An Experience Integrating Response-Time Analysis and Optimization with an MDE Strategy

Juan M. Rivas[1(✉)], J. Javier Gutiérrez[1], Mario Aldea[1], César Cuevas[1], Michael González Harbour[1], José María Drake[1], Julio L. Medina[1], Laurent Rioux[2], Rafik Henia[2], and Nicolas Sordon[2]

[1] Software Engineering and Real-Time Group, University of Cantabria, Santander, Spain
{rivasjm, gutierjj, aldeam, cuevasce, mgh, drakej, medinajl}@unican.es

[2] Thales Research and Technology, Palaiseau, France
{laurent.rioux, rafik.henia, nicolas.sordon}@thalesgroup.com

Abstract. The objective of this experience is applying Model-Driven Engineering (MDE) to the development of complex design toolchains for distributed real-time systems by integrating stand-alone tools for this kind of system. MDE provides the capability to present to each tool the view of the design that is required in each case and also provides the traceability between models to return the results of applying a tool to the original model where the whole information of the developed system persists. Since the tools require complex and interrelated scenarios of model transformation processes they need to be programmed and optimized to obtain acceptable execution times and scalability. The experience described in this paper is the development of a Model-Driven Engineering (MDE) toolchain to support the design of distributed real-time systems using stand-alone tools for calculating response times, assigning priorities to tasks and allocating tasks to processors. The process starts from a base design described with a model that follows the OMG MARTE specification. This toolchain can be applied at any stage of the design process using timing parameters with different degrees of refinement, thus allowing the exploration of different design solutions when needed.

Keywords: MDE tools · IDE · Schedulability analysis · Optimization · Real-time · Design space exploration

1 Introduction

Development toolchains for embedded real-time systems require exploring multiple optimization aspects at the design phase, before their implementation. Aspects like the architectural design, the concurrency model, the deployment on the distributed platforms, the timing behavior, security and others require the description of specific system information and are heavily interrelated. Each of these aspects is supported by a

P. Milazzo et al. (Eds.): STAF 2016, LNCS 9946, pp. 303–316, 2016.
DOI: 10.1007/978-3-319-50230-4_23

community of experts that generates knowledge and provides tools to manage it. A challenge of current software engineering is to facilitate the integration of this knowledge and the resources produced by the experts in different domains in such a way that the engineers developing the system can use them in an efficient and easy way.

Model-Driven Engineering (MDE) [1] is the methodology that currently provides the most promising approach to manage the complexity and multi-aspect nature of embedded real-time systems development:

- In MDE all the information on the system is formulated as models and the development and design processes are specified in terms of model transformations. A system may be represented by a large number of models, some of which have the objective of storing a persistent description of the system, but others are temporarily generated to extract and adapt the information required by the specific tools used during the development to handle each aspect of the design.
- In MDE each model is defined as an instance of a meta-model that describes the contents, structure and format of the terminal model containing the system information. This formalization allows generating transformation tools that are generic and reusable over many kinds of models by using the meta-models as reflective information.
- The use of references between the elements of different models avoids duplicating information and facilitates the global coherence and maintenance of the system information.

Conventional MDE toolchains are sequences of model transformations. However, a toolchain for embedded real-time systems must contain different branches working on the aspects of the design supported by stand-alone tools. In this case we find the following requirements:

- The transformation branches are usually of double direction. The direct branch adapts the information stored in the base design models to the representation required by the stand-alone tool, while the reverse branch returns the results generated by the tool to the system base description. Since these branches are interdependent they need to be co-designed and co-maintained, and they need to exchange transversal configuration information among them.
- Sometimes the transformation processes are iterative rather than linear, since the transformation branch must be repeated until a particular objective is achieved.
- Efficiency is a concern, especially in these iterative processes, and this requires finding imaginative ways to store the models and avoid repeating transformations as much as possible.

The experience described in this paper is the implementation of a toolchain for designing embedded real-time distributed systems starting from a base description of the system that follows the OMG MARTE standard profile (The UML Profile for Modelling and Analysis of Real-Time and Embedded Systems) [2]. The Schedulability Analysis Modeling chapter (SAM) of this standard supports all the necessary modelling elements to perform the schedulability analysis or optimization of a real-time system. The process followed by the toolchain includes the schedulability analysis of the system, which allows ensuring whether the timing requirements imposed in the software can be met or not. When a system meets all its timing requirements, we say that the system is schedulable. In common real-time distributed systems, like those that can be found on cars or airplanes, the schedulability analysis normally consists on the calculation of worst-case response times in order to compare them with the timing requirements imposed on specific actions of the software.

On the other hand, the toolchain pays attention to two particular optimization aspects: (1) finding the scheduling parameters (usually priorities) that allow the system to be schedulable, and (2) in a distributed system finding a suitable allocation of tasks and messages to processors and networks, respectively. This is what we call the architecture optimization. During the optimization process these two aspects can be combined in order to find the best possible solution. The schedulability analysis and optimization of a real-time distributed system are supported by complex techniques that are implemented in the appropriate stand-alone tools.

In this paper, we propose the integration of schedulability analysis and optimization tools within an MDE strategy with the following characteristics:

- A general meta-model based on the MARTE standard is used at the base design level. This meta-model allows modelling systems independently of the application domain.
- An intermediate model particularly suitable for analysis will allow the connection (through the appropriate transformations) between the general model coming from the design phase and the specific model used by the selected analysis tool. Therefore, any existing analysis tool can be integrated in the toolchain through the adequate model transformation.
- A special-purpose tool is created to optimize (1) the priority assignment for the tasks and messages (the latter only if fixed-priority communication networks are specified), and (2) the allocation of tasks and messages to processors and networks.

As a proof of concepts, in a previous work [3] we presented a prototype of the toolchain described in this paper. In the current work, the integration of the stand-alone tools has been completed and implementation details are presented.

This document is organized as follows. Section 2 reviews some tools that are related with the approach we present in this paper. In Sect. 3 we review the real-time system model and current schedulability analysis and optimization techniques. In Sect. 4, we provide an overview of the architecture proposed for the integration of the tools in the MDE toolchain. Section 5 describes the integration of an available schedulability analysis tool in our MDE strategy. In Sect. 6, we describe the design of the optimization tool for priorities and architecture. Section 7 provides an industrial case study to which the tools have been applied. Finally, Sect. 8 draws the conclusions.

2 Related Work

There are other recent works in line with this approach. For example, [4] presents Optimum, a MARTE-based methodology for designing real-time applications in a schedulability-aware fashion, i.e., enabling schedulability analysis of UML models at early stages. In [5], the MoSaRT analysis repository is presented as a helpful modelling support to avoid wrong design choices at an early design phase, thus helping designers to cope with the scheduling analysis difficulties. The work in [6] proposes an integrated approach for prediction of performance in the context of distributed real-time embedded defense systems. In this case, performance prediction aims at addressing issues related to the integration of realistic data sources or the visualization of the causes of performance issues. In [7], model-driven development is applied to the integration of schedulability analysis in the development process of high-integrity distributed real-time systems programmed in the Ada programming language.

The main difference with previous works is that our MDE approach is more complete and modular in the sense that the tools for analysis, priority optimization, or architecture optimization, are independent and could be easily changed if, for example, a better tool is available. On the other hand, our approach can be applied to any application domain that would need verifying real-time properties or optimizing the whole or a part of the application, once it has been designed following the general base model.

3 Real-Time Model

This section gives a high-level overview of the real-time model behind the presented MDE strategy, along with the different versions of this model used by the schedulability analysis and optimization tools integrated in the toolchain. We follow the OMG MARTE standard [2] in which the system's software is described as a set of distributed end-to-end flows executing in multiple processors that can be communicated through one or more networks. This model is commonly used by schedulability analysis and optimization algorithms for distributed real-time systems.

The model follows an event-driven approach, in which each end-to-end flow is re-leased by an event that can be periodic or sporadic. Sporadic events must have a minimum inter-arrival time. Each end-to-end flow is composed of a series of steps that execute sequentially. A step represents the execution of a thread in a processor, or the transmission of a message through a network. Each step has a worst-case execution time (WCET), which specifies the maximum amount of time (or an upper bound on it) that the step needs to execute if it were alone in the system. Likewise, a step can also have a best-case execution time or lower bound on it (BCET). Each step is scheduled by using the scheduling parameters assigned according to the scheduling policy used. In this work we consider a fixed-priority policy.

Figure 1 shows an example of one end-to-end flow with 6 steps executing in two processors and one network. After the first step, there is a forking action that simultaneously releases two different branches of the end-to-end flow. The timing requirements are given as end-to-end deadlines which reference the event triggering the end-to-end flow, and must be met by the last step in the corresponding end-to-end flow branch. As a result of applying a schedulability analysis technique, a worst-case response time (WCRT) is calculated for each step. If the WCRT of the last step of each end-to-end flow branch is less than or equal to its deadline, the system is said to be schedulable, that is, it is guaranteed to meet its deadlines in all cases, including the worst one.

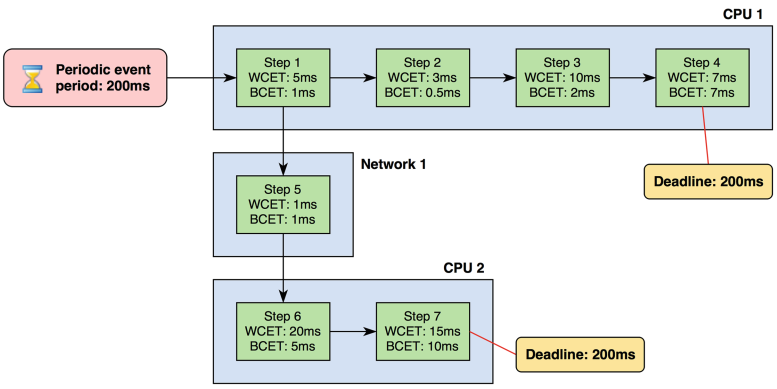
CPU 1
Periodic event period: 200ms
Step 1 WCET: 5ms BCET: 1ms
Step 2 WCET: 3ms BCET: 0.5ms
Step 3 WCET: 10ms BCET: 2ms
Step 4 WCET: 7ms BCET: 7ms
Deadline: 200ms
Network 1
Step 5 WCET: 1ms BCET: 1ms
CPU 2
Step 6 WCET: 20ms BCET: 5ms
Step 7 WCET: 15ms BCET: 10ms
Deadline: 200ms

Fig. 1. Example of an end-to-end flow

At the base design level we will use the TEMPO-MARTE meta-model presented in a previous work [8] which enables the description of the real-time system design. This meta-model is suitable for different application domains, and it has been successfully applied to the industrial design of on-board satellite software. This design model can be transformed into another one, compliant to another meta-model called TEMPO-Analysis, which contains the relevant information (concurrency and real-time aspects) organized in a format more suitable for the schedulability analysis and optimization tools. Finally, this analysis model should be transformed into the specific model of the selected analysis tool.

In this paper, we present the integration of MAST (Modeling and Analysis Suite for Real-Time Applications) in the proposed toolchain for the schedulability analysis of the system. MAST defines a meta-model [9, 10] to describe the timing behavior of real-time systems which is aligned with the MARTE standard, and it also provides a selection of techniques [11] to perform schedulability analysis, priority optimization, or sensitivity analysis (assessment of how far or close is the system from meeting its timing requirements). Regarding schedulability analysis, this selection includes representative algorithms such as: the classic holistic analysis [12] that considers the steps as if they were independent; (2) an offset-based technique [13] that exploits the interdependencies among the steps of the same end-to-end flow through the use of task offsets; and (3) another offset-based technique [14] that exploits the precedence relations among the steps.

4 Toolchain Architecture Overview

We propose an MDE toolchain that implements the necessary underlying infrastructure to perform schedulability analysis and optimization of real-time systems inside an EMF/Ecore environment based on the TEMPO meta-model aforementioned. An overview of the architecture of the toolchain is shown in Fig. 2. The toolchain is composed of two different interoperable tools:

1. A schedulability analysis tool that determines if the modelled system is schedulable by calculating the worst-case response times of the steps.
2. Two optimization tools that modify certain characteristics of the TEMPO-Analysis model to achieve schedulability: (1) a priority optimization tool that assigns optimized priorities to steps, and (2) an architecture optimization tool that allocates steps to processors and networks.

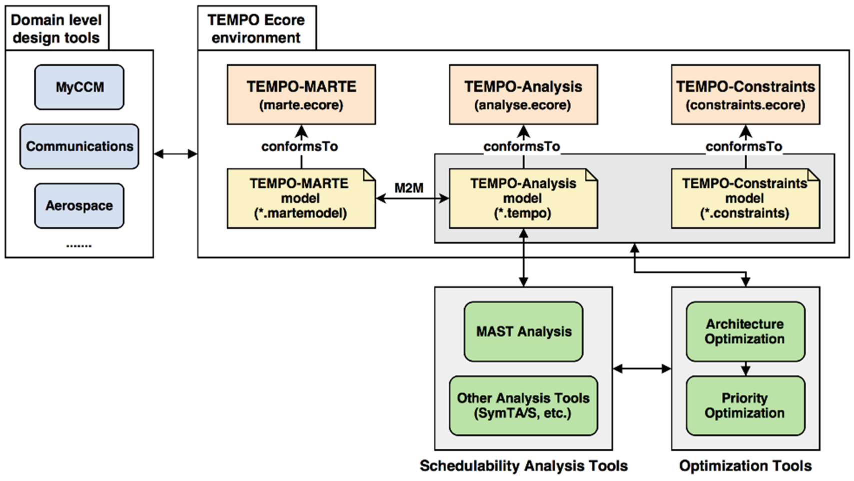
Domain level design tools
MyCCM
Communications
Aerospace
....
TEMPO Ecore environment
TEMPO-MARTE (marte.ecore)
TEMPO-Analysis (analyse.ecore)
TEMPO-Constraints (constraints.ecore)
conformsTo
conformsTo
conformsTo
TEMPO-MARTE model (*.martemodel)
M2M
TEMPO-Analysis model (*.tempo)
TEMPO-Constraints model (*.constraints)
MAST Analysis
Other Analysis Tools (SymTA/S, etc.)
Architecture Optimization
Priority Optimization
Schedulability Analysis Tools
Optimization Tools

Fig. 2. Architecture of the TEMPO analysis and optimization toolchain

While TEMPO-MARTE represents the basis for the design view, its SAM-like derivative (TEMPO-Analysis) specifically targets schedulability analysis and optimization tools. Hence, the toolchain we propose operates on the TEMPO-Analysis models directly, thus avoiding unnecessary TEMPO-MARTE to TEMPO-Analysis (and vice versa) transformations.

The architecture optimization tool relies on the priority optimization tool to calculate a priority assignment for each allocation of steps to processors tested. Similarly, the priority optimization tool uses a schedulability analysis tool that accepts a TEMPO-Analysis model as input. In this experience we use MAST as the schedulability analysis tool inside this Ecore environment, but the toolchain is designed to work with any other TEMPO-Analysis compatible tool. In this way, the optimization tools can be used in conjunction with other analysis tools such as SymTA/S [15].

In order to allow the specification of other parameters not included in the TEMPO-Analysis meta-model but necessary for the optimization process, the input model for the optimization tool is complemented with an additional constraints model that is compliant to the TEMPO-Constraints meta-model. This model contains information such as valid priority ranges, which step priorities cannot be modified, or step-to-processor affinities.

The toolchain implements the necessary model-to-model (M2M) transformations to operate with the different meta-models involved. All these transformations work under the Eclipse Modeling Framework (EMF). Some of them are implemented in Java, while others rely on ATL [16]. The following sections explain in more detail the design of these tools and their interconnections. Details on the TEMPO-MARTE to TEMPO-Analysis transformations and back can be found in [8].

5 Integration of the Schedulability Analysis Tool

The goal of the schedulability analysis tool is to determine if the system is schedulable. This is carried out by calculating the WCRTs of the steps and comparing them with the imposed deadlines. In this work we define a tool to perform the schedulability analysis on models compliant to the TEMPO-Analysis meta-model using the MAST analysis tools. The link between the TEMPO-Analysis input model and MAST is automatically established using ATL M2M transformations. Two transformation chains are defined in this link:

1. A TEMPO-Analysis to MAST transformation to create the input model for the MAST analysis tool.
2. A return transformation, in which the worst-case response times obtained by MAST are incorporated into the TEMPO-Analysis model.

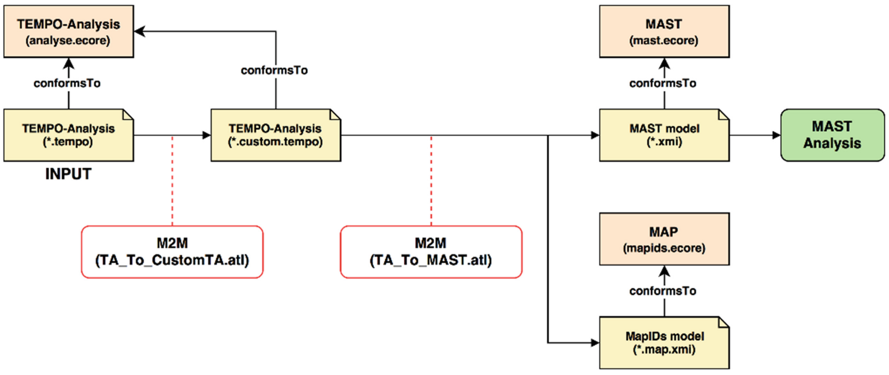

Fig. 3. TEMPO-Analysis to MAST transformation

The TEMPO-Analysis to MAST transformation chain is depicted in Fig. 3, and is composed of two ATL transformations applied sequentially. In the first ATL transformation (TA_To_CustomTA.atl), a new TEMPO-Analysis customized model is generated from the TEMPO-Analysis input model. This new model adds the information that is optional in TEMPO but is mandatory in a MAST model, such as names for every element. Afterwards, the second transformation is applied to this intermediate customized TEMPO-Analysis model (TA_To_MAST.atl). This transformation generates two models, (1) a MAST model that can be used as input to the MAST analysis tool, and (2) a mappings model (compliant to the MapIDs meta-model) that maps the names of the TEMPO-Analysis model elements to the names of their counterpart MAST model elements.

A diagram of the return MAST to TEMPO-Analysis transformation is shown in Fig. 4. It consists of one transformation (TA_MASTResults_MAPIDS_to_TA.atl). This transformation has three inputs, (1) a MAST-Results model that contains the worst-case response times of the steps, (2) the MapIDs model generated previously, and (3) the TEMPO-Analysis custom model that has also been generated previously. The output of this transformation is a TEMPO-Analysis model which now includes the worst-case response times of the steps.

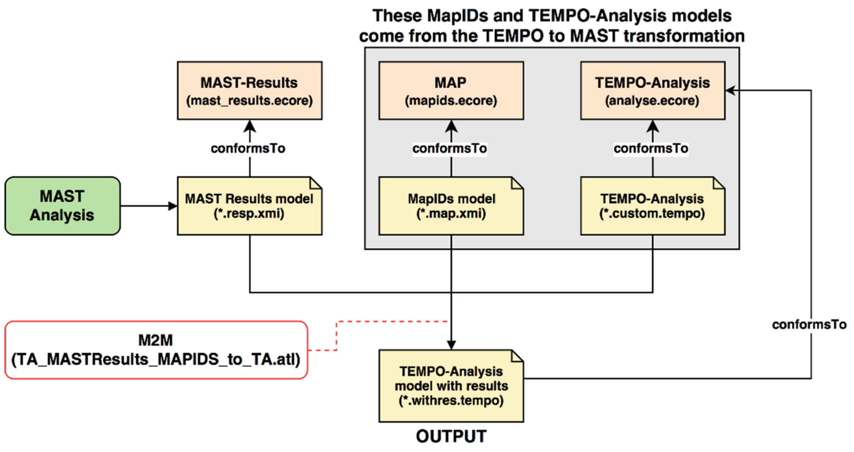
These MapIDs and TEMPO-Analysis models come from the TEMPO to MAST transformation
MAST-Results (mast_results.ecore)
MAP (mapids.ecore)
TEMPO-Analysis (analyse.ecore)
conformsTo
conformsTo
conformsTo
MAST Analysis
MAST Results model (*.resp.xmi)
MapIDs model (*.map.xmi)
TEMPO-Analysis (*.custom.tempo)
M2M (TA_MASTResults_MAPIDS_to_TA.atl)
conformsTo
TEMPO-Analysis model with results (*.withres.tempo)
OUTPUT

Fig. 4. MAST-Results to TEMPO-Analysis transformation

6 TEMPO Optimization Tools

The optimization tools are stand-alone tools written in Java that implement all the necessary infrastructure to automatically perform priority and architecture optimizations on TEMPO-Analysis models. The information in a TEMPO-Analysis model is complemented with a TEMPO-Constraints model that stores parameters that are not modelled by TEMPO-Analysis but must be specified for the optimization process.

6.1 Priority Optimization

The goal of the priority optimization is to find a suitable assignment of priorities (if possible) that makes the system schedulable. This assignment assumes that steps have been statically allocated to processors or networks.

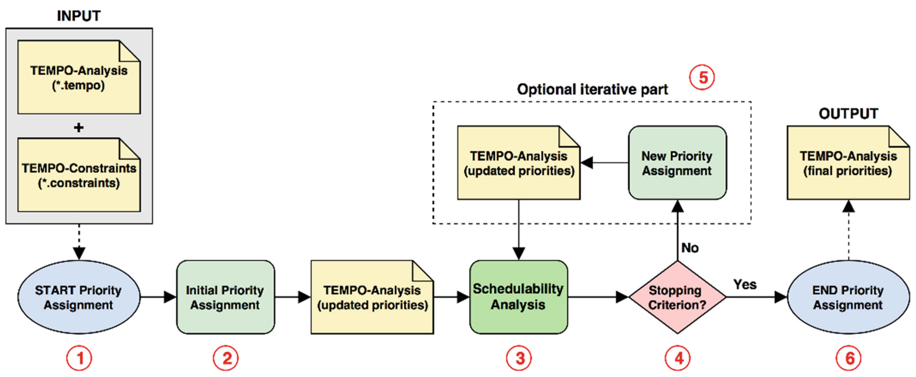
INPUT
TEMPO-Analysis (*.tempo)
+
TEMPO-Constraints (*.constraints)
Optional iterative part
5
TEMPO-Analysis (updated priorities)
New Priority Assignment
OUTPUT
TEMPO-Analysis (final priorities)
No
START Priority Assignment
Initial Priority Assignment
TEMPO-Analysis (updated priorities)
Schedulability Analysis
Stopping Criterion?
Yes
END Priority Assignment
1
2
3
4
6

Fig. 5. Design of the priority optimization tool

The priority optimization uses an iterative process that implements the HOSPA heuristic algorithm [17] included in the MAST tools. The flow of this optimization algorithm is depicted in Fig. 5, and is composed of the following stages:

1. The input of the priority optimization tool is a pair of TEMPO-Analysis and TEMPO-Constraints models. The parameters of the TEMPO-Constraints model relevant to the priority optimization are (1) the priority ranges from which the priorities of the steps can be selected, and (2) the list of steps that have pre-assigned priorities that cannot be changed.
2. The priority optimization process starts with an initial priority assignment. As with HOSPA, this initial assignment is a fast non-iterative algorithm that distributes the deadlines of the end-to-end flows among the tasks, and then the Deadline Monotonic criterion is applied. The TEMPO-Analysis input model is updated with these initial priorities.
3. The next step is to determine if this new priority assignment makes the system schedulable. This is achieved by applying a schedulability analysis tool on the TEMPO-Analysis model with the updated priorities.
4. Once the schedulability analysis has been performed, the priority optimization process finishes if any of the following stopping criteria is met:
   (a) The last priority assignment makes the system schedulable.
   (b) Two consecutive priority assignments were identical.
   (c) A pre-established maximum number of iterations have been reached. This number of iterations can be set to zero, thus making the priority optimization process non iterative, finishing with the initial assignment.
   (d) A pre-established maximum number of iterations on an already schedulable solution are reached. The priority optimization process has the capability of improving an already schedulable solution by iterating over it.
5. If no stopping criterion is met, a new priority assignment is made by using the same formulation as in HOSPA. This new assignment takes into account the previously calculated worst-case response times of the steps to reorganize the priorities in the system, giving higher priorities to those tasks that are farther from meeting their

deadlines. The TEMPO-Analysis model is updated with these new priorities, and the process continues with stage 3 of the priority optimization process (the schedulability analysis).

6. The priority optimization finishes by returning a TEMPO-Analysis model with the best priority assignment that was found, which could be schedulable or not, and its corresponding worst-case response times.

6.2 Architecture Optimization

The objective of the architecture optimization is to find an allocation of steps to processors or networks that makes the system schedulable. Secondary optimization criteria can be set, i.e., to balance the workload among the processors, or to minimize the inter-processor communications.

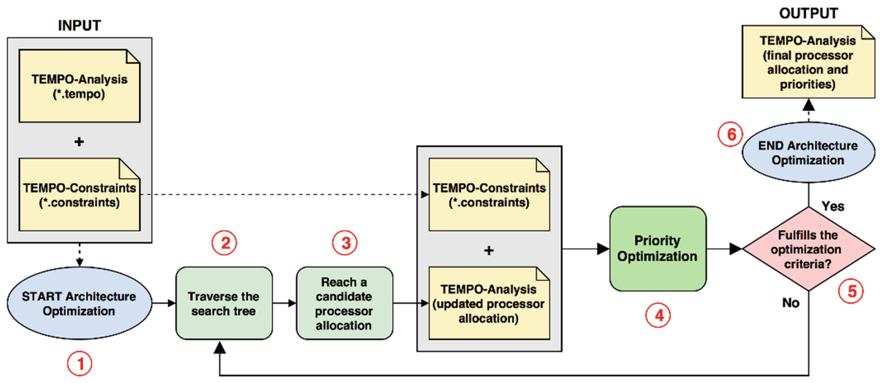
INPUT
TEMPO-Analysis (*.tempo)
+
TEMPO-Constraints (*.constraints)
START Architecture Optimization
1
Traverse the search tree
2
Reach a candidate processor allocation
3
TEMPO-Constraints (*.constraints)
+
TEMPO-Analysis (updated processor allocation)
Priority Optimization
4
Fulfills the optimization criteria?
5
Yes
No
END Architecture Optimization
6
OUTPUT
TEMPO-Analysis (final processor allocation and priorities)

Fig. 6. Design of the architecture optimization tool

As with the priority assignment, the architecture optimization process operates on a TEMPO-Analysis model complemented with a TEMPO-Constraints model. A diagram of the architecture optimization algorithm is depicted in Fig. 6 and comprises the following stages:

1. The input is a pair of TEMPO-Analysis and TEMPO-Constraints models. The parameters of the constraints model relevant to the architecture optimization are: (1) the step to processor affinities (subset of processors to which each step can be assigned to); (2) inter-processor latencies (communication latencies of messages between pairs of processors); and (3) default inter-processor latencies for processor pairs whose latencies were not explicitly specified. These inter-processor latencies can be interpreted as message transmission times if a full schedulability analysis of the networks is specified.
2. A brute-force search algorithm is intractable for non-trivial systems, thus an advanced algorithm is required. We are experimenting with a backtracking algorithm

that is providing promising results. The search tree is traversed and nodes are pruned according to their *a priori* likelihood of satisfying the optimization criteria (nodes with low *a priori* likelihood are pruned). This likelihood is determined taking into consideration indications such as the utilization in the processors, the number of messages that need to be transmitted, or a quick estimation of the worst-case response times.

3. By traversing the search tree, the algorithm reaches a candidate allocation. The input TEMPO-Analysis model is updated with this candidate step to processor or network allocation.
4. The priority optimization tool described in Sect. 6.1 is applied on the TEMPO-Analysis model with the updated architecture. The TEMPO-Constraints model is also provided. The priority optimization returns a TEMPO-Analysis model with the best priority assignment that could be found for this allocation, together with the associated worst-case response times.
5. The optimization process finishes when the resulting TEMPO-Analysis model meets the optimization criterion set by the user. The main criterion is that the system has reached schedulability. Secondary criteria that can be set are: (1) to minimize the number of processors used, (2) to balance the load, or (3) to minimize the inter-processor communications. A maximum number of iterations can also be set. If no criterion is met, the process continues on stage 2, continuing the tree traversal until another candidate allocation is reached.
6. When a stopping criterion is satisfied, the architecture optimization finishes by returning a TEMPO-Analysis model with an updated allocation, a priority assignment for this allocation, and the associated worst-case response times for this system configuration.

7 Industrial Case Study

In this section we present a case study consisting on the application of our MDE analysis and optimization strategy on a simplified version of an industrial system. The system is a robot controller composed of two specialized nodes (Teleoperation Station and Local Controller) connected via Ethernet. Both nodes and use fixed priority (FP) scheduling while the network has no contention given the offsets of the messages sent. The software consists of one end-to-end flow crossing both nodes and the network, and two independent tasks. Figure 7 shows an overview of the system, including the worst-case execution times (WCET) of each task.

The end-to-end flow is triggered by a periodic event with a period of 50 ms. The first task in the end-to-end flow is Trajectory Planner, executing in the Teleoperation Station. Once finished, this task sends a message to the Local Controller which launches the execution of task Command Manager. The flow continues with the execution of task Data Sender on the Local Controller, which sends a message to the Teleoperation Station, launching task Reporter, which is the last task in the flow. This task must finish at the latest 50 ms after the external event that triggered the flow arrived. Additionally, the system has a task (GUI) with a period of one second executing in the

Teleoperation Station, and another task (Servo Control) with a period of 5 ms executing in the Local Controller. These two tasks must finish before their next activation (deadline equal to their periods). The system also has three shared resources (mutual exclusion resources) that are accessed via the immediate ceiling protocol. These shared resources are named Commands, Servo Data, and Status. Which tasks access each shared resource, and for how long, is also shown in Fig. 7.

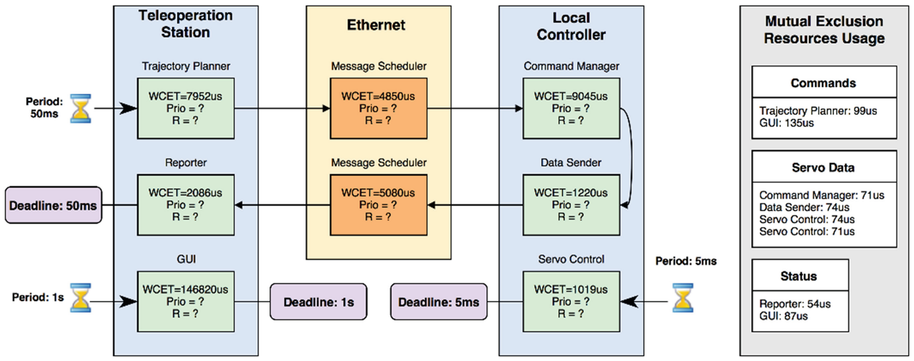

Fig. 7. Robot controller system

During the design phase inside our MDE strategy, the system is modeled with the TEMPO-MARTE meta-model. During this phase, the priorities (Prio) and associated worst-case response times of the tasks (R) are still unknown. Once all the tasks and timing requirements have been laid out, a designer of such system must determine these values of the system, so it can be guaranteed whether deadlines are going to be met in the worst-case. For this purpose, the TEMPO-MARTE model is transformed into another model that conforms to the TEMPO-Analysis meta-model, which is suited for the schedulability analysis. We now use the proposed analysis and optimization toolchain to achieve this goal.

The schedulability analysis tool is implemented as an Eclipse plug-in that operates on TEMPO-analysis (*.tempo) models. The optimization tool uses the same internal functionality as the schedulability analysis tool, but is implemented as a stand-alone tool that can operate without Eclipse. We model the robot controller system using the TEMPO-analysis meta-model, and then use this stand-alone tool to calculate a priority assignment and its associated worst-case response times. We also define a TEMPO-Constraints model specifying that the priorities should be selected in the range [1, 10]. Since each processor has at most three tasks, this range is large enough to guarantee that tasks don't share the same priority level. No architecture optimization is needed in this case because the tasks have a fixed allocation driven by their hardware requirements. The priorities and worst-case response times calculated by the tool are depicted in Fig. 8. Since the worst-case response times of the tasks (R) are lower than the associated deadlines, we can now determine that the system, with the priorities provided by the tool, is schedulable. It is worth noting that the worst-case response

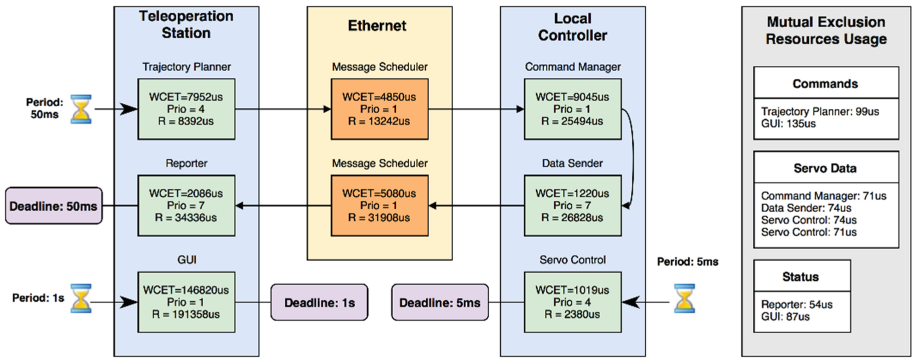

Fig. 8. Robot controller system with results (priorities and response times)

times are given as an upper bound of the latency from the arrival of the external event until the task finishes its execution.

Including all the underlying M2M transformations, the schedulability analysis of this system required approximately 1 s, while the priority assignment required 2 s.

8 Conclusions

We have presented a toolchain to perform, inside the Thales' TEMPO Ecore environment, three of the most important actions in the development of real-time systems: the schedulability analysis, the optimization of the priorities and the optimization of the allocation of tasks and messages to processors and networks. This toolchain implements the necessary M2M transformations to take advantage of the widely used MAST analysis tool, although the optimization tools can use any TEMPO-Analysis compatible schedulability analysis tool.

This toolchain is a result of the fruitful collaboration between Thales Research & Technology and the University of Cantabria (industry and academia). It represents a contribution to the industrial exploitation of model-driven technologies, schedulability analysis and optimization in the design of real-time systems in a variety of application domains.

Acknowledgment. This work has been funded in part by the Spanish Government under grant number TIN2014-56158-C4-2-P (M2C2).

References

1. Schmidt, D.C.: Model-driven engineering. IEEE Comput. **39**(2), 26–31 (2006)
2. Object Management Group: UML profile for MARTE: modeling and analysis of real time embedded systems, version 1.1. OMG document formal/2011-06-02 (2011)

3. Rioux, L., Henia, R., Sordon, N., González Harbour, M., Gutiérrez, J.J., Rivas, J.M., Cuevas, C., Drake, J.M., Medina, J.L.: Schedulability analysis and optimization in a model-based integrated toolchain: synthetic MARTE models for optimizing real-time design with MAST and TEMPO. In: Proceedings of the Forum on specification & Design Languages, FDL 2015, Barcelona, Spain, Demo Night Session (2015)
4. Mraidha, C., Tucci-Piergiovanni, S., Gérard, S.: Optimum: a MARTE-based methodology for schedulability analysis at early design stages. ACM SIGSOFT Softw. Eng.Notes **36**, 1–8 (2011)
5. Ouhammou, Y., Grolleau, E., Richard, M., Richard, P.: Towards a model-based approach guiding the scheduling analysis of real-time systems design. In: Proceedings of the 5th International WATERS Workshop, pp. 19–24 (2014)
6. Falkner, K., Chiprianov, V., Falkner, N.J., Szabo, C., Hill, J., Puddy, G., Fraser, D., Johnston, A., Rieckmann, M., Wallis, A.: Model-driven performance prediction of distributed real-time embedded defense systems. In: Proceedings of the 18th International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 155–158 (2013)
7. Pérez, H., Gutiérrez, J.J., Asensio, E., Zamorano, J., de la Puente, J.A.: Model-driven development of high-integrity distributed real-time systems using the end-to-end flow model. In: Proceedings of the 37th Euromicro SEAA Conference, pp. 209–216 (2011)
8. Henia, R., Rioux, L., Sordon, N., Garcia, G.-E., Panunzio, M.: Integrating formal timing analysis in the realtime software development process. In: WOSP 2015, pp. 35–40 (2015)
9. González Harbour, M., Gutiérrez, J.J., Palencia, J.C., Drake, J.M.: MAST: modeling and analysis suite for real time applications. In: Proceedings of the 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, pp. 125–134 (2001)
10. González Harbour, M., Gutiérrez, J.J., Drake, J.M., López, P., Palencia, J.C.: Modeling distributed real-time systems with MAST 2. J. Syst. Architect. **56**(6), 331–340 (2013). Elsevier
11. MAST. http://www.mast.unican.es
12. Tindell, K.W., Clark, J.: Holistic schedulability analysis for distributed hard real-time systems. Microprocessing Microprogramming **40**(2–3), 117–134 (1994)
13. Mäki-Turja, J., Nolin, M.: Efficient implementation of tight response-times for tasks with offsets. Real-Time Syst. J. **40**(1), 77–116 (2008)
14. Palencia, J.C., González Harbour, M.: Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In: Proceedings of the 20th Real-Time Systems Symposium, Phoenix, AZ, USA, pp. 328–339. IEEE (1999)
15. SymTA/S. https://www.symtavision.com/
16. Frédéric, J., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1), 31–39 (2008)
17. Rivas, J.M., Gutiérrez, J.J., Palencia, J.C., González Harbour, M.: Schedulability analysis and optimization of heterogeneous EDF and FP distributed real-time systems. In: Proceedings of the 23rd Euromicro Conference on Real-Time Systems, Porto, pp. 195–204 (2011)