timing requirements are formulated on the transactions, as temporal constraints between their internal execution states.

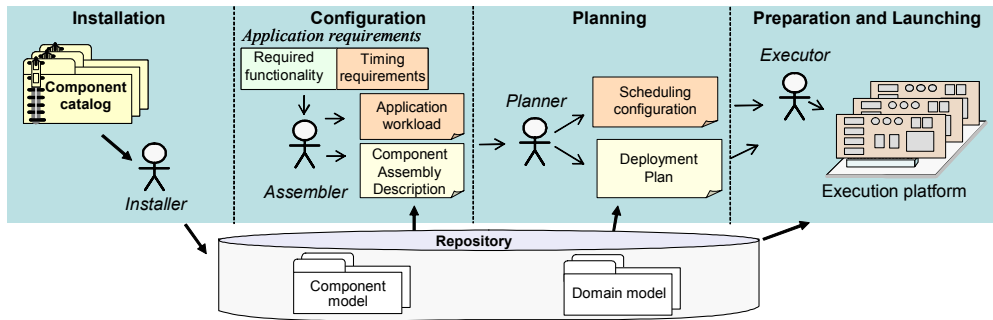
Component-based approaches differ from other software design strategies mainly in the opaqueness. During the development of an application, components must be managed without knowing any detail about their implementation or code. With that purpose, a deliverable component must include, together with its code, the models and metadata that are required to use the component in this opaque way. In a conventional application, these metadata describe the functionality, composability properties and instantiation requirements of the component. In case of a real-time application, components must include additional metadata and models that describe the parameters that control their access to the shared resources. Suitable values must be assigned to these parameters to satisfy the timing requirements of the application. This complementary information related to temporal behaviour is defined by the actors that take part in the component development process, and managed by the actors that are involved in the application development process.

The D&C real-time extension proposed, called RT-D&C, defines the real-time metadata that must be included in a deliverable component package, so that the application designers can i) predict or analyse the temporal behaviour of the applications in which the component is used, and ii) configure the platform and the component instances that form an application to guarantee that the application execution meets the specified timing requirements.

There is an important difference between the functional metadata of a component and the metadata that describes its temporal behaviour. The functionality is completely described in a self-contained way by specifying the interfaces of the component ports. On the contrary, the real-time model of a component is used to evaluate the behaviour of

---

1. This work has been funded by the EU under contracts FP7/NoE/214373 (ArtistDesign); and by the Spanish Government under grant TIN2008-06766-C03-03 (RT-MODEL). 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.** Extended D&C configuration and deployment process for a real-time component-based application

the applications in which the component is used, and not to specify the temporal behaviour of the isolated component. In general, the response times of the services offered by a component can not be independently described, since:

- They depend not only on the component's own code, but also on the behaviour of services of other components that the component uses to implement its functionality.
- They depend on the speed of the processor in which the component's code is executed, and the network through which it communicates with other components.
- They depend on the availability of the resources that the component requires to execute (processor, communication services, synchronization mechanisms, etc.), which in turn depends on the workload of the platform.

This means that the set of real-time metadata provided by a component is not a complete model. The actual response times of the services offered by a component can only be evaluated for each component instance in the context of a complete application, once the execution platform, the components that it uses to implement its functionality and the workload of the application are known.

Figure 1 shows the process of development of an application as it is defined in D&C. The new pieces of information and complementary phases introduced in this process due to the real-time nature of the applications are:

- The assembler must elaborate the complete application specification, i.e. both the functionality and the timing requirements. The assembler has access only to the components specifications, and not to the implementations, so those specifications must include real-time metadata that allow the assembler to:
  - Verify that the assembled components are composable from the real-time point of view. This means that their real-time models can be composed in order to obtain the final real-time model of the application.
  - Build the application's workload model, by identifying the end-to-end flow transactions initiated by each instance, and assigning them their corresponding activation frequencies and timing requirements.
- The planner must know how to evaluate and incorporate

to the deployment plan the schedulability properties that must be assigned to the component instances and to the platform in order to make the application execution satisfy its timing requirements. With that aim, the planner must be able to build the reactive model that describes the application temporal behaviour. Real-time analysis tools are applied to this model to calculate the set of schedulability configuration values that leads to an schedulable application.

- Based on the deployment plan, the executor must be able to configure the elements of the platform to make available the resources required to execute the application.

There are some component technologies that uses the D&C specification to configure and deploy their applications. In [4], D&C is extended to support the configuration of QoS aspects of the applications, but this configuration is assigned directly at the deployment level, without having been extracted from the analysis of the application, or from information provided by the components that form the application. Besides, it has been defined at PSM level, so it is only targeted to CORBA applications.

The rest of the paper describes and justifies the extensions introduced in the D&C specification to describe components, platforms and applications with real-time nature.

## 2. Specification and implementation of real-time components

In D&C, a `ComponentInterfaceDescription` element describes the external view of a component. It contains all the functional characteristics shared by all the implementations of the component. It constitutes the information that the assembler inspects to decide the usage of a component in an application according to its functionality, and to verify the compatibility between two components that are to be connected. The purpose of the real-time extension of this element is to provide the assembler with the information that he needs to:

- Know whether integrating the component will lead to an application for which a complete analysable model can be obtained.

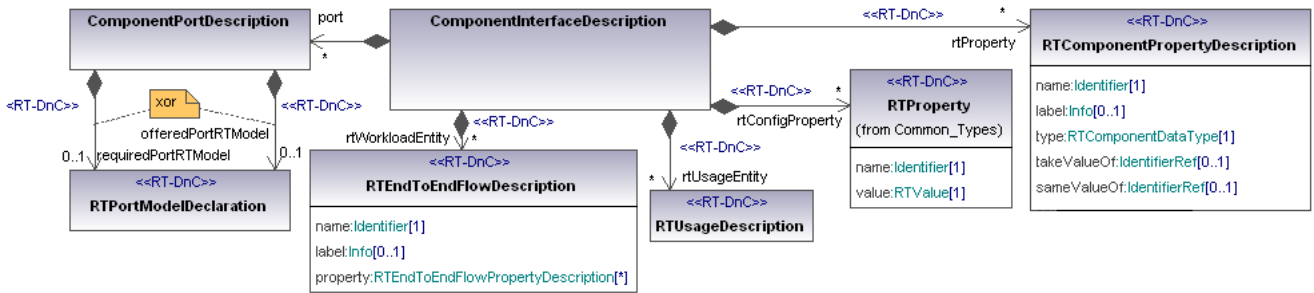


Fig. 2. RT-Metadata added to a ComponentInterfaceDescription

- Adapt the component’s real-time model to describe the temporal behaviour of the component in a specific application context.
- Add to the application workload the business end-to-end flow transactions that have their origin in the component instances.

Figure 2 shows the real-time extensions defined for the ComponentInterfaceDescription element of the D&C<sup>1</sup>:

- Specification of the composability of the component’s real-time models. These are the metadata used to confirm if an assembly of connected components has a real-time model that can be used to evaluate the temporal behaviour of the applications in which the assembly is used. It is specified through new attributes added to the component ports descriptions. For provided ports, the attribute is called *offeredPortRTModel*, and it enumerates the port services that have a timing behaviour model associated. In case of a required port, it is called *requiredPortRTModel*, and it declares the services whose real-time models are needed by the component, and therefore must be supplied by any component connected to the port.
- Configuration parameters of the real-time model of the component. They will be assigned concrete values when an instance of the component is used in an concrete application, to adapt the component’s temporal behaviour to it. They are formulated by elements of the new type RTComponentPropertyDescription, and the values to assign are of the new RTProperty type.
- Specification of the workload that can be initiated in the component. By means of elements of the new RTEndToEndFlowDescription type, the end-to-end flow transactions that the component can initiate in an application are defined. The assembler will use these metadata to define the set of transactions with which each instance contributes to the application workload. The transactions can declare parameters, through which the concrete frequency of activation, the timing requirements they must meet or the activities that they involve, can be assigned

in the context of a concrete application.

The description of a component implementation is formulated in D&C by means of a ComponentImplementationDescription element. Figure 3 shows the real-time extensions defined at this level:

- Real-time model of the component. The developer, who elaborates a concrete implementation of a component interface, knows the details of the internal code, so he must also elaborate the model that describes all the characteristics about it that are required to evaluate the temporal behaviour of the application in which the component is used. This model must be formulated following a concrete 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 (component instances and platform resources). The modelling methodology is independent of the D&C specification, although the usage of modelling standards, as MARTE [3], could ease the compatibility between components and platforms of independent vendors. The real-time model of each component can be accessed by the composition tools (or by the planner) through the new attribute *rtModel* defined in the MonolithicImplementationDescription element. This model must be in accordance with the characteristics exposed in the component interface, i.e. it must offer models of the services enumerated in the offeredPortRTModel elements, and models for the transactions declared in the interface. It must also have the parameters defined at the interface level.
- Specification of the schedulability configuration parameters of the component. In case of real-time components, a new set of configuration parameters appears at implementation level. They do not affect the functionality of the component, but the way in which the platform resources used by the component must be managed to guarantee the schedulability of the application execution. These kind of parameters, e.g. priorities, schedulability policies, priority ceilings, etc. depend on the way in which the component is implemented and on the platform characteristics. Because of that, their declaration is made at implementation level, and not in the component

1. The pictures in the paper only shows the real-time extensions added to the D&C modelling elements, and not their complete structure.



Fig. 3. RT-Metadata added to a ComponentImplementationDescription

interface, as it occurs with purely functional properties. As a consequence, they will be assigned by the planner. These properties are defined through the new *specificProperty* attribute.

- Specification of new parameters of the real-time model. The properties of the real-time model of a component can also be affected by the way in which it is implemented. So, new real-time configuration properties can be declared in the component implementation.

### 3. Description of a real-time domain

The capacity and characteristics of the platforms are required to evaluate the temporal behaviour of the software components that execute on it. So, the real-time models of all the resources that form the execution platform are needed to analyse the behaviour of an application. As it occurs with the components, these models can be parameterized, and hence, reusable. In that way, configurable platforms can be defined.

The current D&C specification does not formalize any modelling element to store information about platforms. A platform, called Domain in D&C, is conceived and described directly as an instance, i.e. as the set of concrete nodes and interconnects on which the application is to be deployed. So, the real-time extensions proposed for the D&C's Target Data Model consists in defining new modelling elements that allow storing information about the real-time models of the resources that can form an execution platform. With the proposed extensions, each node, interconnect or shared resource instance declared in a domain can reference, through the new *source* attribute, its corresponding descriptor. A domain element descriptor, as the one shown in Figure 4 for the case of a node, includes a reference to the real-time model of the element (*rtModel* attribute), and declares the set of configurable parameters of that model, through elements of type RT<Ele-

ment>PropertyDescription. As it happens with component instances, concrete values can be assigned through the attribute *rtConfigProperty* to those parameters, for each node, network or resource instance declared in a domain.

Although the purpose of D&C is supporting the deployment of component-based applications in heterogeneous platforms, it does not define any formalism to specify the concrete mechanism to use for a connection between components. Connections are conceived always as direct connections, implemented on top of a concrete middleware. So, there is no possibility to describe connections through different mechanisms in the same application. The extension proposed allows defining the concrete mechanism to use for each connection between components. The way in which the connection characteristics are defined is explained in the next section, since they are assigned in the deployment plan. However, there is an important aspect to remark at this level: the introduction of the models that describe the behaviour of the local or remote invocations between components is essential in the construction of the real-time model of a component-based application. So, for each mechanism offered by the platform to implement connections, the model of the platform must include its temporal behaviour model. A new type of element, shown in Figure 4, may be included in a platform description, called ConnectionMechanismDescription, which describes the main characteristics, both functional and real-time, of each communication mechanism supported by the platform.

The work in [5] proposes also an adaptation of the D&C to support heterogeneous connections. They also qualify each component connection, but at the assembly description level, i.e. when the application is defined as a set of connected components. As it is detailed in section 4, our approach qualifies the connection at deployment plan, when the platform is known and the planner can choose among the available communication mechanisms.

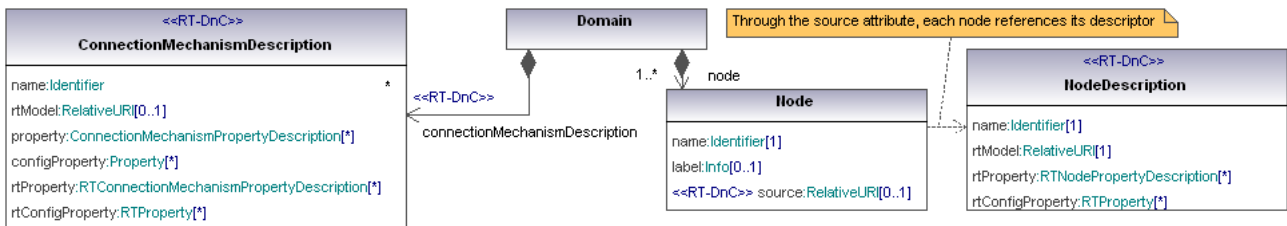


Fig. 4. RT-Metadata added to the Domain descriptor

#### 4. Specification and deployment of real-time applications

The phases involved in the application design and deployment process defined by D&C were shown in Figure 1. When the application has real-time requirements, new tasks must be performed in each phase, so the D&C must be extended in order to include information to support them. These additional tasks are:

- The application specification must be completed with the definition of the real-time requirements imposed on its execution. The assembler, together with the description of the application as an assembly of component instances that implements the required functionality, must describe the timing requirements that must be fulfilled during the application execution. Following the real-time paradigm in which this proposal is based, this means defining three aspects: i) The reactive model, i. e. identifying the set of end-to-end flow transactions that have timing requirements, ii) The workload, i.e. the frequency and the activation patterns of these transactions, and iii) the timing requirements imposed on the transactions, formulated as timing constraints between their internal states. All these aspects constitute the AnalysisContext[3], or Real-Time Situation[6], which constitutes the scenario in which the real-time model of the application must be analysed.
- Configuration of the parameters that controls the application schedulability. In a conventional application, the assembler must assign values to the configuration parameters of the components to adapt them to the application functionality. However, the assembler has no capacity to assign the schedulability configuration parameters that appear in a real-time application, since i) they are specific to the implementations, ii) they depend on the platform, and iii) a global analysis of the application is required to calculate their values. Therefore, the schedulability configuration is responsibility of the planner, who does have this information. The tasks that the planner must tackle are complex. First, he has to build the complete reactive model of the application, using the models of the components and the platform resources. Then, he must apply algorithms or analysis and design tools [7][8] on this model to obtain suitable values for each schedulability parameter of the application.
- Modelling of the mechanisms used for each connection between components. As it was explained before, the planner can specify the concrete mechanism to use for each connection, referencing and configuring the model that describes its temporal behaviour in order to estimate its influence in the application.
- Configuration of the domain for the application execution. The preparation and launching phases of the

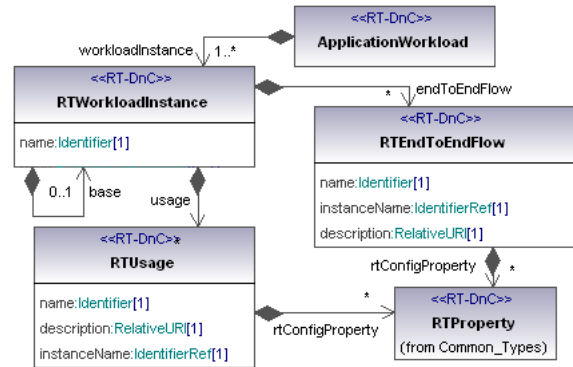


Fig. 5. New Application Workload Model defined in RT-D&C

deployment usually consists in verifying code availability, instantiating, configuring and connecting the components, and launching the application. In a real-time system the code availability is not enough. To guarantee that the timing requirements are met, the platform resources must be available when they are required by the application. With that purpose, the deployment plan must include the configuration values that the executor must set or negotiate with each platform resource (node, network, etc.) to guarantee the availability of the required resources.

Since the D&C specification does not deal with real-time, it does not define any element to describe application workloads. Figure 5 shows the modelling elements proposed in the RT-D&C extension to describe the workloads that an application may exhibit. The workloads are basically lists of transaction instances, each of them qualified with specific characteristics (rtConfigProperty attribute). The parameterization of the end-to-end flow transactions is an essential aspect for the definition of workloads in component-based applications, since it facilitates defining the transactions in the timing model of the components. They are defined there as templates, i.e. leaving some characteristics as unsolved parameters, which will be defined when an instance of the transaction is declared in the context of the workload of an application. These parameters correspond to those declared in the template of the transaction included in the external interface of the component.

The D&C specification defines in detail deployment plans. They include in a self-contained way, all the information required by the executor to deploy, configure and execute the application in the corresponding platform. However, to give support for these new tasks, the following extensions, shown in Figure 6, have been defined in a deployment plan descriptor:

- Access to the real-time models: The planner must have access to the metadata that describe the temporal behaviour of the application, so that he can evaluate, directly or assisted by tools, the optimal values to assign to the

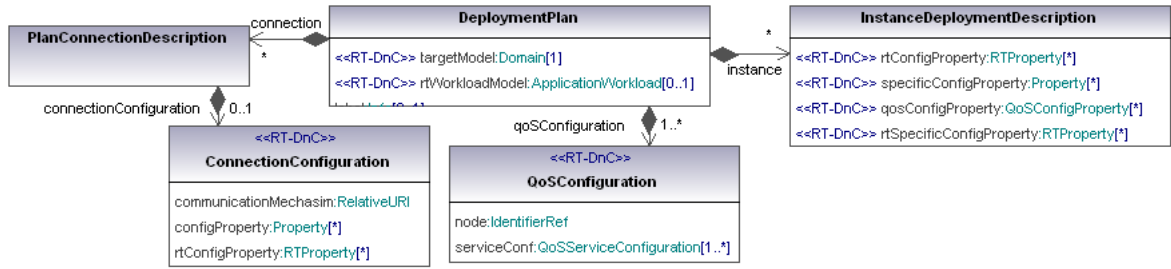


Fig. 6. RT-Metadata added to a DeploymentPlan

schedulability parameters. In a component-based application, the real-time model is built by integrating pieces, distributed among the different elements that form the application. First, the models of the software components, which are referenced in the component descriptors themselves. These descriptors are referenced from the InstanceDeploymentDescription elements, which are used to describe instances in a deployment plan. The model of a component is a parameterized model, so it has to be configured for each component instance. With that purpose, two new properties have been added to an InstanceDeploymentDescription, *rtConfigProperty* and *rtSpecificConfigProperty*, which are used to assign values to the parameters of the corresponding real-time model. Second, the real-time models of the domain elements, that can be accessed through the property *targetModel* added to the DeploymentPlan element. Third, the real-time models that describe the temporal behaviour of the mechanisms chosen for the components interactions, which can be accessed through the new attribute of type ConnectionConfiguration with which each connection can be configured. Through this element the corresponding descriptor of the mechanism (that must be declared in the domain) is referenced, and in consequence its real-time model can be accessed and configured. The last element required to generate the real-time model of the application is the workload model, which is referenced through the new property *rtWorkloadModel* defined in the deployment plan.

- Configuration of the schedulability parameters. From the real-time design point of view, the duty of the planner consists in generating the set of schedulability configuration values that guarantee an execution of the application that satisfies all its timing requirements. The schedulability configuration concerns the software components, the communication mechanisms, and the domain resources. Some extensions have been defined in RT-D&C to support the assignment of these values in the deployment plan, so they can be used later in the preparation and launching phases. The schedulability configuration values of the component instances are assigned in the corresponding InstanceDeploymentDescription element. If the parameter has been defined at component implementa-

tion level, it is assigned through the *specificConfigProperty* attribute. If it corresponds to parameters assigned by default in a concrete technology, they are assigned through the *qosConfigProperty* attribute. The assignment of values to the communication mechanisms is made through the property *configProperty* of the ConnectionConfiguration element. Finally, the schedulability configuration of the domain is defined by means of QoSConfiguration elements, which defines the configuration of each of the elements of the domain.

At PIM level, the types of the values used to configure the schedulability are not known, they depend on the concrete platform and technology. So the proposed extension has been introduced in an abstract way, in the sense that the types used to assign schedulability properties, as QoSConfigProperty or QoSConfiguration, are defined as abstract elements, which must be concreted in the PSM model that must be defined for a concrete technology. Besides, this kind of abstract definition allows using the same structure to configure other types of non-functional properties.

## 5. Example of RT-D&C usage

An example of usage of the proposed extensions is explained in this section using the SCADA (Supervisory Control and Data Acquisition) application shown in Figure 7. The application is composed of four components, organized in a three-layer architecture. The ScadaManager component constitutes the user interface, through which the operator can order new supervision tasks or visualize the stored data. The ScadaEngine component is the intermediate element. Its function consists in supervising periodically a set of physical magnitudes, which it reads by means of acquisition cards managed by IOCard components. With a lower frequency, it also stores average values about the

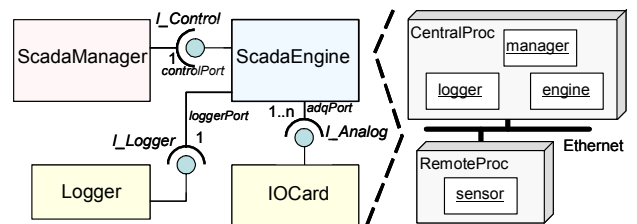


Fig. 7. Scada Application example

read magnitudes. Figure 7 shows also a possible deployment for the application. There is an instance of each component, and the domain is form of two nodes, CentralProc and RemoteProc, executing the MaRTE OS operating system, and connected through an Ethernet network using the real-time protocol RT-EP. Its reactive model is constituted by four transactions:

- thePollingTrans: It is triggered periodically ( $T_p=5$  ms) in the engine instance. It reads the supervised magnitudes and maintains an statistic about them. It has a temporal deadline equal to its period.
- theLoggingTrans: It is triggered periodically ( $T_l=1000$  ms) in the engine instance. It stores the average values in the logger instance. It has a deadline equal to the period.
- theMonitoringTrans: It is triggered periodically ( $T_m=1000$  ms) in the manager instance. It reads the value of certain magnitude from the engine instance and shows it to the operator. Its deadline is equal to the period.
- theCommandTrans: It is triggered sporadically in the manager instance. It attends the keyboard commands, and accesses the engine component to modify its operation mode. It has no timing requirements.

Figure 8 shows some of the elements used in the ScadaEngine component description using the RT-D&C specification. First of all, in the ComponentInterfaceDescription, the component includes its real-time composability properties. The figure shows only the requirements attached to the required port adqPort. Any component connected to that port must provide a real-time model for the ai\_ReadCode service in order to be able to build the real-time model of a ScadaEngine instance. Besides, the component configuration properties are defined. Together with the pollingPeriod property, which defines the period used for the supervision task, a real-time property, called rtmPollingPeriod, is defined. This property is directly mapped to the corresponding value of the pollingPeriod property. This real-time property is defined at this level since for any possible implementation of the component, this period will influence its temporal behaviour. The periodic activity that executes the supervision is modelled as a transaction that the component can initiate in an application. Its descriptor is called PollingTrans and it declares a parameter, pollingDeadline, which represents the deadline with which the transaction must be executed.

Figure 8 shows also a part of the descriptor of a concrete implementation of the ScadaEngine interface. The component is called AdaScadaEngine and it has been developed according to the Ada-CCM component technology [9]. This technology defines its own PSM for the RT-D&C specification, but it is not explained here since it would make the example too complex. The descriptor includes the

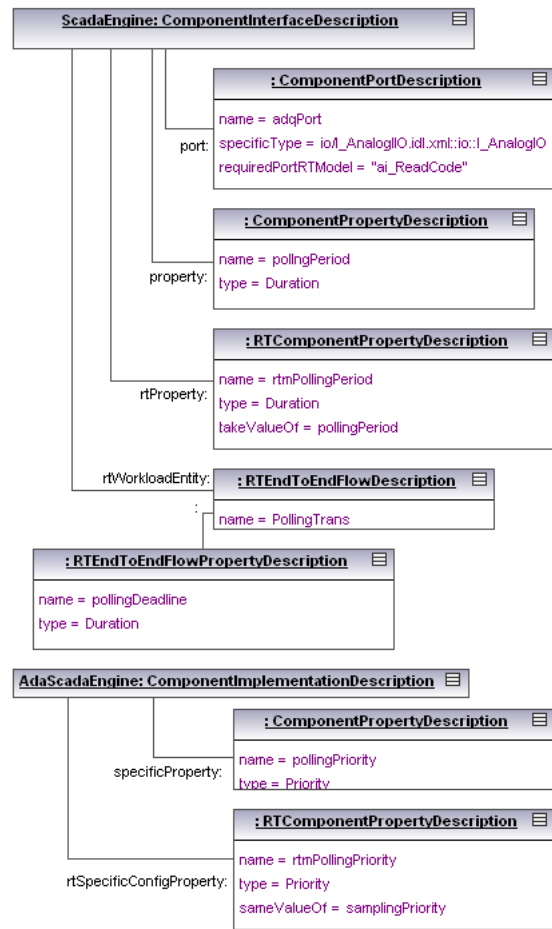


Fig. 8. RT-metadata of a ScadaEngine component.

reference to the component’s real-time model, which in Ada-CCM is developed according to the CBSE-MAST [10] modelling methodology. This is a modular modelling strategy for formulating the temporal behaviour of component-based systems based on the MAST methodology [6]. It allows obtaining the MAST model of a CBSE application by composition of the models of the elements that form it. This component is implemented by means of two threads, which execute the polling and the logging activities, so new schedulability properties appear. The figure only shows those related to the polling activity. An schedulability configuration property, called pollingPriority, is defined to make the thread’s priority configurable. It is also a parameter of the real-time model, so it is declared as a real-time property, rtmPollingPriority. These properties are related by means of the “sameValueOf” attribute, meaning that the value of the rtmPollingPriority property can be calculated by analysis tools, and in that case the obtained value must be assigned to the pollingPriority property.

When the assembler defines the application, he declares the instances that form it and configures each of them. This information is directly translated to the deployment plan by

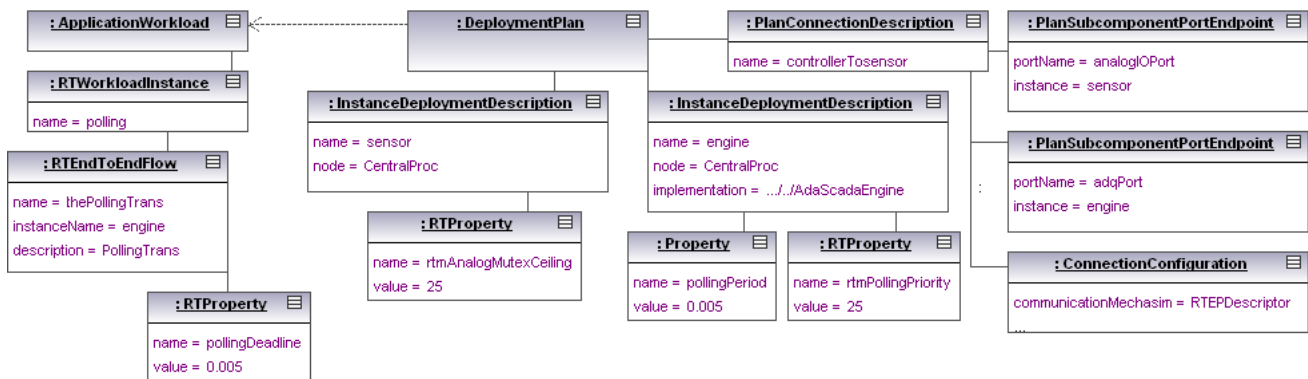


Fig. 9. RT-metadata in the deployment plan of the Scada application

the planner. Figure 9 shows a partial vision of the deployment plan of the example, in which only the declaration of the engine and sensor instances are shown. A period of 0.005 is assigned to the pollingPeriod property of the engine instance. The assembler also defines the workload of the application. The figure shows the definition of the thePollingTrans transaction, whose pollingDeadline parameter receives also a value of 0.005.

Once the planner has defined the execution platform, he must assign values to the properties related to schedulability (the ones defined at component implementation level). In this case, a value of 25 is assigned to the rtmPollingPriority property (it is not required to assign values to the rtmPollingPeriod or pollingPriority properties since they are related to other properties in the component descriptors). The planner also has to define the communication mechanism to use for each remote connection between components, in this case he chooses a communication based on the RTEP protocol, which is supported in the Ada-CCM technology. Its descriptor is supposed to be included in the platform descriptor, in order to be able to access to its real-time model. With the deployment plan, the workload model and the platform model, the composition tool can generate the final analysis model of the application. It is used as input of the set of tools that the MAST suite offers for real-time analysis, and as a result, the set of optimal values for the schedulability configuration properties can be obtained and re-assigned in the deployment plan, configuring the application for meeting its real-time requirements. For the part of example shown in this paper, optimal values for the pollingPriority and digitalMutexCeiling could be obtained from the analysis.

## 6. Conclusions

The current version of the OMG's D&C specification gives support only to functional aspects. This work proposes an extension of this specification to incorporate metadata about the temporal behaviour of components and platforms, which allows the designers of real-time compo-

nent-based applications to analyse their temporal behaviour, and to extract from this analysis the schedulability configuration parameters to be assigned to the component instances and to the platform in order to guarantee the fulfillment of the timing requirements of the application.

## References

- [1] Object Management Group, Deployment and Configuration of Component-based Distributed Applications Specification, OMG doc. formal/06-04-02 (2006).
- [2] I. Crnkovic, and M. Larsson, Building Reliable Component-Based Software Systems, Artech House Publishers, 2002.
- [3] Object Management Group, UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) Beta 2, OMG doc. ptc/08-06-09, 2008
- [4] G. Deng, C. Gill, D. C. Schmidt, N.Wang, QoS-enabled Component Middleware for Distributed Real-Time and Embedded Systems, Handbook of Real-Time and Embedded Systems (I. Lee, J. Leung, and S. Son, ed.), CRC Press, 2007
- [5] L. Bulej, T. Bures, Using Connectors for Deployment of Heterogeneous Applications in the Context of OMG D&C Specification, in Proc. 1st Intl. Conf. Interoperability of Enterprise Software and Applications, Switzerland, Feb 2005
- [6] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M.Drake, MAST: Modeling and Analysis Suite for Real-Time Applications, in Proc. of the *Euromicro Conference on Real-Time Systems*, June 2001. <http://mast.unican.es/>
- [7] M.H. Klein, T.Ralya, B. Pollack, R. Obenza M. González Harbour, A practitioner's handbook for Real-Time Analysis, Kluwer Academic Publishers Group, 1993.
- [8] J. Liu, Real-Time Systems. ISBN 0-13-099651-3. Prentice Hall Inc., 2000
- [9] P. López, J.M. Drake, P. Pacheco, and J.L. Medina, An Ada 2005 Technology for Distributed and Real-Time Component-based Applications, in Proc. of the *13th Intl. Conference on Reliable Software Technologies Ada-Europe*, Venice, 2008
- [10] 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