UML/MARTE methodology for modelling of Mixed-Criticality Systems

Microelectronics Engineering Group
of the
University of Cantabria

Authors: F. Herrera, P. Peñil, J.Medina, and E. Villar

Date: 2015, April

Content

1 Introduction

This document presents how to model mixed-criticality systems in the UML/MARTE modelling methodology of the Microelectronics Engineering Group of the University of Cantabria.

Mixed criticality is a natural trend of complex systems once they are integrating more and more functionality and cost requirements get more demanding. The functionalities integrates are of different nature and with different requirements in performance, energy efficiency and cost. Moreover, safety requirements get into the play and the set of constraints associated to the different requirements have not the same criticality.

The first version of this document is direct result of the work developed in the context of the CONTREX project. The methodology, relies on the MARTE meta model and the extension defined CONTREX. Moreover, as for that extension, it has been developed taking into account the modelling needs exposed by CONTREX partners, the state-of-art on mixed-criticality systems, and a survey of several safety standards.

The document presents first in Section 2 the modelling needs which have been detected.

Then, in Section 3 presents the modelling techniques enabled for covering, at least, the aforementioned modelling needs.

2 Modelling Needs for Mixed-Criticality

The following needs for mixed-criticality modelling are devised in the methodology:

- the association to an extra-functional property of a set of potentially different values, each one corresponding to a given level of criticality,
- the association of a criticality-level to requirements and contracts,
- the association of a criticality-level directly to a modelling element, e.g an application component, or a platform resource.

The former case is the case of recent real-time analysis extensions to MCS. For instance, new schedulability analysis have been developed which required the association of several WCET figures to the same task. Each WCET figure correspond to a given criticality level.

The second case, i.e. the association of criticality-levels to requirements and contracts is funded in the fact that not all the requirements and/or contracts have the same importance in the correct design of a system. The importance of each constraint/contract depends on the system and on the design scenario (i.e. a system can be targeted to work in different environments). Notice that the association of criticality levels-to requirements is quite generic. Criticalities can refer to requirements of the same type of constraints (e.g. time constraints, on a deadline and on the throughput) or not (e.g. a latency and the power consumption); and to different applications or not.

The need to associate criticalities directly to modelling elements obeys to practical modelling practices. In the same way that the association of a criticality to a “task” modelling element simplifies in reference work the manner to state actually the criticality of fulfilling a deadline constraint on the task, a criticality can refer also to an application component in the methodology. The association of a criticality constraint to a platform component, i.e. a resource of the platform, is also a practical nice, once safety standards use a criticality level to impose requirements on how that component has to be developed.

3 Mixed-Criticality Modelling Techniques

The modelling techniques available for covering the modelling needs introduced in Section 2 consist in enabling the association of levels of criticality to the following modelling elements:

- value annotations of non-functional properties,
- constraints on non-functional properties,
- application or platform components

These modelling techniques rely on the two extensions proposed in the CONTREX metamodel reported in D2.1.1 [2]. The following sections detail how the modelling techniques are applied in for covering the afore introduced modelling needs.

3.1 Criticality-based Values for extra-functional properties

In classical schedulability analysis techniques, the worst-case execution time (WCET), a typical non-functional property, is associated to each piece of functionality or task. Some methodologies require the association of a list of WCETs to each task. For instance, some approaches [3][4] provide analysis and design techniques for heterogeneous multi-core platforms, which require the annotation of several WCET values for each task in the input model. There, each WCET value represents the worst-case execution time required to execute a piece of functionality in a specific processing resource.

Similarly, in recently proposed schedulability analysis techniques for mixed-criticality systems [5], the input model requires the annotation of several WCETs values for each task. However, in this case, what distinguishes each WCET on the list is that it corresponds to a specific level-of-criticality. For these, and even more generic modelling cases, the capability to associate levels of criticality to value annotations of non-functional properties is required.

The modelling methodology covers all the modelling needs of thus combining the modelling needs of [3][4] [5]**.** Figure 1 illustrates the capability of the methodology to annotate different WCETs considering both the processing element where the functionality is computed, and the criticality. This way, the technique enables the capture of a WCET matrix, required for analysis methods capable to consider both heterogeneous platforms and mixed-criticalites, In Figure 1 example, UML associations with the <<resourceUsage>> stereotype are used to specify the time workloads corresponding to the execution of an application functionality (encapsulated within an application component) on a specific processing element.

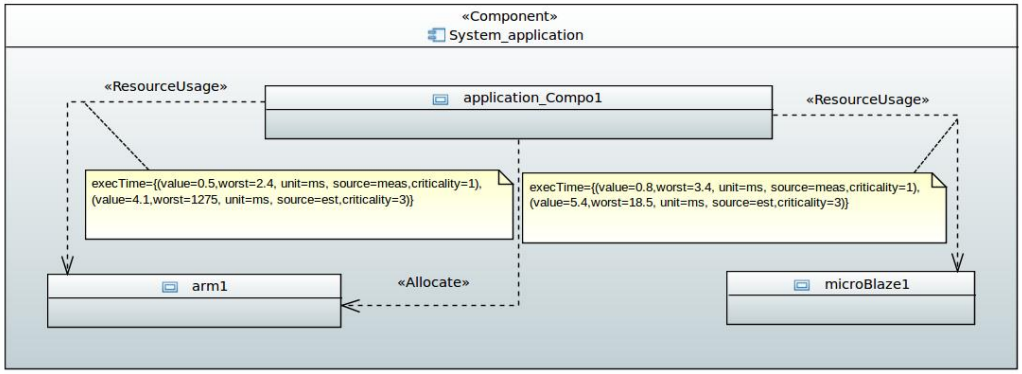

Figure 1. Criticality associated to an application model.

3.1.1 Annotation of different time values depending on the allocation

The annotation of a time property associated to the application component is done via an UML association stereotyped with the MARTE <<resourceUsage>> stereotype. The association points to a processing resource element, e.g. "arm" in Figure 1. This stereotype provides the "execTime" attribute, of *NFP_Duration* type, which enables the capture of the execution time of application component functionality on the pointed processing resource. Therefore, the type of time annotation can be, for instance, an average, a worst-case value or even both types of values can be annotated: "(value=…, worst=…)"

The annotation of different worst-case execution times, WCETs, each corresponding to the mapping of the application component functionality to a specific type of processing resource, is supported by capturing several <<resourceUsage>> allocations [1].

Figure 1 also shows, although the annotation of execution times for different types of processing resource is done via UML associations, the shall not be confused with the the actual allocation of the component functionality to the processing resource, performed with the <<allocate>> association.

3.1.2 Annotation of different time values depending on the criticality

Now, it can be stated how to model the association of different time values depending on the criticality for a given <<ResourceUsage>> "application component – processing resource" association. It can be done for each <<ResourceUsage>> "application component – processing resource" association, which yields a matrix of time annotations depending on the processing resource the functionality is allocated, and on the criticality.

For it, the "execTime" attribute is given a list of values, instead of a single value. The format of the list of values is *"{(value1), (value2), … (valueN)}"*, where in turn, each value between "(…)" can contain a list of attributes. Such list of attributes is defined for

[1] Similarly can be said for average time values.

the MARTE *NFP_Duration* type. *NFP_Duration* inherits NFP_CommonType, and thus its attributed can be used. The extended *NFP_CommonType* defined in [2] and reproduced in Figure 2[2] is used.

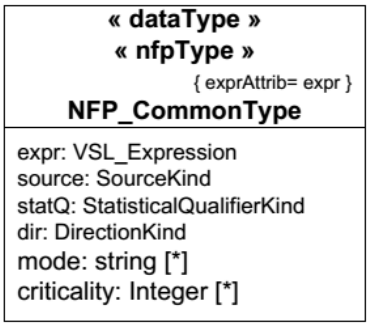

Figure 2. First extension of MARTE metamodel proposed.

It enables to load the "exec_time" attribute with a list of the following type:

"{(…,worst=2.4, criticality=1,…), (…,worst=1275, criticality=3,…), … ()}".

The enabled mechanism is sufficiently generic to support modelling cases requiring even richer annotations. For instance, the criticality can be associated to average times (using "value" attribute) or to a group of annotated values. For instance, in Figure 1 example, through the expression "(…, value=…, worst=…, criticality=…)", a criticality is associated to both, the average and the worst execution time.

3.2 Criticalities for constraints on non-functional properties

A main modelling need is the support of specification of levels of criticality for requirements and contracts, and specifically requirements and contracts on extra-functional properties.

Requirements can be understood as Boolean expressions which can refer both to functional and to extra-functional properties. The latter, that is, the support of requirements on extra-functional (called non-functional in the MARTE specification) properties is especially important in CONTREX, which will provide mechanisms for analysis and design under consideration of extra-functional properties related to time, power consumption, temperature, and reliability. Moreover, CONTREX is committed to enable the design of mixed-criticality systems, which in this context means that it will be possible to assign different "levels of importance" to the different extra-functional requirements.

[2] Here is where a first extension of MARTE proposed in [2] is exploited. The extension is reproduced in Figure 2 for reader convenience, and consists in a new attribute named criticality in the *NFP_CommonType*. Notice also that the extension of the *NFP_CommonType* involves a MARTE extension, but not an extension of the MARTE profile itself. The impact of this MARTE extension is on the tools which have to parse and/or edit the expression, and that have to select the values according to the level of criticality in the corresponding modelling case (timing or schedulability analysis, allocation, etc).

The CONTREX UML/MARTE modelling methodology will provide a very versatile and generic mechanism based on the extension proposed in D2.1.1. The extension is reminded here for convenience and it is shown in Figure 3. It consists in adding a "criticality" attribute to the MARTE *NFP_Constraint* stereotype.

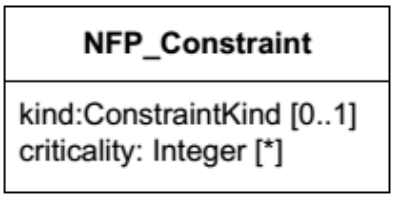

Figure 3 . Second extension for criticality proposed in D2.1.1.

Figure 4 shows an example[3] where the criticality refers to an expression made explicit in the constraint. The expression ("$latency==(1,s)") is expressed in MARTE VSL and reflects a requirement on an extra-functional property, i.e. on a latency, which shall be in the scope of the modelling element referred by the NFP constraint. The "$" states that the latency attribute is a parameter, e.g. that it can be calculated by an underlying analysis tool, instead of being a direct, manually annotated input to the model. Therefore, in Figure 4 example, the full semantics of the NFP constraint states that the latency of the "System_application" component has to be 1s, and that the criticality level associated to such a constraint is 4. That is, the criticality is assigned to the constraint, not to the component (this other case is explained in Section 3.3).

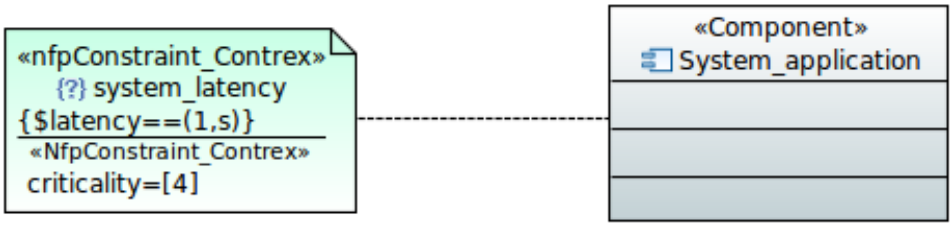

Figure 4. Criticality associated to a performance constraint.

Notice that, in the same way that the constraint fulfilment might depend on the accuracy of the underlying analysis tool to calculate the latency, the interpretation of the meaning of that criticality level is methodology dependent. That is, it depends on the domain and concrete safety standard or certification process that is planned to be faced later on. Also the use of VSL to express the constraint is a design choice; the methodology allows the use of natural language or some other forms of constraints adapted to the modeller needs.

[3] Notice that in the examples a stereotype called <<nfpConstraint_Contrex>> is used. Currently, the proposed extensions have been implemented in a CONTREX specific profile. The long term intent, as it was proposed in the deliverable D2.1.1, Section 4.1, is the actual extension of MARTE, such the "criticality" attribute, stating the criticality level belongs to the MARTE <<nfpConstraint>> stereotype.

The specification of the constraint (Boolean expressions) in the body of the NFP_Contraint annotated with a concrete criticality may serve the purpose of transposing such domain defined criticality semantics into specific requirements in the model.

As mentioned at the beginning of the Section, another modelling need foreseen is the association of criticality to contracts. In its most generic understanding the contract can be understood as a pair of Boolean expressions, of the type "assumption" and "guarantee" linked by an "implies" relationship:

$$\text{Assumption} \rightarrow \text{Guarantee}$$

Moreover, in a contract typically, two parts are involved in such way that one part sources the variables which control the fulfilment of the assumption, while the other sources the variables which source the guarantee.

This means that different types of contracts can appear depending on the sides involved, e.g. two application components, one application component and a resource component, one parent component with its children, etc. Because of that, the modelling mechanism has to be sufficiently generic to support the different types of contracts. The extended NFP constraint proposed is capable to cover this generic modelling of contracts.

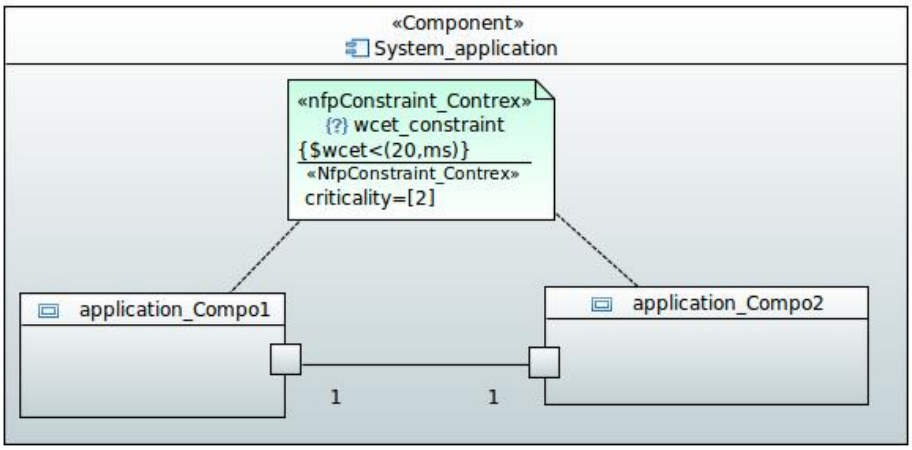

Figure 5. Criticality associated to several modelling elements, specifically to two application components.

There are scenarios where it can be interesting the association of the same requirement to several modelling elements. Figure 5 shows a case where the same constraint with the same criticality affects several modelling instances.

Another consideration is that the criticality can be specified either for individual instances, or for all instances of a classifier. In the examples shown so far, the criticality is associated via note links to instances of the modelling elements the constraints refer to. It is possible to associate the NFP constraint to a classifier. In such a case, the criticality becomes an attribute of the class definition, and therefore of all its instances.

The specification of the constraints may be related to any functional or extra-functional quality of the elements annotated, so they may be related to timing properties as well as

to power consumption, temperature, heat dissipation, space, memory or any other extra-functional property.

3.3 Criticalities directly on modelling elements

As enumerated at the beginning of the Section, another detected modelling need is the association of levels-of-criticality directly to modelling elements.

At least two modelling cases of interest have been identified. A first scenario is the verifiable deployment of application elements to platform resources. That is, in certain contexts, it is expected that a methodologies assigns a criticality levels both, to the application components and to the platform resources. In the former case, the criticality of application components would stand for a "required" criticality. In the latter case, the criticality would have a semantic more related to the criticality which the platform resource can support. From this information, analyses checking the coherence of the deployment, e.g. to check if a given application component can be allocated to a second component is enabled.

Another scenario is the support of simplified modelling of mixed-criticality models. That is, to make it possible to build more synthetic models, and more agile modelling methodologies, whenever the criticality levels which are assigned correspond to well-known restrictions or contracts which can be implicitly derived from the association to the modelling element, Figure 6 provides a basic example of this modelling case.

In Figure 6 case, the criticality level is associated to an application component. As shown, the modelling mechanism consists in the association of a UML constraint with no specification, stereotyped with a MARTE NFP constraint, to the modelling element which the criticality refers to. The kind of constraint may be "offered" or "required" according to the modelling needs, but to avoid confusion, it should NOT be of the kind "contract".

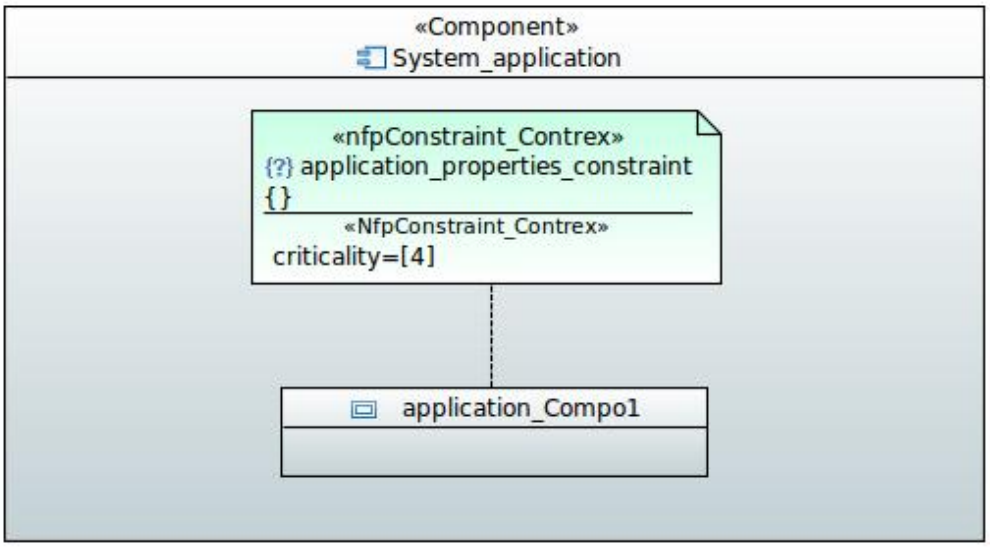

Figure 6. Criticality associated to a modelling element, specifically an application component.

This simple mechanism will enable, for instance, a simplified capture of mixed-criticality application models, e.g. a set of independent and periodic tasks, where each task has associated a deadline constraint. This can be done in MARTE by stereotyping the application component with the <<RtSpecification>> stereotype, which provides the "deadline" attribute. Then, as long as the UML constraint has no expression, the modelling methodology can interpret that the "criticality" associated to the application component indirectly and implicitly refers to the deadline constraint associated to the application component. This way, there is no need to write and repeat the expression of the deadline constraint for each NFP constraint associated to the application component and containing the criticality attribute.

All these methodological proposals will be essayed along the modelling cases in the CONTREX use cases and in ad-hoc experiments and a refined version with more complete examples will be included in the final version of this document.

4 References

[1] D.Quaglia et al. “CONTREX System modelling methodology (preliminary)”. ”. Deliverable D2.1.1 of the FP7 CONTREX project. Sept. 2014.

[2] J. Medina. “CONTREX System Metamodel”. Deliverable D2.1.1 of the FP7 CONTREX project.

[3] A. Kumar and T. Srikanthan, “Accelerating throughput-aware run-time mapping for heterogeneous MPSoCs,” ACM Transactions on Design Automation of Electronic Systems (TODAES), 2012

[4] S. Stuijk, T. Basten, M. C. W. Geilen, and H. Corporaal. 2007. Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs. In Proceedings of the 44th annual Design Automation Conference (DAC '07). ACM, New York, NY, USA, 777-782. DOI=10.1145/1278480.1278674 http://doi.acm.org/10.1145/1278480.1278674.

[5] S. Baruah, H. Li, and L. Stougie, “Towards the design of certifiable mixed-criticality systems,” in Proc. of RTAS, 2010.