Bridging the gap between practical cases and temporal performance analysis: a models repository-based approach

Anh-Toan Bui Long
LIAS / ISAE - ENSMA
Futuroscope, France
builonga@ensma.fr

Yassine Ouhammou
LIAS / ISAE - ENSMA
Futuroscope, France
yassine.ouhammou@ensma.fr

Emmanuel Grolleau
LIAS / ISAE - ENSMA
Futuroscope, France
grolleau@ensma.fr

Loïc Fejoz
RealTime-at-Work
Nancy, France
loic.fejoz@realtimeatwork.com

Laurent Rioux
Thales Research & Technology
Palaiseau, France
laurent.rioux@thalesgroup.com

ABSTRACT

Despite various analysis models and tests proposed by the real-time community to validate the temporal performance of hard real-time systems, several cases derived from industrial practices do not find their corresponding tests in standard schedulability analysis tools. They need to be transformed and adapted in terms of qualitative and quantitative information while remaining conservative. The current state-of-practice of transformation relies solely on designers expertise. As a result, the adaptation/transformation task tends to be time- and effort-intensive.

This papers proposes to capitalize adaptation efforts by storing them in a repository, hence designers can be helped to automatically transform and adapt their practical designs to analyzable models while staying conservative. Thanks to model-driven engineering settings the capitalization is illustrated by the implementation of a repository as a decision support and the automatic adaptation relies on model-based transformations. Also, a case study is presented to stress the importance of our contribution.

CCS CONCEPTS

• **Computer systems organization → Real-time system specification**; **Real-time system architecture**; • **Software and its engineering → Scheduling**; *Process validation*;

KEYWORDS

Real-Time, Embedded System, critical embedded system, model, meta-model, modeling, model transformation

ACM Reference format:
Anh-Toan Bui Long, Yassine Ouhammou, Emmanuel Grolleau, Loïc Fejoz, and Laurent Rioux. 2017. Bridging the gap between practical cases and temporal performance analysis: a models repository-based approach. In *Proceedings of RTNS conference, Grenoble, France, Oct 2017 (RTNS17),* 10 pages.

RTNS17, Oct 2017, Grenoble, France

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

https://doi.org/10.475/123_4

1 INTRODUCTION

Critical real-time systems need to be analyzed to check their temporal performance. That is, tasks, messages and end-to-end flows have to meet the requirements in terms of scheduling, deadlines, delays, processors utilization, etc.

Schedulability tests are among techniques that can be used to conclude about the temporal performance of a given system. They are basically based on algebraic methods and are known by their scalability referring to the number of tasks and messages of systems especially industrial ones. Since the 1970s, the real-time community has continuously proposed a plethora of schedulability tests [18] while each test is dedicated to a specific analysis model (a.k.a. task model). This latter relies on a set of assumptions describing the architecture, the tasks structure and the system worst-case behavior. In other words, the result of an analysis test can never be reliable unless the analyzed system fits accurately with the analysis model of the applied test.

1.1 Context and problem statement

The design of real-time systems becomes more and more complex since - in addition to the functional requirements - it spreads across various domains (e.g., time, energy, safety, security, etc.). The design process may then spread over several years and is essentially composed of two main phases: modeling and analysis.

The modeling phase leads to get a representation combining many artifacts originating from different domains while the analysis phase is separated into different domain-specific analyses. That is, the temporal validation is among those analyses that are applied on the qualitative and quantitative information derived and/or extracted from the modeling. In fact, the transition from the modeling phase to the analysis phase faces two main inter-dependant problems. The first problem is the difficulty of choosing the appropriate schedulability test according to the input model. The second one is that despite the existence of numerous tests related to different architectures, tests do not cover all possible situations of real-time systems. Hence, the modeling has to be transformed to conform to an existing test. Moreover, this transformation must be conservative (see Section 2.1), i.e., the worst case behavior of the target model should never be optimistic compared to the source model.

Currently, the transition process form the modeling to the analysis is still driven by designers experiences which makes the transition endeavour tedious, time-consuming and error-prone due to the complexity and precision required to manipulate real-time theory.

1.2 Work positioning

While our recent works [15, 21, 22] proposed to cope with the first problem by helping designers to choose the suitable analysis tests, the second problem still lacks a solution that can help designers to choose which conservative adaptations their models have to undergo in order to generate models compliant with the source ones in terms of temporal behaviour and to take benefits from analysis tests.

In this paper, we propose an approach to conceptualize the required knowledge and to capitalize efforts to ease the transformation and the adaptation of practical cases to match suitable analysis models. Our proposition is based on a an extensible models repository, which allows experts to share their knowledge and that can be used by non-experts as a decision support. Our proposition is based on model-driven engineering settings to make it utilizable with design languages.

1.3 Paper outline

The rest of the paper is organized as follows. Section 2 the state of the art and related works. Section 3 is dedicated to present the contribution. The implementation and tooling are presented Section 4. Section 5 presents a case study stressing the paper contribution. Finally, Section 6 summarizes and concludes this paper.

2 STATE OF THE ART

In this section, we present the theoretical and technological background of this work. We introduce some motivational examples and we discuss related work.

2.1 Background

Software engineers and architects design their real-time systems by using modeling languages [3, 11]. As such, their designs are described referring to functional and non-functional specifications to be easily understood. The description done by the design engineer do often not correspond to "conventionnal" analyses.

In the following, we present some of the cases that we considered as motivational examples for this work.

2.1.1 Example 1: alternative task triggering. A task can be activated by several ways: through a predetermined arrival pattern, or by other tasks. Figure 1 presents a case of OR precedence relationship. We consider three communicating tasks. Tasks τ_1 and τ_2 are sporadic tasks. Task τ_3 is activated by either task τ_1, or τ_2, through a message to τ_3. Therefore, τ_3 is preceded by τ_1, or τ_2. To the best of our knowledge, there is no off-the-shelf tool developed especially for this kind of architecture. That is, to analyze the system of Figure 1, one has to transform the architecture to be adapted to another analyzable structure. Moreover, the temporal behavior of the transformation result model has to dominate the origin model.

Since it is difficult to determine finely the execution of task τ_3, this latter can be represented as two tasks, τ_{3A} and τ_{3B}, as it is shown in Figure 2 . This leads in a task activated by τ_1 and the other one is activated by τ_2. Indeed, τ_3 is a task with two intertwinned sporadic activations. Since in the original model, τ_3 is usually considered as non-reentrant (i.e. FIFO activations), τ_{3A} and τ_{3B} should not be able to preempt each other in the second model.

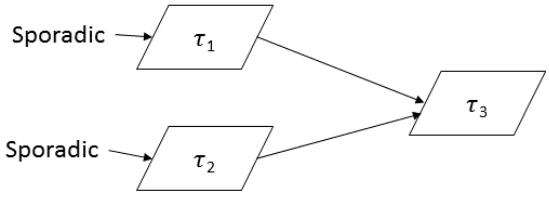

Figure 1: OR precedence relationship between 3 tasks

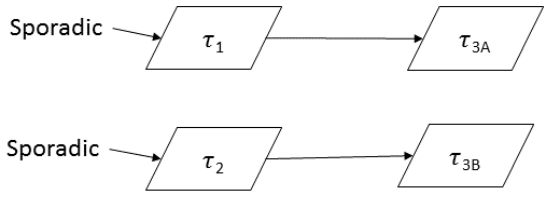

Figure 2: Transformation result of example shown in Fig. 1

Finally, note that this transformation is valid in a lot of contexts (uniprocessor, global multiprocessor, paritioned multiprocessor) as long as we can enforce that the two subtasks may never overlap, or are executed on the same processor if we have a partitioned scheduler. The priority should also remain consistant with the source model, for example, if the scheduler based on fixed-task priority, the two subtasks shall have the same priority to reflect reality.

2.1.2 Example 2: probabilistic task execution. Probabilistic schedulability analysis [16] allows to get an approached estimation of the system. Nevertheless, very few tools handle probabilistic worst-case execution time as input, but an upper bound on this wcet. Then, in case of probabilistic distributions of execution times (which for instance can be calculated via Weibull law), the designers need to consider the worst-case execution time to obtain a valid response time value. Consequently, a model transformation should provide the designers with an automatic mean to transform probabilistic tasks parameters into worst-case ones.

2.1.3 Example 3: sliding window arrival pattern. A sliding window is an arrival pattern type of task activation. A task τ_i activated in a sliding window is such that for any window time of size w_i, there are at most n_i activations of τ_i.

We came across this model in an industrial case study, and, there again, no available tool considers this pattern as a possible activation pattern.

We consider a task-set constituted by n tasks $\tau_{i,i=1\ldots n}$ with worst-case execution time C_i and a sliding window defined by n_i and w_i. We cannot determine the exact position of the activations to determine the evolution of the system and analyze the system's schedulability.

Nevertheless, if we are in a context where the request bound function (RBF) [2] (resp. the demand bound function [2] (DBF)) can be used to represent the worst-case activation pattern of tasks (resp. the worst-case cumulative processor demand to be satisfied), we can observe that it is possible to envelop every possible set of releases (resp. every possible set of deadlines) in a sliding window by a single sporadic release of n_i jobs with a period w_i. It shows that, in a context where the RBF or the DBF is used for analysis, a sliding window activation pattern can be equivalently represented by sporadic activations.

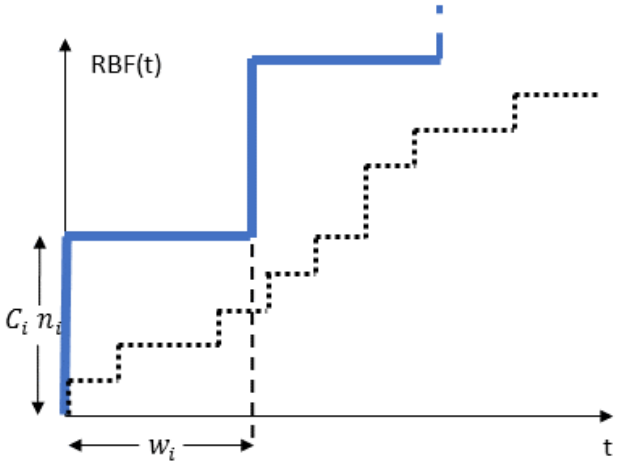

Figure 3: RBF of a sliding window

Figure 3 present two cases considering, for task τ_i, $n_i = 5$:

the blue curve represents the request bound function for a sporadic task τ_i with WCET $n_i C_i$ and period equal w_i,
the black (or dashed) curve represents different possible activation patterns for τ_i activated with a sliding window pattern.

This transformation from sliding window to sporadic preserves the interference as well as the worst-case response time for a sliding window task in the case of fixed-task priorities on a single processor. Feasibility is can also be studied using the RBF of a sporadic task rather than the sliding window task.

2.1.4 Example 4: watchdog tasks. The Watchdog is also a particular type of task activation. A task activated by a watchdog of timeout w is such that, if the watchdog is not reset during w, the task is activated. This kind of tasks usually plays the role of an alert to perform some action (e.g., switch to degraded mode, resolve errors, rebooting the system, sound an alarm, etc.). As watchdogs should not be triggered under normal operations, it is uncertain of when they can occur. For a safe and accurate temporal behavior analysis, the watchdog task can be assumed to be triggered every w time units at most, and, therefore, be modeled as a sporadic task with period (minimum inter-arrival time) equals to the timeout of the watchdog.

Several other examples could be found to highlight the gap between practical use cases and analysis models.

2.2 Model-driven Engineering for modeling and transformation

The model-driven engineering (MDE) [17] is a generative process enabling to get, treat and produce information from structured data entities called models. MDE settings are mainly based on modeling, meta-modeling and transformation aspects. Another important MDE aspect is the conformity relationship. A model conforms to the meta-model as a program conforms to the programming language's grammar in which it is written. This means that a meta-model describes the different types of model elements and how they are arranged. Thanks to the transformations techniques the transition of data from models to other models is possible. There are two kinds of model-to-model transformations, the exogenous and the endogenous transformation. While the first kind consists of having meta-models of the source models different to meta-models of the target models, the second kind means that source and target models are both conform to the same meta-model.

The need of MDE is increasingly growing to ease development and reduce its costs by following correct-by-construction approaches. Real-time systems actors have taken benefits form the MDE paradigm by proposing and using modeling languages such as the standards UML-MARTE (Modeling and Analysis of Real-Time Embedded systems)[13] and AADL (Architecture Analysis and Design Language) [5]. Several temporal performance analysis tools have also been proposed based on MDE technologies and have been integrated in the design process tool-chain (e.g., MAST [7], Cheddar [19], etc.).

2.3 Related work and discussion

Transformations are nowadays lead by either the analyst expertise or automatically by frameworks but opaque, non-extensible and without any information given to users.

There exist only few tools that explicit the transition from design models to analysis models: Tempo and MoSaRT propose such mechanism. Tempo [8] is a MARTE-compliant tool based on a subset of scheduling analysis concepts. Its transformations are driven by concepts semantics and are expressed by annotations. That is, the knowledge of the target schedulability test should be known by the user beforehand. Hence, the non-expert user cannot be totally autonomous. Moreover, the transformation is more guided by the initial input models of the analysis tools supported by Tempo. In other words, the goal of the transformation is to adapt models to analysis tools and not to adapt practical cases.

MoSaRT analysis repository [15, 22] proposes to capitalize the transformation from design language to analysis tools. It can be enriched by real-time researchers and practitioners by adding their assumptions that input analysis modelsmust meet and adding transformations towards associated analysis tools. However, the repository proposed by MoSaRT supports only conventional analyzable models and helps users to find the appropriate scheduling tests that match their models.

Since we focus on critical real-time systems, we realize that current transformations from the design models to the adapted design models still requires expertise from a scheduling analysis expert. The very important point is to verify that the output model (adapted model) is either equivalent to, or a worst-case of, the original model.

In this paper, we take benefits from MoSaRT analysis repository principles and we propose an extensible repository of endogenous model transformations called CONSERT (CONServative Endogenous Repository-based Transformations). It is in charge of finding and applying the necessary transformations to models based on experience of expert designers and analysts. CONSERT repository enables to store assumptions of models that need to be adapted and the appropriate transformations that should be run to obtain analyzable models that will be fed to schedulability analysis tools. This repository will also help designers to have a traceability of the changes applied to their original input model before analysis. Even if CONSERT does not depend on a particular design language and it can be compliant with several languages, in this paper we show the usefulness of our contribution by integrating it within a development framework called Time4Sys (which is a new evolution of MARTE). It stems from the Waruna project[1] where we are involved with other academic and practitioners partners.

3 CONSERT FUNDAMENTALS

The underlying idea behind proposing such repository is to store the knowledge of design/analyst experts and to share it with other collaborators and users. Hence, the repository can play the role of a decision support and make users autonomous during the analysis phase.

3.1 Transformation contexts

The context represents the required assumptions for a model to be considered as input model for the transformation. For example, the transformation from sliding window to sporadic can only be applied if all the jobs are to be executed on the same processor. We define a context C as a set of assumptions $\{a_1, a_2, .., a_n\}$ such as "the hardware architecture is uniprocessor", or "tasks priorities are fixed". Each assumption a_i is then a set of propositions, whose values are expressed as constraints on the meta-model, that an input model satisfies, or not. For example, characteristics of contexts tackled in Section 2.1 can be described by a set of assumptions as follows:

Excerpt of alternative task activation context (C_A):
- "The hardware architecture is uniprocessor"
- "There exists a least one task preceded by two or more tasks and the precedence kind is an *OR* relationship"
- "Every precedence constraint is either one-to-one or OR"
- etc.

Excerpt of sliding window context (C_B):
- "The hardware architecture is uniprocessor"
- "There is at least one task activated with a sliding window"
- "There is no self-suspension"
- etc.

Excerpt of tasks with probabilistic parameters context (C_C):
- "There is at least a probabilistic parameter"
- "Task(s) concerned by a probabilistic value:
  "is independent"
  "is not self-suspended"
- etc.

Excerpt of watchdog task activation context (C_D):
- "There is at least a watchdog activation"
- etc.

CONSERT can also be enriched with transformations requiring analysis, as long as the output model of the analysis can be expressed in the same meta-model as the source meta-model. For example, if tasks are periodic, and offset free [6], a third-party analysis tool could be used to assign a value to these offset such that the system is schedulable by a giving scheduling algorithm. A chain of subtasks with a final deadline could also be modified by a third-party tool proposing intermediate (tasks) deadlines. We could also see response time computation tools as third-party tool used for endogenous transformation, as long as the response time can be expressed in the meta-model. It is not yet implemented in our framework, but is a serious reflection for future work, in order to achieve an holistic analysis per part on an input system.

3.2 Detection rules

Each design which might be transformed/adapted would conform to a context stored in the repository. For this, we have to define detection rules to spot the assumptions in the design to apply the correct transformations.

Let $D_R = d_1, d_2, .., d_n$ be a set of detection rules. Since it is common to find several contexts characterized by the same subset of assumptions, each rule d_i may or not verify an assumption a_i and an assumption belongs to several contexts C_j. Indeed, each rule d_i is an interrogation checking the system design and leads to an assumption characterizing one or several contexts. For example, d_p is "number of processor equals to 1", thus, if d_p is true, it verifies assumption a_0 "the hardware architecture is uniprocessor", and if d_p is false, it verifies the assumption "the hardware architecture is not uniprocessor". Therefore, for factoring the number of assumptions and to guarantee the scalability and the incremental enrichment of CONSERT repositories, let S_f be a function for specifying contexts and defined as $S_f : C \quad D_R \; ! \; \mathit{Boolean}$. For instance, $S_f(C_A, d_p)$ = True.

3.3 CONSERT metamodel

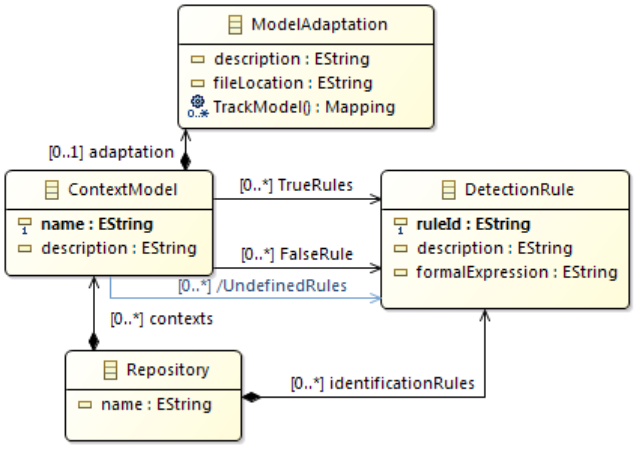

Figure 4: Excerpt of CONSERT metamodel

[1] www.waruna-projet.fr

Figure 4 shows the metamodel that embodied our proposition and which any CONSERT repository has to be conform to. Figure 4 is expressed in Ecore language [20] which is a graphical meta-modeling language with concepts quite similar to UML class diagram [14]. Indeed, the root element is the `Repository` class which each of their instances is composed of a set of detection rules (i.e. instances of `DetectionRule` class) and a set of contexts (i.e., instances of `ContextModel` class). Each detection rule is characterized by an `id`, a `description` written in native language and a `formal expression`. This latter relies on the modeling language expressing the design which requires transformations. For example, let $Sys = \{\tau_1, \tau_2, .., \tau_n\}$ be a system composed of a set of tasks, where every task τ_i is characterized by a period T_i and a relative deadline D_i, then τ_i:=<T_i, D_i>. Therefore, the formal expression attribute of a detection rule aiming to check if the deadlines of all tasks are constrained will be expressed as follows: $\forall \tau_i \in Sys, T_i \leq D_i$. In our case the formal expression will be written in OCL (Object Constraint Language [12]) since we would like to make our contribution compliant with Time4Sys, see Section 4).

Figure 4 also shows the interpretation of S_f function to two relations (`TrueRules` and `FalseRules`) between `ContextModel` and `DetectionRule` classes. Besides, every instance of `ContextModel` is characterized by an adaption (instance of `ModelAdaptation`) which consists of describing the necessary transformations and their implementations as a transformation program. The `fileLocation` defines the location where it exists the program to run to get transformed designs. Thanks to the `trackModel` operation, every execution of the transformation program on a design model generates a model that stores the transformations and then helps designers to follow the traceability and the evolution of initial models. The generated model conforms to the traceability meta-model that we present hereafter.

3.4 Traceability - Transformation tracking

The transformations conducted on a given model impact the contained information. As such, after a transformation, it is impossible to go back to the previous model unless we keep all the models in memory, i.e. giving a high amount of memory to unused models. We propose then to track the modifications made on the system model, by user or by the framework.

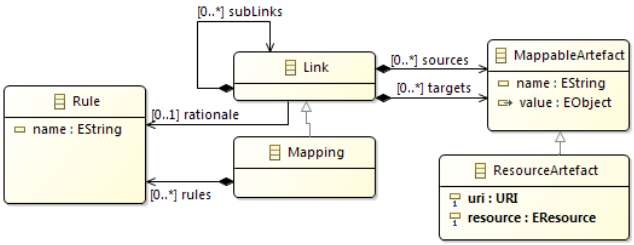

Figure 5: Traceability meta-model

Figure 5 shows the meta-model to which every generated traceability instance will be conform. The cornerstone of the traceability meta-model is the `Mapping` class that represents the link between source and target elements (both of them are instances of `MappingArtefact` class). Source and target elements can be distinguished thanks to the instances of the `ResourceArtefact` class since they are containers of these elements. In addition, every `ResourceArtefact` instance is characterized by a URI (Uniform Resource Identifier) and a resource (i.e., the root element of the model that contains the resource artefact). The mapping is characterized by a set of transformation rules expressed by a transformation language (like ATL [9] or QVT [4]). Moreover, the mapping can be a result of merging many source elements to get one target element or one source element leading to a bifurcation to many target elements.

4 TOOLING AND IMPLEMENTATION

This section presents the implementation of CONSERT and the software tools and the support of development used for our proposal.

4.1 Waruna Project

Waruna Project is a collaborative project reuniting industrial and academic partners from the field of real-time systems. It comes from the need of industrials to analyze real-time systems in their functional and non-functional requirements by using MDE paradigm. It would reduce the time of testing and reduce the cost of real-time related errors.

4.2 Integration in Time4Sys

4.2.1 Time4Sys in a nutshell. *Time4sys* is a framework developed as a Polarsys plugin[2] and available on GitHub [3]. The framework reuse the philosophy from *MoSaRT* and *Tempo* frameworks presented in Section 2.3. We present main functionalities.

Design. *Time4Sys* makes available a concrete syntax to model Real-Time Embedded Systems on a large panoply of classes extracted from UML - MARTE profile.

Traceability. There is a gap between design and analysis. Analysis models and design models are expressed on the same meta-model. Because there are some differences between the analysis and design model, there is a need of a tracking of modifications. *Time4Sys* proposes a traceability meta-model made with QVT [4] and ATL.

Analysis. *Time4Sys* provides connections with analysis tools such as *MAST*[7] or *RTaW-Pegase*[1]. Exogenous transformation from the design model to the analysis input is provided in the framework.

Result feedback. The plug-in intends to provide tools from modeling to analysis feedback (including model to model transformation to the analysis tool). To this extent, Time4Sys is capable to handle traces to track modifications made on the model.

The *Time4Sys* framework proposes an approach of Real-Time systems from the design to the analysis and the redesign if necessary. Figure 6 present the whole process available with Time4Sys. Model can be imported from a design language. The model and the framework are based on a subset of UML - MARTE profile. The model can also be established in the framework thanks to the

[2]https://www.polarsys.org/projects/polarsys.time4sys
[3]https://github.com/polarsys/time4sys

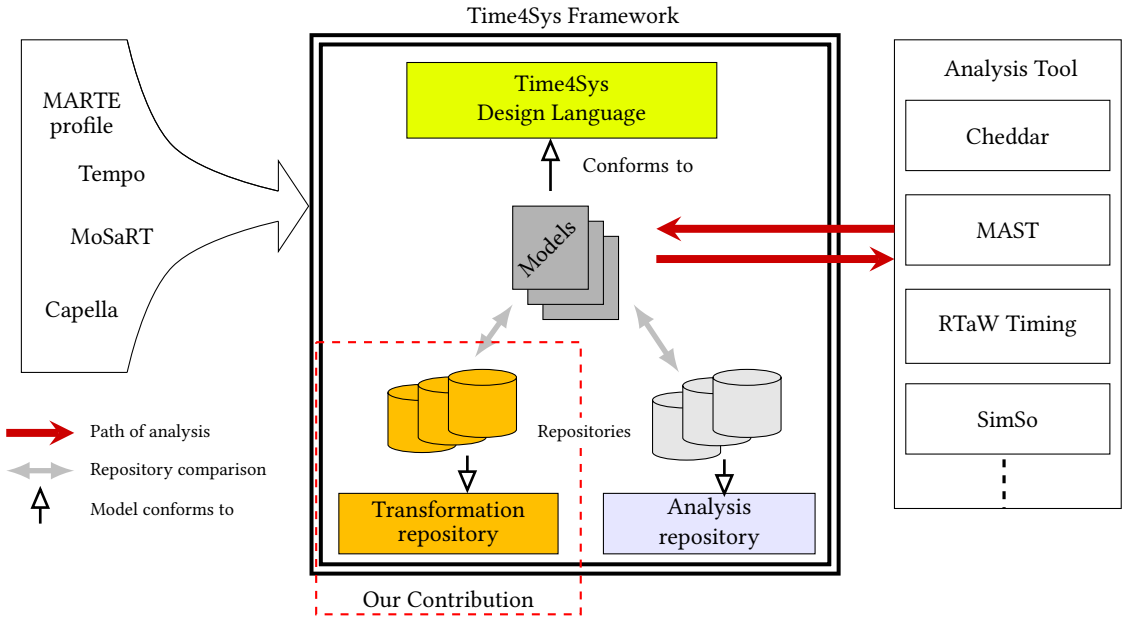

Figure 6: Abstract of the process of Time4Sys

concrete syntax embedded in the framework. Then, the model is compared in the transformation context repository to check for any available transformation to simplify the analysis. After the endogenous transformation, creating a model also conforms to the Time4Sys design language profile. The model is then compared to the analysis repository to verify for analysis that suits the model context or at least that approaches the model context.

4.2.2 Positioning of CONSERT inside Time4Sys. Figure 7 shows the detailed process of analysis in Time4Sys, integrating our workflow. Our repository intends to be the first step after modeling in Time4Sys framework (marked as (1) on the figure). From instances of rules provided by CONSERT, the framework is able to compare (marked (2) on the figure) with the model and determine the transformation(s) to retrieve the adapted model. The transformations (marked (3) on the figure) are conducted by the framework. Afterwards, the model is compared with the analysis repository to determine the analysis tool (marked (4) and (5) on the figure). A transformation from the adapted model to the analysis tool input leads the model to its analysis (marked (6) on the figure). The output is retrieved to enrich the model for eventual further analyses.

4.3 General structure and scenario usage

The meta-model is established on EMF (Eclipse Modeling Framework [20]). To obtain the context and compare to the repository, OCL (Object Constraint Language [12]) is used to implement transformation rules between models. The endogenous transformation is done through the ATL (Atlas Transformation Language [9])

If Time4Sys is used as support of our proposal, Figures 8 and 9 presents examples of instances of CONSERT and trace repositories

4.3.1 Pessimism on transformations. To transform real-time systems, we need to be conservative. In software certification, this means the transformation cannot make the new model more optimistic, scheduling-wise, than the model considered in the previous step.

4.3.2 Classes transformation with Time4Sys. The transformations impact few classes on the meta-model. It is possible then to verify the model obtained at each step. Table 1 summarizes all transformations applied on Time4Sys classes for the cases we presented at Section 2.1.

In the case of precedence constraints, a step can be preceded by one or more step. It can be triggered by any step. All properties of the triggered step, such as priority, blocking time, are duplicated into a new task. The duplicated steps and tasks have the same properties, except these have to access a mutual exclusion resource to avoid these tasks preempting each other.

For probabilistic values, we convert the class associated to the probabilistic value to a simple duration and take the minimum or the maximum depending on the probabilistic parameter it was affected to.

For Sliding windows activation, classes are transformed into a burst pattern retrieving the parameters contained in the previous class.

A watchdog is currently designed as an Alarm with a reset service avoiding the execution of the alarm. The service is converted into an independent sporadic task.

5 CASE STUDY - APPLICATION ON A MINEPUMP

We present a case study of a simple example. First, we present the initial design of the case study expressed in Time4Sys. Then, we emphasize the practical case that should be adapted. We show how the usage of a CONSERT repository helps to perform automatically

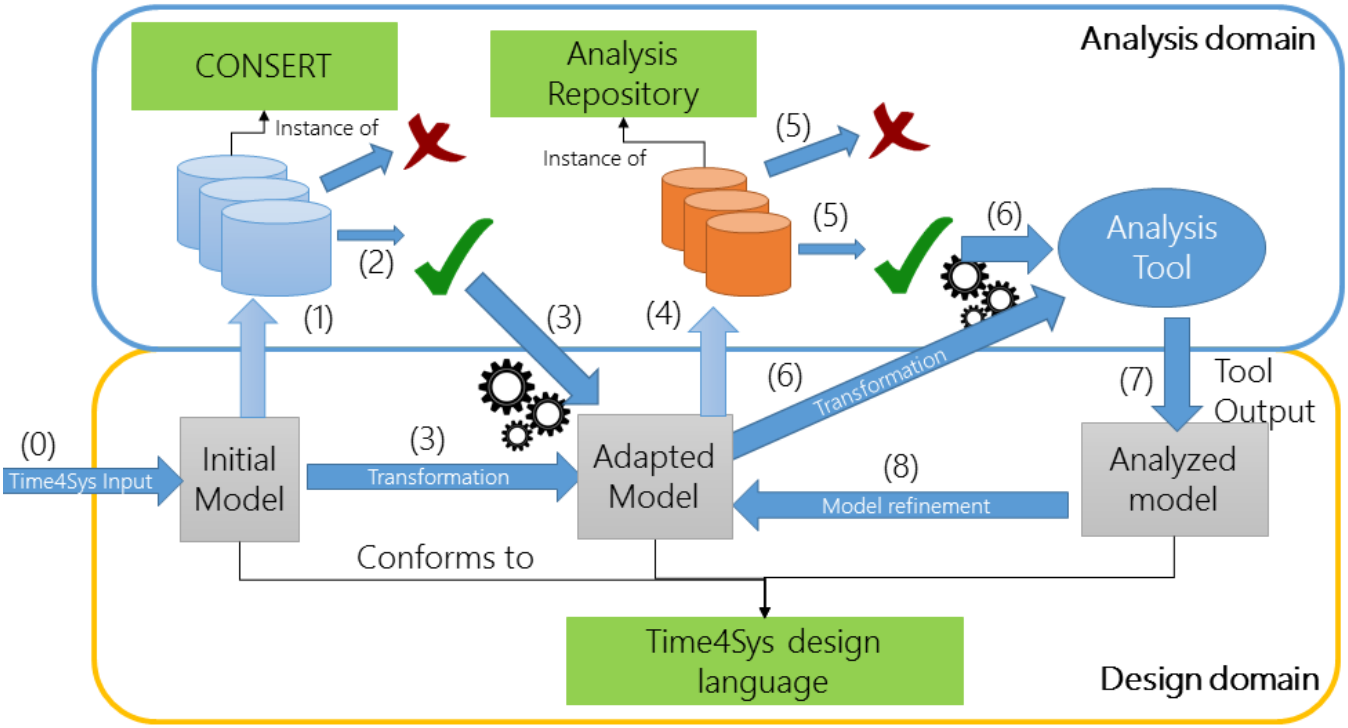

Figure 7: Workflow of Time4Sys and integration of CONSERT

Time4Sys Class	Before Transformation	After Transformation
Alternative precedence relations context C_A		
`PrecedenceRelation`	Connector kind set to Merge	Connector kind set to Sequence
`Step`	Single step with two (or more) precedences	As much steps as there were precedences
`SoftwareSchedulableResource`	One task	As much tasks as there were precedences
`MutualExclusion`	None	A mutual exclusion for each instance of the task(s)
Sliding Windows activation pattern context C_B		
`SlidingWindowPattern` (inherited from `ActivationPattern`)	n_i maximum occurences in any window of size w_i	Non-existent
`BurstPattern`	Non-existent	Burst of n_i occurences every w_i
Probabilistic values context C_C		
`ProbabilisticDuration` (inherited from `NFP_Duration` & `NFP_TimeInterval`)	Probabilistic distribution of time	Non-existent
`Duration` (for WCET values)	Probabilistic	Equals to maximum duration value
`Duration` (for inter-arrival time)	Probabilistic	Equals to minimum duration value
Watchdog context C_D		
`Alarm`	Designed as a watchdog	Non-existent
`SoftwareSchedulableResource`	Taskset	Creation of task Watchdog
`ActivationPattern`	N/A	Creation of a sporadic activation associated to the task Watchdog

Table 1: Table of transformations

a series of transformation in order to get an analyzable design while remaining the pessimist temporal behavior.

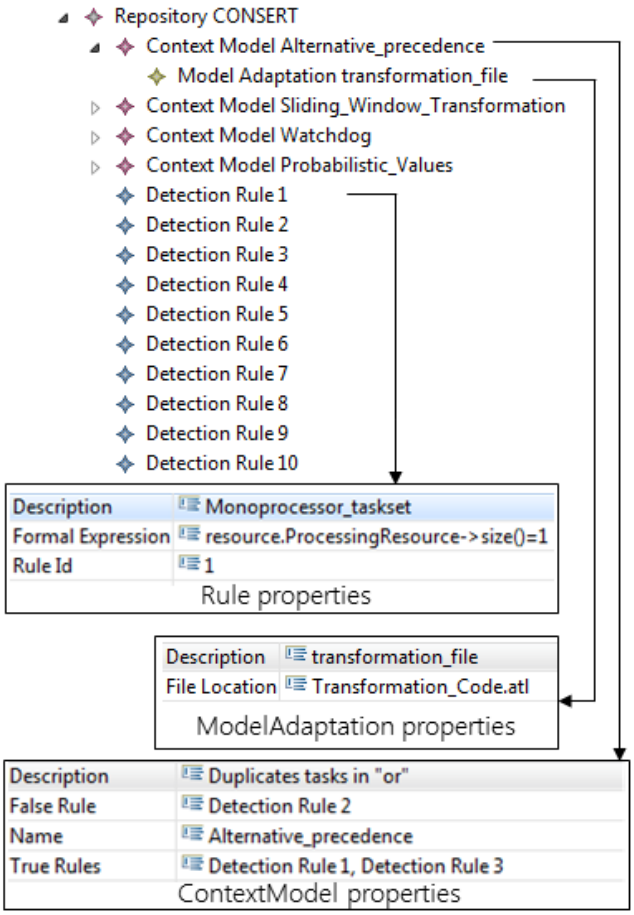

Figure 8: Instance of the transformation repository

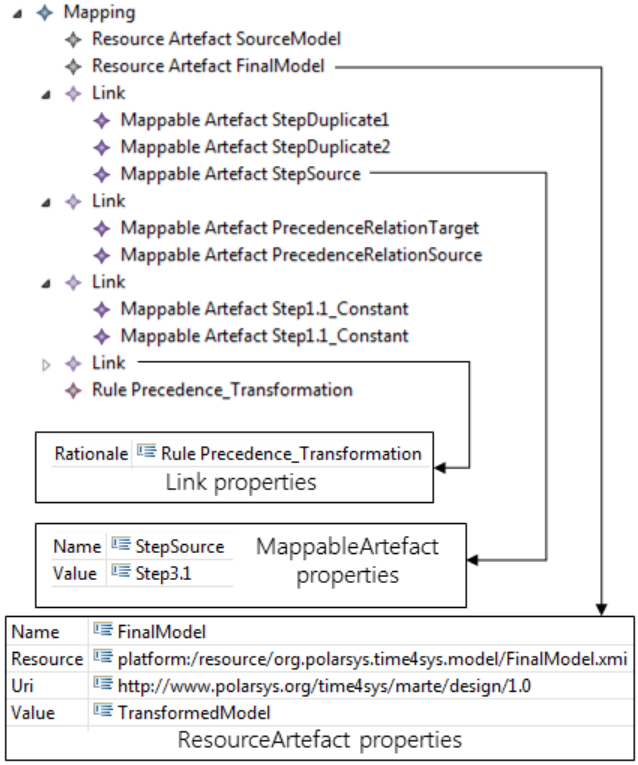

Figure 9: Example of an trace instance

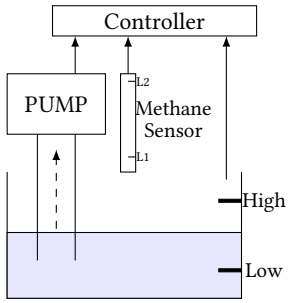

Figure 10: MinePump use case

5.1 A sample practical use case

We consider a deep well filled with water inspired from [10]. It is required to keep the water between Low and High levels. A methane sensor is located in the well, to shut down the pump when the air is explosive (in case of a sparkle), but also to evacuate the employees when the level is dangerous for a human being. If the methane level is over $L1$, an alarm should be triggered. If the methane level is over $L2$, the pump shall be deactivated. At Low level, the pump should be off as it cannot pump any more water. All software components are executed on a uniprocessor controller. Figure 10 presents a schematic of the system.

5.2 Initial modeling

The software architecture is composed of three tasks. Task 1 is periodic, and acquires the methane level every 50ms, task 2 retrieves periodically information from Low level and High level sensors, and task 3 has to compute whether or not the pump should work and check if alarm should be triggered or stopped.

Task 1 presents three steps corresponding to functions of the task defined at functional modeling stage: acquire signal, compare methane, operate alarm. Task 2 has only 1 step of water level acquisition. These tasks sends their data to task 3 in charge of computing the signal to send to the pump.

Task 1 has a periodicity of 50ms and Task 2 a periodicity of 500ms.

There is a watchdog in the system to be reset every 50ms. The watchdog is reset at each successful frame read. A trigger of the watchdog implies a late arrival of a frame or a bad transmission of the message.

We first try to analyze this system. For this kind of system, MAST software is proposed (version 1.5.1.0). This implies a single transaction for the system in MAST.

Task	T_i	C_i	Priority
T_1	50*ms*	10*ms*	15
! Step1.1	—	2*ms*	—
! Step1.2	—	3*ms*	—
! Step1.3	—	5*ms*	—
T_2	500*ms*	10*ms*	13
! SingleStep	500*ms*	10*ms*	—
T_3	—	10*ms*	5
! Step3.1	—	10*ms*	—
! Step3.2	—	5*ms*	—
Watchdog	—	5*ms*	1

Table 2: Summary of the system

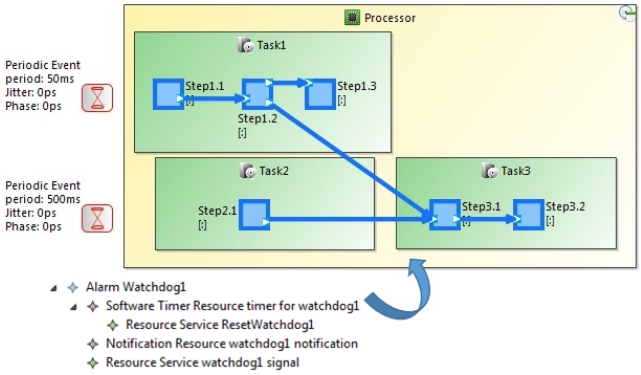

Figure 11: Transformation separating the transactions

The model cannot be analyzed as a tool failure exception stops the execution of the tool. A transformation is necessary.

5.3 First step

In this modeling, there is an "OR" precedence: task 3 preceded by either Task 1 or 2. Figure 11 presents the Time4Sys model of the case study. In Time4Sys, a watchdog cannot be represented at this moment in the graphical syntax. The steps are given periods and computation times C_i as follows in Table 2.

5.4 Second step of the transformation

The transformation conduct to the creation of two transactions, instead of one transaction as previously, beginning with the activation of tasks 1 and 2. The tasks must be in mutual exclusion, to avoid two jobs of task 3 preempting each other during the process.

5.5 Third Step

The transformation creates a watchdog task as a sporadic task of minimum interarrival time equals to 50ms. Figure 13 presents the model transformed after transformation.

As the watchdog activates a service on task 3, alerting it of a frame failure, the transformation implies the creation of a third instance of Task 3 in mutual exclusion with the other instances of "task 3" As it is a safety task on system, it is affected the highest priority on the processor. Table 3 presents the results on the system with a watchdog.

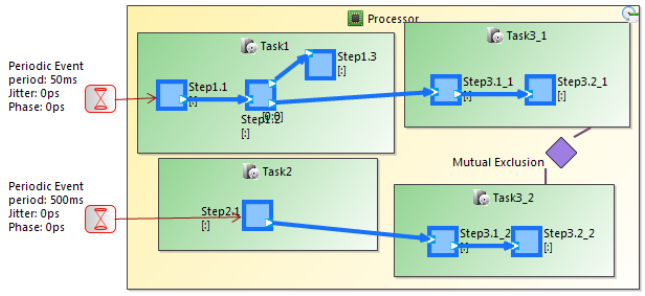

Figure 12: Transformed model separating the transactions

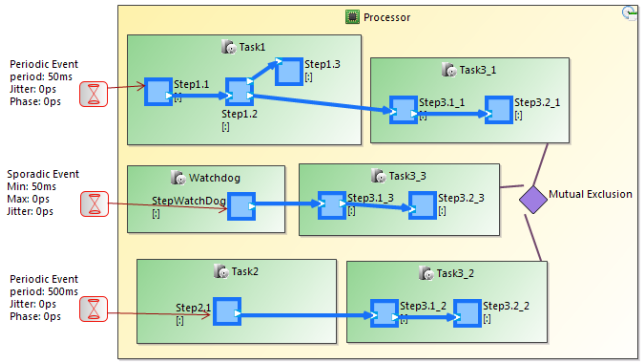

Figure 13: Transformation adding a watchdog task

5.6 Analysis

We conduct the analysis on MAST. Table 2 summarizes the results obtained through the analysis software. Column **Case 1** designates worst case response time excluding the watchdog in the computation, and column **Case 1** including the watchdog as a sporadic task.

Task	WCRT	
	Case 1	Case 2
Task1	285*ms*	545*ms*
Task2	10*ms*	15*ms*
Task3 (activation by T_1)	190*ms*	345*ms*
Task3 (activation by T_2)	110*ms*	210*ms*
Watchdog	—	5*ms*

Table 3: MAST Results - Sliding Window

6 CONCLUSIONS

This papers tackled the difficulty faced by designers to analyze some practical use cases. The designs of these use cases require to be adapted to become analyzable (in a temporal performance point of view) hence fit a model analysis to conclude about their schedulability.

Basically, the required adaptation and transformation are guided by analysts expertise since it is important that the resulting design

should never be optimistic compared to the source one. To capitalize this expertise and make designers autonomous during the analysis phase, we proposed CONSERT (CONServative Endogenous Repository-based Transformations). Thanks to CONSERT, any expert can share his/her knowledge with other collaborators by creating a CONSERT repository where it is possible to store the necessary transformation for each kind of design. Consequently, the transformation can be performed automatically by connecting the repository to a modeling language. Moreover, the utilization of CONSERT repositories enables users to get, in addition to transformed models, the evolution traces (also as model) of their initial model that underwent the transformations.

CONSERT has been integrated within Time4Sys framework and supports OCL to enable users express detection rules. The transformation and the traceability is managed thanks to ATL. A sample use case has been treated in this paper to show how such work can be needful in terms of safety and time-shortening since the content of the repository is supposed to be done by an expert and the implicit and manual transformations become explicit and automatic.

ACKNOWLEDGMENTS

The authors would like to thank all the partners of the project.

This work is funded through the Waruna project by the French Ministry of the Economy, Finances and Industry. This project is co-funded by the following Regions: Ile-de-France, Occitanie, Grand-Est, and Grand-Nancy. This project is co-funded by the European Union. Europe is committed in France through the European Funds for Regional Development and the European Social Fund.

REFERENCES

[1] RealTime at Work. 2017. RTaW-Pegase. http://www.realtimeatwork.com/software/rtaw-pegase/. (2017). Accessed: 2017-07-17.

[2] S. K. Baruah, A. K. Mok, and L. E. Rosier. 1990. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *[1990] Proceedings 11th Real-Time Systems Symposium*. 182–190. https://doi.org/10.1109/REAL.1990.128746

[3] Cinzia Bernardeschi, Marco Di Natale, Gianluca Dini, and Dario Varano. 2017. Modeling and generation of secure component communications in AUTOSAR. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*. 1473–1480. https://doi.org/10.1145/3019612.3019682

[4] Waruna consortium. 2017. Query/view/transformation QVT. http://www.omg.org/spec/QVT/. (2017). Accessed: 2017-07-17.

[5] Peter H. Feiler and David P. Gluch. 2012. *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language* (1st ed.). Addison-Wesley Professional.

[6] Joël Goossens. 2003. Scheduling of offset free systems. *Real-Time Systems* 24, 2 (2003), 239–258.

[7] M. González Harbour, J. J. Gutiérrez García, J. C. Palencia Gutiérrez, and J. M. Drake Moyano. 2001. MAST: Modeling and Analysis Suite for Real Time Applications. In *Proceedings of the 13th Euromicro Conference on Real-Time Systems (ECRTS '01)*. IEEE Computer Society, Washington, DC, USA, 125–. http://dl.acm.org/citation.cfm?id=871910.871923

[8] Rafik Henia, Laurent Rioux, and Nicolas Sordon. 2013. TEMPO: Performance Viewpoint for Component-based Design of Real-time Systems. *SIGBED Rev.* 10, 2 (July 2013), 12–12. https://doi.org/10.1145/2518148.2518150

[9] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. 2008. ATL: A model transformation tool. *Science of Computer Programming* 72, 1 (2008), 31 – 39. https://doi.org/10.1016/j.scico.2007.08.002 Special Issue on Second issue of experimental software and toolkits (EST).

[10] Brendan P Mahony and Ian J Hayes. 1992. A case-study in timed refinement: A mine pump. *IEEE transactions on Software Engineering* 18, 9 (1992), 817–826.

[11] Chokri Mraidha, Sara Tucci-Piergiovanni, and Sebastien Gerard. 2011. Optimum: A MARTE-based Methodology for Schedulability Analysis at Early Design Stages. *SIGSOFT Softw. Eng. Notes* 36, 1 (Jan. 2011), 1–8. https://doi.org/10.1145/1921532.1921555

[12] OMG. 2006. Object Constraint Language, OMG Available Specification, Version 2.0. (2006).

[13] OMG. 2015. UML profile for MARTE: modeling and analysis of real-time embedded systems. (2015).

[14] OMG. 2015. Unified Modeling Language (UML) version 2.5. *OMG Specification* (2015).

[15] Yassine Ouhammou. 2013. *Model-based Framework for Using Advanced Scheduling Theory in Real-Time Systems Design*. Ph.D. Dissertation. ENSMA.

[16] Luca Santinelli and Liliana Cucu-Grosjean. 2015. A Probabilistic Calculus for Probabilistic Real-Time Systems. *ACM Trans. Embed. Comput. Syst.* 14, 3, Article 52 (April 2015), 30 pages. https://doi.org/10.1145/2717113

[17] Douglas C Schmidt. 2006. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-* 39, 2 (2006), 25.

[18] Lui Sha, Tarek Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K Mok. 2004. Real time scheduling theory: A historical perspective. *Real-time systems* 28, 2-3 (2004), 101–155.

[19] F. Singhoff, J. Legrand, L. Nana, and L. Marcé. 2004. Cheddar: A Flexible Real Time Scheduling Framework. In *Proceedings of the 2004 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-time &Amp; Distributed Systems Using Ada and Related Technologies (SIGAda '04)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/1032297.1032298

[20] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. 2008. *EMF: Eclipse Modeling Framework*. Pearson Education.

[21] Ouhammou Yassine, Grolleau Emmanuel, Richard Michael, and Richard Pascal. 2012. Reducing the gap between Design and Scheduling. In *20th International Conference on Real-Time and Network Systems*, ACM (Ed.). ACM, Nancy, France, 10.

[22] Ouhammou Yassine, Grolleau Emmanuel, Richard Michael, and Richard Pascal. 2014. Towards a model-based approach guiding the scheduling analysis of real-time systems design. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2014)*. 19–24.