Proceedings of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems

July 8th, 2014, Madrid, Spain

Held in conjunction with:
26th Euromicro Conference on Real-Time Systems
(ECRTS 2014)
8-11 July 2014

Edited By Julio Medina and Tullio Vardanega

Preface

Welcome to Madrid and to the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS'2014).

Broadly speaking, this workshop series pursues the mission to help bridge the gap between industrial practice and state-of-the-art research into methods for the rigorous, rapid and assured development of real-time embedded systems. Despite the evident disparity between the vastness of the problem area and the tiny size of the workshop crowd, the challenge is worth taking. This workshop series styles itself as intending to foster experimentation and practice for all software tools, methodologies, comprehensive data sets from real-world use cases, and representative benchmark suites that have potential for helping in the quest for mastering the rising complexity of modern systems.

A very distinct theme emerges for this fifth edition of the workshop series: the quest for benchmark-type solutions to help developers understand and dimension schedulable utilization in complex software systems. The sought solutions cover a broad space of possible approaches: ways to relate software and systems models (which precede and determine implementation): schedulability tests to support for exploring the implementation space (often called "platform" in the modelling speak); simulation tools to assist in the choice of scheduling algorithms when insufficient experience is available at the developer's end (and, for the case of multicore processors, also at the research end).

In keeping with its very nature, WATERS seeks solutions that are mature enough for industrial fruition or can be made so within modest time and effort. This theme distinctly emerged out of the push from two independent and compelling pressure points. One was the evidently common slant of most of the papers accepted for inclusion in the workshop program; the other was the research agenda in the back of the mind of the program co-chairs.

This year's program includes presentations, demonstrations and a panel discussion that all revolve around the above questions. It is very healthy and reassuring to see that the all of the three elements of the program were easy to fill with valuable contents. These proceedings reflect this worth and we hope the readers will see and enjoy that.

In closing these proceedings, we wish to thank the members of the WATERS technical program committee for their reviewing efforts, the authors for their submissions and their interest in this workshop as a community-building effort and as the means to advance their research work. Special thanks also go to the ECRTS 14 organizers, Juan Antonio de la Puente, Rob Davis, and Gerhard Fohler, for their support and helpful guidance.

Julio Medina and Tullio Vardanega

Program Co-Chairs of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems
July 2014

WATERS'14 Technical Program Committee

Program Committee

Table of Contents

This page has been left intentionally blank.

Implementing and Evaluating Various Response-Time Analyses for Mixed Messages in CAN using MPS-CAN Analyzer

Saad Mubeen*†, Jukka Mäki-Turja*† and Mikael Sjödin*
* Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden
† Arcticus Systems, Järfälla, Sweden
{saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se

***Abstract*—We integrate the Response Time Analysis (RTA) with offsets for mixed messages in Controller Area Network (CAN), where the CAN controllers implement abortable transmit buffers, with the MPS-CAN analyzer. Mixed messages are partly periodic and partly sporadic. They are implemented by several higher-level protocols for CAN that are used in the automotive industry. MPS-CAN analyzer is a free tool that supports several other existing RTA for periodic, sporadic and mixed messages in CAN. We perform extensive evaluation of the newly integrated analysis profile. Using the analyzer, we also perform a detailed comparative evaluation of various RTA for CAN.**

I. Introduction

Controller Area Network (CAN) [1] is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 Mbit/s. It has been standardized as ISO 11898-1 [2]. It is a widely used protocol in the automotive domain. There are several higher-level protocols for CAN that are developed for various industrial applications such as CAN Application Layer, CANopen, J1939, Hägglunds Controller Area Network (HCAN) and MilCAN. Often, CAN finds its applications in hard real-time systems that must ensure that their deadlines are met. For this purpose, *a priori* analysis techniques, such as schedulability analysis [3], [4], [5], have been developed. Response-Time Analysis (RTA) [3], [4], [5], [6] is a powerful, mature and well established schedulability analysis technique to calculate upper bounds on response times of tasks or messages in a real-time system or a network respectively. Tindell et al. [7] developed RTA for CAN which is later revised by Davis et. al [8].

A. Previous work and paper contribution

In our previous work [9] we presented first implementation of MPS-CAN Analyzer. It is the first and only freely-available tool that supports RTA of periodic, sporadic as well as *mixed* messages in CAN[1]. Mixed messages are partly periodic and partly sporadic. They are implemented by several higher-level protocols used in the industry. In [9], we discussed the implementation of basic RTA for mixed messages in CAN [11], whereas the implementation of other analyses was an ongoing work. Moreover, [9] did not discuss comparative evaluation of the extended analyses for mixed messages in CAN. In [12], we discuss the implementation of several other extensions of RTA for periodic, sporadic and mixed messages in CAN. These extensions support response-time calculations for messages scheduled with or without offsets; messages having arbitrary jitter and deadlines; CAN controllers implementing different queueing policies, e.g., priority and FIFO; and controllers implementing abortable or non-abortable transmit buffers. However, the implementation in [12] does not support analysis of mixed messages that are scheduled with offsets in the network where controllers implement abortable or non-abortable transmit buffers [13]. In this paper we implement RTA for CAN in the system where periodic and mixed messages can be scheduled with offsets while the CAN controllers implement abortable or non-abortable transmit buffers. We also improve the graphical layout of the tool to support better usability. Furthermore, we perform extensive evaluation of newly added analysis. We also perform a detailed comparative evaluation of various RTA for CAN and provide recommendations.

[1]A commercial tool implements basic analysis for mixed messages [10].

II. Mixed transmission patterns supported by higher-level protocols

There are several higher-level protocols and commercial extensions of CAN that support mixed transmission. In this transmission, the task that queues messages can be invoked periodically as well as sporadically. If a message can be queued for transmission periodically as well as sporadically, it is said to be mixed. In other words, a mixed message is simultaneously time- and event-triggered. We identify three different implementations of mixed messages by higher-level protocols for CAN used in the industry namely CANopen [14], AUTOSAR [15] and HCAN [16]. The transmission pattern of a mixed message in these protocols is shown in Fig. 1(a), 1(b) and 1(c) respectively. The down-pointing arrows symbolize queueing of messages while the upward lines (labeled with alphabetic characters) represent arrival of events.

The CANopen protocol supports mixed transmission that corresponds to the Asynchronous Transmission Mode coupled with the Event Timer. A mixed message can be queued for transmission at the arrival of an event provided the *Inhibit Time* has expired. The Inhibit Time is the minimum time that must be allowed to elapse between queueing of two consecutive messages. A mixed message can also be queued periodically at the expiry of the Event Timer. The Event Timer is reset every time the message is queued. Once a mixed message is queued, any additional queueing of it will not take place during the Inhibit Time [14].

AUTOSAR can be viewed as a higher-level protocol if it uses CAN for network communication. Mixed transmission mode in AUTOSAR is widely used in practice. In AUTOSAR, a mixed message can be queued for transmission repeatedly with a time period. The mixed message can also be queued at the arrival of an event provided the Minimum Delay Time (MDT) has expired. However, each transmission of a mixed message, regardless of being periodic or sporadic, is limited by the MDT. This means that both periodic and sporadic transmissions are delayed until the MDT expires.

A mixed message in the HCAN protocol contains signals out of which some are periodic and some are sporadic. A mixed message is queued for transmission not only periodically, but also as soon as an event occurs that changes the value of one or more event signals, provided the Minimum Update Time (MUT) between the queueing of two successive sporadic instances of the mixed message has elapsed. Hence, the transmission of a mixed message due to arrival of events is constrained by the MUT.

In CANopen, the Event Timer is reset with every mixed transmission. The implementation of a mixed message in AUTOSAR is similar to CANopen to some extent. The main difference is that the periodic transmission can be delayed until the expiry of the MDT in AUTOSAR as indicated in Fig. 1(b). Whereas in CANopen, the periodic transmission is not delayed, in fact, the Event Timer is restarted with every sporadic transmission as shown in Fig. 1(a). The MDT timer is started with every periodic or sporadic transmission of a mixed message. Hence, the worst-case periodicity of a mixed message in CANopen and AUTOSAR can never be higher than the Inhibit Timer and MDT respectively. As a result, the mixed message can be treated as a special case of sporadic transmission. Therefore, all existing RTA are still applicable.

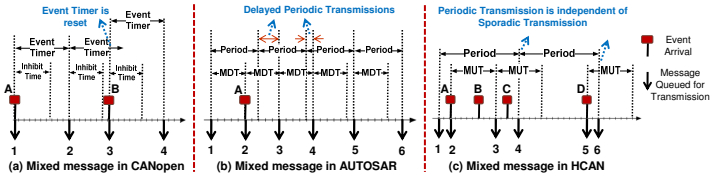

Fig. 1. Mixed transmission pattern in higher-level protocols for CAN

However, the periodic transmission is independent of the sporadic transmission in the HCAN protocol. The periodic timer is not reset with every sporadic transmission. A mixed message can be queued for transmission even if the MUT is not expired, e.g., see the transmission of instances 4 and 6 of the mixed message in Fig. 1(c). This indicates that the periodic transmission of a mixed message cannot be interfered by its sporadic transmission which is unlike in CANopen and AUTOSAR. The worst-case periodicity of a mixed message is neither bounded by the period nor by the MUT. Therefore, the existing analyses cannot be applied in this case. To the best of our knowledge, there is no free tool except for the MPS-CAN analyzer that analyzes this type of mixed messages.

III. Buffer Limitations and Queueing Policies

The different types of queueing polices implemented by CAN device drivers and communications stacks, internal organization, and hardware limitations in CAN controllers can have significant impact on the timing behavior of CAN messages. If an Electronic Control Unit (ECU) transmits more messages compared to the number of transmit buffers, the messages may be subjected to extra delay and jitter due to priority inversion.

A. Abortable transmit buffers

Let us consider the case in which the CAN controllers support transmission abort requests, e.g., Atmel AT89C51CC03/AT90CAN32/64 and Microchip MPC2515 [17]. In order to demonstrate an additional delay due to priority inversion in this case, consider the example of a message set shown in Fig. 2(a). Assume there are three nodes CC_c, CC_j and CC_k in the system and each node has three transmit buffers. m_1 is the highest priority message in the node CC_c as well as in the system. When m_1 becomes ready for transmission in the message queue, a lower priority message m_6 belonging to node CC_k is already under transmission. m_6 cannot be preempted because CAN uses fixed priority non-preemptive scheduling. This represents the blocking delay for m_1. At this time, all transmit buffers in CC_c are occupied by lower priority messages (say m_3, m_4 and m_5). The device drivers signal an abort request for the lowest priority message in the transmit buffers of CC_c) that is not under transmission. Hence, m_5 is aborted and copied from the transmit buffer to the message queue, whereas m_1 is moved to the vacated transmit buffer. The time needed to do the swapping is identified as *swapping time* in Fig. 2(a). A series of events may occur during the swapping : m_6 finishes its transmission, new arbitration round starts, message m_2 belonging to node CC_j and having priority lower than m_1 wins the arbitration and starts its transmission. Thus m_1 has to wait in the transmit buffer until m_2 finishes its transmission. This results in the priority inversion for m_1 and adds an extra delay to its response time. In [18], Khan et al. pointed out that this extra delay of the higher priority message appears as its additional jitter to the lower priority messages, e.g., m_5 in Fig. 2(a).

1) Discussion on message copy time and delay: If the message copy time is smaller than or equal to the inter-frame space (i.e., time to transmit 3 bits on CAN bus or $3*\tau_{bit}$ time), a lower priority message in the transmit buffer (that is not under transmission) can be swapped with a higher priority message in the message queue before transmission of the next frame [1]. Hence, there will be no priority inversion. This means that the message copy time must be, at least, $4*\tau_{bit}$ for the priority inversion to occur. In Legacy systems, there may be slow controllers, i.e., the speed of the controllers can be slower than the maximum operating speed of the CAN bus (1 Mbit/s). Since the amount of data transmitted in a CAN message ranges from 0 to 8 bytes, the transmission time of a message also varies accordingly. According to [8], the transmission time of a CAN message with standard frame format ranges from $55*\tau_{bit}$ to $135*\tau_{bit}$ for the amount of data contained in the message that ranges from 0 to 8 bytes respectively. Intuitively, the message copy time of $4*\tau_{bit}$ can range from 7.3% to 3% of transmission time of a message with 0 to 8 bytes of data respectively. Due to slow controllers in legacy systems, the message copy time can be greater than $4*\tau_{bit}$, hence, higher than 7.3% of its transmission time.

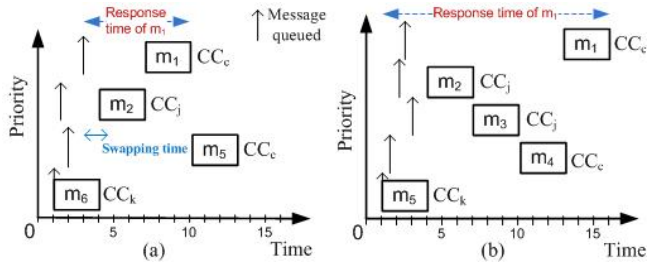

Fig. 2. Demonstration of priority inversion in the case of (a) abortable transmit buffers, (b) non-abortable transmit buffers

B. Non-abortable transmit buffers

Now we consider the case in which the CAN controllers implement non-abortable transmit buffers, e.g., Philips 82C200 [19], [20], [7]. Consider an example of three controllers CC_c, CC_j, CC_k connected to a single CAN network in Fig. 2 (b). Let m_1, belonging to CC_c, be the highest priority message in the system. Assume that when m_1 is ready to be queued, all transmit buffers in CC_c are occupied by lower priority messages which cannot be aborted because the controllers implement non-abortable transmit buffers. In addition, m_1 can be blocked by any lower priority message because the lower priority message already started its transmission. In this example m_1 is blocked by m_5 that belongs to node CC_k. Since all transmit buffers in CC_c are full, m_1 has to wait in the message queue until one of the messages in the transmit buffers of node CC_c is transmitted.

Let m_4 be the highest priority message in the transmit buffers of node CC_c. m_4 can be interfered by higher priority messages (m_2 and m_3) belonging to other nodes. Hence, it can be seen that priority inversion for m_1 takes place

because m_1 cannot start its transmission before m_4 finishes its transmission, while m_4 has to wait until messages m_2 and m_3 are transmitted. This adds an additional delay to the worst-case response time of m_1. In this example, this additional delay is the sum of the worst-case transmission times of m_2, m_3 and m_4. This additional delay appears as additional jitter of m_1 as seen by the lower priority messages.

C. Priority and FIFO queues

The most natural queuing policy suited to CAN nodes is priority-based queuing. However, due to simplicity of FIFO policy some CAN controllers implement FIFO queues, e.g., Microchip PIC32MX, Infineon XC161CS, Renesas R32C/160 and XILINX LogiCORE IP AXI Controller [17], [18]. In case of nodes implementing priority queues, each node selects the highest priority message from its transmit buffers while entering into the bus arbitrations. The highest priority message among them wins the bus arbitration. On the other hand, when the nodes implement FIFO queues, the oldest message in the transmit queue of each node competes for the bus. However, the bus arbitration among these messages is done on priority basis. Consider an example of three nodes that are connected to a single CAN network as shown in Fig. 3. Assume that Node A sends the messages m_1, m_3 and m_5; Node B sends the messages m_2, m_4 and m_9; and Node C sends the messages m_6, m_7 and m_8. The priority of a message is indicated by its subscript (smaller the subscript, the higher the priority).

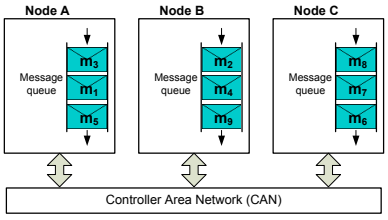

Fig. 3. Example to demonstrate different queueing policies

Let the nodes implement priority queues. In the first round, Nodes A, B, and C pick messages m_1, m_2 and m_6 respectively. m_1 wins the arbitration because of higher priority and is transmitted over the network as shown in Fig. 4. In the second round, Nodes A, B, and C pick messages m_3, m_2 and m_6 respectively. m_2 wins the arbitration and is transmitted over the network. Similar priority-based selection and arbitration occur during the rest of the rounds as shown in Fig. 4.

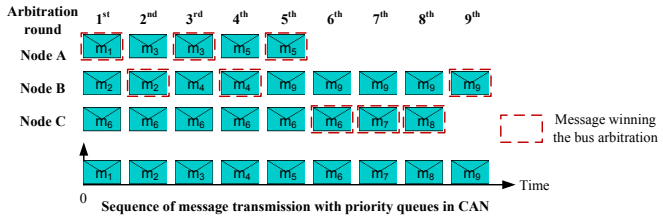

Fig. 4. priority-based queues and CAN arbitration

Now we assume that the nodes implement FIFO queues. In the first round, Nodes A, B, and C pick the oldest messages m_5, m_9 and m_6 respectively. m_5 wins the bus arbitration due to its higher priority and is transmitted as shown in Fig. 5. In the second round, Nodes A, B, and C pick messages m_1, m_9 and m_6 respectively. This time, m_1 wins the bus arbitration and is transmitted over the network. Similar FIFO selection and priority-based arbitration occur during the rest of the rounds as shown in Fig. 5. It can be seen that the priorities of messages are sometimes not respected in the FIFO queue within a node, e.g., a lower priority message m_5 is transmitted before the higher priority message m_1 as shown in Fig. 5. This results in priority inversions due to which higher priority messages may have very large response times, e.g., different response time of m_2 in the systems with priority and FIFO queues in Fig. 4 and Fig. 5 respectively.

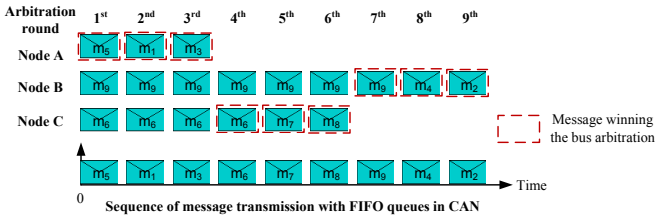

Fig. 5. FIFO-based queues and CAN arbitration

IV. Related Work and Implemented Analysis

A. Related work

In [21], Davis et al. extend the analysis of [7], [8] which is now applicable to the CAN network where some nodes implement priority queues and some implement FIFO queues. The message deadlines in [21] are assumed to be smaller than or equal to the corresponding periods. This assumption is lifted in [22] by by supporting the analysis of messages with arbitrary deadlines. Moreover, they extend their work to support RTA of CAN for FIFO and work-conserving queues. The analysis in [7], [8] assumes that the CAN controllers have very large transmit buffers. However, most CAN controllers have small number of transmit buffers [23], [22]. If all buffers in the controller are occupied by lower priority messages, a higher priority message released in the same controller may suffer from priority inversion [7], [18], [20], [24]. The analysis in [7], [8] has been extended in [18] and [23] to support the analysis of network that contain abortable and non-abortable transmit buffers in the controllers respectively. Most of the CAN enabled ECUs support transmit abort requests [18].

All these analyses assume that the messages are queued for transmission periodically or sporadically. Mubeen et al. [11] extend the existing analysis [7], [8] to support mixed messages in CAN where nodes implement priority queues. Mubeen et al. [25] further extend their analysis to support mixed messages in the network where some nodes implement priority queues while others implement FIFO queues. RTA for mixed messages in CAN [11] has been extended to support the analysis of network that contain abortable and non-abortable transmit buffers in the controllers in [26] and [27] respectively. But, none of the analyses discussed above supports messages that are scheduled with offsets i.e., using externally imposed delays between the times when the messages can be queued. In order to avoid deadlines violations due to high transient loads, current automotive embedded systems are often scheduled with offsets [28]. The worst-case response-times of lower priority messages in CAN can be reduced if the messages are scheduled with offsets [29], [30]. A method for the assignment of offsets to improve the overall bandwidth utilization is proposed in [30]. RTA with offsets for CAN has been developed by several researchers [31], [32], [29], [33], [28].

None of the above analyses supports mixed messages that are scheduled with offsets. Offset-based analysis [31] is extended in [34] to support response-time calculations for mixed messages in CAN. However, this analysis is restricted due to limitations regarding message jitter and deadlines. The source of these limitations comes from the base analysis [31]. In [35], Mubeen et al. removed these limitations and extended the analysis for mixed messages [11] with offsets [28]. Mubeen et al. further extend the analysis for mixed messages with offsets in CAN supporting abortable transmit buffers [13].

B. Related tools

VNA [36] is a communication design tool that supports RTA for CAN. It implements RTA of CAN developed by Tindell et al. [7]. Vector [37] is a tools provider for the development of networked electronic systems. CANalyzer [38] supports the simulation, analysis and data logging for the systems that use CAN. CANoe [39] is a tool for simulation of functional and extra-functional (e.g., timing) behavior of ECU networks. Network Designer CAN is another tool by Vector that is able to perform timing analysis of CAN. SymTA/S [40] is a tool for model-based timing analysis and optimization. Among other analyses, it supports statistical, worst- and best-case timing analyses for CAN. RTaW-Sim [41] is a tool for the simulation and performance evaluation of the CAN network. The Rubus-ICE is a commercial tool suite developed by Arcticus Systems [42] in close collaboration with Mälardalen University Sweden. Among other analyses, it supports RTA of CAN [7], [8] and RTA of CAN for mixed messages[11], [43]. To the best of our knowledge, there is no freely-available tool that implements RTA of CAN for mixed messages. The main purpose of MPS-CAN Analyzer is to support RTA of periodic, sporadic and mixed messages in CAN. The analyses implemented in MPS-CAN analyzer are shown in Fig. 6.

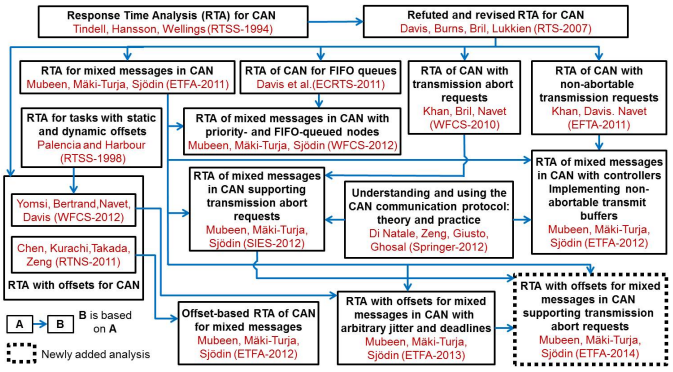

Fig. 6. Graphical representation of Response Time Analysis (RTA) and its extensions implemented in MPS-CAN Analyzer

C. Implementation and distribution

The tool is implemented in C language. Each analysis profile supported by the tool is implemented as a separate C file. The Layout of the tool is shown in Fig. 7. It has a scope for further extensions in the future. The link to the tool can be found at https://github.com/saadmubeen/MPS-CAN.

V. Evaluation of various RTA for CAN

A. Experimental setup

The system consists of six ECUs that are connected to the CAN network. The speed of the network is set to 250 Kbit/s. There are 60 messages in the system. The message set is generated from the NETCARBENCH tool [44] which is a benchmark used in the design of automotive embedded systems. It should be noted that NETCARBENCH cannot generate mixed messages. We randomly assign mixed, periodic, and sporadic transmission types to 40%, 30%, and 30% generated messages respectively. This means, there are 24 mixed, 18 periodic and 18 sporadic messages in the system. The messages are equally distributed among the ECUs, i.e., each ECU sends 4 mixed, 3 periodic and 3 sporadic messages over the network. All the attributes of these messages are tabulated in the Fig. 8. The attributes of a message m_m are identified as follows. The priority, sender node ID, transmission type, number of data bytes in the message, offset, jitter, period, minimum update time and deadline are represented by P_m, CC_m, ξ_m, s_m, O_m, J_m, T_m, MUT_m and D_m respectively. All timing values in the table are expressed in milliseconds. We perform a number of tests on the message set. The network bandwidth utilization calculated by MPS-CAN analyzer for this message set in each test is equal to 59.203793%.

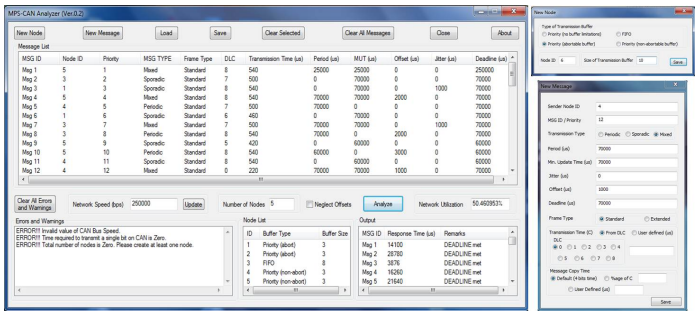

Fig. 7. MPS-CAN Analyzer layout, inputs and outputs

P_m	CC_m	ξ_m	s_m	O_m	J_m	T_m	MUT_m	D_m	P_m	CC_m	ξ_m	s_m	O_m	J_m	T_m	MUT_m	D_m	P_m	CC_m	ξ_m	s_m	O_m	J_m	T_m	MUT_m	D_m
1	5	M	8	0	0	25	25	25	21	4	M	8	3	0	70	70	70	41	2	M	1	7	1	70	70	70
2	3	S	7	0	0	0	70	70	22	1	M	0	4	1	60	60	60	42	1	P	1	6	0	70	0	70
3	1	S	8	0	1	0	70	70	23	2	S	0	0	1	0	70	70	43	4	S	8	0	2	0	80	80
4	5	M	8	2	0	70	70	70	24	3	S	6	0	0	0	70	70	44	5	S	8	0	2	0	70	70
5	4	P	7	0	0	70	0	70	25	3	M	8	5	1	70	70	70	45	6	S	8	0	2	0	70	70
6	1	S	6	0	0	0	70	70	26	2	P	6	3	0	70	0	70	46	3	M	2	8	1	80	80	80
7	3	M	7	0	1	70	70	70	27	5	M	2	7	1	60	60	60	47	3	S	4	0	2	0	70	70
8	3	P	8	2	0	70	0	70	28	4	P	1	5	0	80	0	80	48	6	M	8	7	2	70	70	70
9	5	S	5	0	0	0	60	60	29	3	M	6	5	0	70	70	70	49	1	M	8	8	1	70	70	70
10	5	P	8	3	0	60	0	60	30	1	P	1	5	0	70	0	70	50	6	M	7	8	1	70	70	70
11	4	S	8	0	0	0	60	60	31	2	M	7	4	1	70	70	70	51	6	P	8	0	2	70	0	70
12	4	M	0	1	0	70	70	70	32	1	S	8	0	0	0	70	70	52	6	P	6	2	1	70	0	70
13	1	M	6	2	0	60	60	60	33	2	S	8	0	0	0	70	70	53	6	S	1	0	1	0	70	70
14	3	P	8	3	0	50	0	50	34	2	P	8	5	2	80	0	80	54	6	P	2	3	0	70	0	70
15	5	M	8	4	0	70	70	70	35	2	P	5	5	0	60	0	60	55	6	S	1	0	1	0	70	70
16	4	M	5	4	0	50	50	50	36	4	S	8	0	1	0	70	70	56	6	M	2	4	1	70	70	70
17	2	S	8	0	1	0	80	80	37	1	P	5	6	1	70	0	70	57	5	S	8	0	2	0	20	80
18	2	M	8	1	0	70	70	70	38	4	P	1	6	1	80	0	80	58	1	M	8	7	2	70	70	70
19	5	P	8	4	0	70	0	70	39	3	P	8	7	1	80	0	80	59	6	M	8	8	1	70	70	70
20	5	P	7	5	1	70	0	70	40	4	M	0	7	1	70	70	70	60	2	M	7	8	1	70	70	70

Fig. 8. Attributes of the message set under analysis

B. Comparison of various RTA for CAN

In this subsection, we perform five different tests as follows.

1) All ECUs implement priority queuing policy while the number of transmit buffers are large enough to avoid aborting transmissions. The message set is analyzed using the RTA for mixed messages in CAN with no buffer limitations [11].
2) All ECUs implement priority queuing policy and abortable transmit buffers. The message set is analyzed using the RTA for mixed messages in CAN with abortable transmit buffers [26], [13].
3) All ECUs implement priority queuing policy and non-abortable transmit buffers. The message set is analyzed using the RTA for mixed messages in CAN with non-abortable transmit buffers [27].
4) All ECUs implement FIFO queues. The message set is analyzed using the RTA for mixed messages in CAN with FIFO queues [25].
5) Heterogeneous system: two ECUs implement priority queuing policy and abortable transmit buffers; two ECUs implement priority queuing policy and non-abortable transmit buffers; and two ECUs implement FIFO queues. The MPS-CAN analyzes each ECU differently using the corresponding analysis profile from the above three tests.

Response times of the messages calculated in all five tests are plotted in Fig. 9. It can be seen that the response times are lowest (best) in the first test because we have considered ideal conditions (no buffer limitations in the CAN controllers). The second test results in the second best response times. However, they are higher compared to the first test due to extra delay from priority inversion due to transmission abort requests. The

third test results in overall third best response times (with some exceptions). It can be seen that the extra delay due to priority inversion in non-abortable transmit buffers is higher compared to abortable transmit buffers. The fourth test yields the highest response times because of high buffering time and delays due to priority inversion in FIFO queues. Furthermore, the response times of messages are significantly high compared to the rest of the tests. The response times in the heterogeneous system are higher compared to first three tests but significantly lower than the fourth test where FIFO queues are used. From the results, one can infer that the ECUs that implement FIFO queues in the CAN controllers should be avoided. In order to calculate correct (not optimistic) response times, the RTA for CAN should correctly match the queueing policy and practical limitations in the CAN controllers.

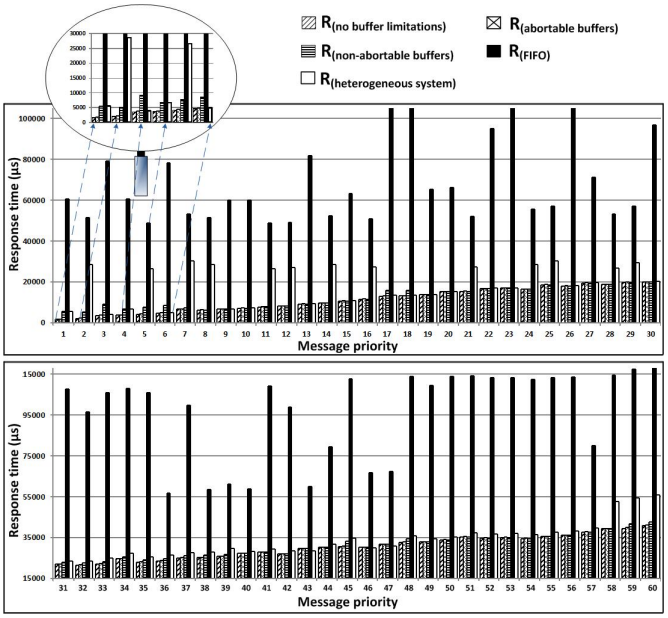

Fig. 9. Analysis results using various RTA for CAN

C. Effect of message copy time on schedulability

In this subsection, we compare the effect of message copy times on their response times. We perform four tests where all ECUs implement abortable transmit buffers. However, the message copy times are different in these tests. In the first test, the message copy time for each message is equal to $4*\tau_{bit}$ time (see Subsection III-A1). In the rest of the tests, it is 10, 20 and 30 percent of corresponding transmission times of messages respectively. The calculated response times of the messages in all test are plotted in Fig. 10. The results indicate that the increase in the response times of messages is directly proportional to the increase in the amount of message copy times. If the message copy time is less than the inter-frame space (time required to transmit 3-bits of data on CAN), the response times of messages in the system with abortable transmit buffers becomes equals to the response times of same messages in the system with no buffer limitations.

D. Effect of offsets on schedulability

Finally, we perform four more tests on the message set to explore the effect of offsets on the schedulability of the message set. In the first two tests, the message set is analyzed using RTA for mixed messages in CAN with no buffer limitations [11] and with abortable transmit buffers [26]. However the offsets of all messages are assumed to be zero. There is a newly added check box "Neglect offsets" in the MPS-CAN analyzer as shown in Fig. 7 that actually neglects the offsets when analyzing the messages. In the next two tests, the first two tests are repeated while considering message offsets. Also, the messages are analyzed using the newly implemented RTA with offsets for mixed messages in CAN supporting transmission abort requests [13]. The response times calculated in the four tests are plotted in Fig. 11. The results indicate that the response times can be reduced when messages are scheduled with offsets. We observe 2.462% improvement in the schedulability of the system when the messages are scheduled with offsets. As discussed earlier, NETCARBENCH cannot generate mixed messages, hence, the offsets assigned to the mixed messages are not optimal. The schedulability can be further improved if an optimal offset assignment algorithm for mixed messages is used. The percentage improvement in schedulability is calculated as follows.

$$\left[\left(\sum_{\forall m_m \in \aleph}\left[\frac{R_m^{\{no-offset\}} - R_m^{\{offset\}}}{D_m}\right]\right) \Big/ (sizeof(\aleph))\right] * 100$$

Where, m_m, R_m, D_m and $\aleph$ represent a message, response time, deadline and the set of all messages in the system respectively.

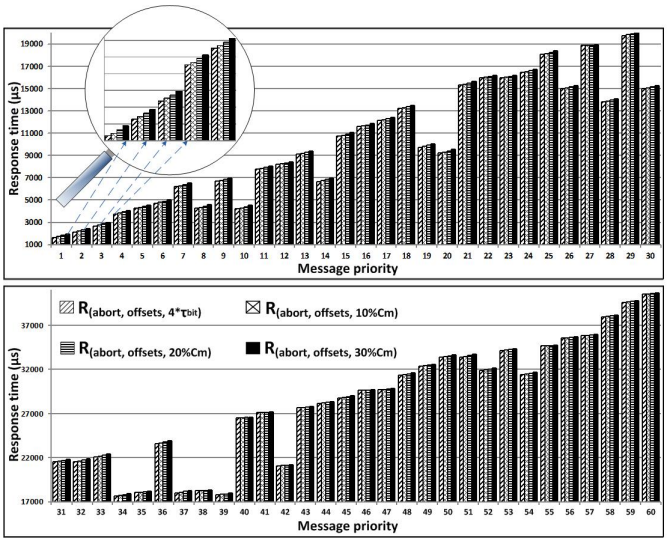

Fig. 10. Analysis results: effect of message copy time on schedulability

VI. Conclusion

We implemented a new RTA for CAN in a free tool MPS-CAN analyzer. The implemented RTA supports periodic, sporadic as well as mixed messages in the system where transmission abort requests in CAN controllers all allowed. Mixed messages are partly periodic and partly sporadic and are implemented by several higher-level protocols for CAN that are used in the automotive industry today. We conducted a number of tests to perform detailed evaluation of all RTA profiles available in MPS-CAN analyzer. These analyses consider various aspects and practical limitations such as mixed messages; messages scheduled with offsets; messages with arbitrary jitter and deadlines; priority or FIFO queueing policies; limitations of transmit buffers in CAN controllers such abortable or non-abortable; and heterogeneous systems that consist of different types of ECU's. We can make several recommendations based on the analyses results and their evaluation. The ECUs that implement FIFO queues should be avoided because of high buffering time and delays due to priority inversion in FIFO queues. Due to lower response times, the controllers that implement abortable transmit buffers should be preferred over those that implement non-abortable

transmit buffers. The schedulability of the system can be improved if messages are scheduled with offsets. We observed 2.462% improvement in schedulability in one of the tests when the messages are scheduled with offsets. Finally, it can be concluded that if RTA for CAN does not correctly account for transmission patterns by higher-level protocols, queueing policies and practical limitations in the CAN controllers, the calculated response times of messages can be optimistic.

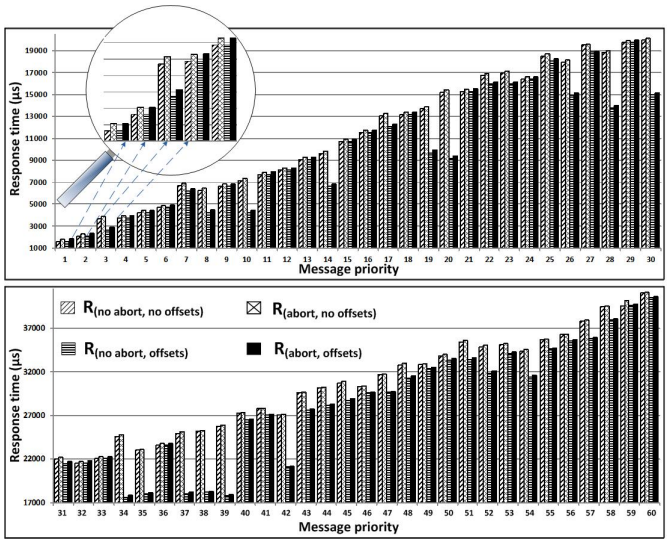

Fig. 11. Analysis results: effect of offsets on schedulability

ACKNOWLEDGEMENT

This work is supported by the Swedish Research Council within the project SynthSoft. The authors thank the industrial partners Arcticus Systems, BAE Systems Hägglunds and Volvo CE, Sweden.

REFERENCES

[1] Robert Bosch GmbH, "CAN Specification Version 2.0," postfach 30 02 40, D-70442 Stuttgart, 1991.
[2] ISO 11898-1, "Road Vehicles interchange of digital information controller area network (CAN) for high-speed communication, ISO Standard-11898, Nov. 1993."
[3] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, 1993.
[4] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historic perspective," *Real-Time Systems*, vol. 8, no. 2/3, pp. 173–198, 1995.
[5] L. Sha, T. Abdelzaher, K.-E. A. rzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, vol. 28, no. 2/3, pp. 101–155, 2004.
[6] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.
[7] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communications: controller area network (CAN)," in *Real-Time Systems Symposium (RTSS) 1994*, pp. 259 –263.
[8] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, pp. 239–272, 2007.
[9] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Many-in-one Response-Time Analyzer for Controller Area Network," in *In WATERS workshop*, 2013.
[10] "Rubus-ICE," http://www.arcticus-systems.com.
[11] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending schedulability analysis of controller area network (CAN) for mixed (periodic/sporadic) messages," in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2011.
[12] S. Mubeen, J. Mäki-Turja and M. Sjödin, "MPS-CAN Analyzer: Integrated Implementation of Response-Time Analyses for Controller Area Network," *Journal of Systems Architecture*, 2014.
[13] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Response Time Analysis with Offsets for Mixed Messages in CAN Supporting Transmission Abort Requests," in *19th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2014.
[14] "CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Version 4.02. February 13, 2002," http://www.can-cia.org/index.php?id=440.
[15] "AUTOSAR Techincal Overview, Version 2.2.2., Release 3.1, The AUTOSAR Consortium, Aug., 2008," http://autosar.org.
[16] "Hägglunds Controller Area Network (HCAN), Network Implementation Specification," *BAE Systems Hägglunds, Sweden (internal document)*, April 2009.
[17] R. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Schedulability analysis for controller area network (can) with fifo queues priority queues and gateways," *Real-Time Systems*, vol. 49, no. 1, pp. 73–116, 2013.
[18] D. Khan, R. Bril, and N. Navet, "Integrating hardware limitations in can schedulability analysis," in *8th IEEE International Workshop on Factory Communication Systems (WFCS)*, may 2010, pp. 207 –210.
[19] D. Khan, R. Davis, and N. Navet, "Schedulability analysis of CAN with non-abortable transmission requests," in *16th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, sept. 2011, pp. 1–8.
[20] M. D. Natale, "Evaluating message transmission times in controller area networks without buffer preemption," in *8th Brazilian Workshop on Real-Time Systems*, 2006.
[21] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller Area Network (CAN) Schedulability Analysis with FIFO queues," in *23rd Euromicro Conference on Real-Time Systems*, July 2011.
[22] R. Davis and N. Navet, "Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-Conserving Queues," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, may 2012, pp. 33 –42.
[23] D. Khan, R. Davis, and N. Navet, "Schedulability analysis of CAN with non-abortable transmission requests," in *16th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, sept. 2011.
[24] Marco Di Natale, Haibo Zeng, Paolo Giusto, Arkadeb Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. Springer, 2012.
[25] S. Mubeen, J. Mäki-Turja and M. Sjödin, "Response-Time Analysis of Mixed Messages in Controller Area Network with Priority- and FIFO-Queued Nodes," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, May 2012.
[26] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Response Time Analysis for Mixed Messages in CAN Supporting Transmission Abort Requests," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2012.
[27] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending response-time analysis of mixed messages in can with controllers implementing non-abortable transmit buffers," in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2012.
[28] P. Yomsi, D. Bertrand, N. Navet, and R. Davis, "Controller Area Network (CAN): Response Time Analysis with Offsets," in *9th IEEE International Workshop on Factory Communication Systems*, May 2012.
[29] A. Szakaly, "Response Time Analysis with Offsets for CAN," Master's thesis, Department of Computer Engineering, Chalmers University of Technology, Nov. 2003.
[30] M. Grenier, L. Havet, and N. Navet, "Pushing the limits of can-scheduling frames with offsets provides a major performance boost," in *4th European Congress on Embedded Real Time Software*, 2008.
[31] Y. Chen, R. Kurachi, H. Takada, and G. Zeng, "Schedulability comparison for can message with offset: Priority queue versus fifo queue," in *19th International Conference on Real-Time and Network Systems (RTNS)*, Sep. 2011, pp. 181–192.
[32] L. Du and G. Xu, "Worst case response time analysis for can messages with offsets," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, nov. 2009, pp. 41 –45.
[33] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.
[34] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Worst-case response-time analysis for mixed messages with offsets in controller area network," in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2012.
[35] S. Mubeen, J. Mäki-Turja and M. Sjödin, "Extending offset-based response-time analysis for mixed messages in controller area network," in *18th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2013.
[36] "Volcano Network Architect (VNA). Mentor Graphics," http://www.mentor.com/products/vnd/communication-management/vna.
[37] "Vector. http://www.vector.com."
[38] "CANalyzer. http://www.vector.com/vi_canalyzer_en.html."
[39] "CANoe. www.vector.com/portal/medien/cmc/info/canoe_productinformation_en.pdf."
[40] A. Hamann, R. Henia, R. Racu, M. Jersak, K. Richter, and R. Ernst, "Symta/s - symbolic timing analysis for systems," 2004.
[41] "RTaW-Sim. http://www.realtimeatwork.com/software/rtaw-sim."
[42] "Arcticus Systems AB," http://www.arcticus-systems.com.
[43] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Computer Science and Information Systems, ISSN: 1361-1384*, vol. 10, no. 1, 2013.
[44] C. Braun, L. Havet, and N. Navet, "Netcarbench: A benchmark for techniques and tools used in the design of automotive communication systems," in *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems*, Nov. 2007.

A System-level Framework for the Evaluation of the Performance Cost of Scheduling and Communication Delays in Control Systems

Matteo Morelli, Fabio Cremona, Marco Di Natale
Scuola Superiore S. Anna, Pisa

***Abstract*—The real-time systems community invests significant efforts in the study and analysis of hard real-time systems, decoupling the functional correctness of the system from the verification of the timing (schedulability) properties. In reality, many control systems are tolerant to delays and deadline misses, and the scheduling choices affect the system performance in a way that is different from a simple binary (safe/not safe) outcome. The Truetime Simulink package developed at Lund is probably the best known attempt at the integration of scheduling and functional control simulation, allowing for the analysis of the performance impact of computation and communication delays on systems. In this paper we present an extension to the TrueTime project that attempts at better modularity and possible integration in a Model-driven flow in which parts of the model are automatically generated from SysML specifications. The project is still under development but its early version, for single- and multi-core systems is freely available as open source.**

I. Introduction

Most Real-time schedulability theory and analysis tools work under the assumption of a separation of concerns between the functional side of the problem and the time properties of the computations. In the envisioned flow, designers select the functional model of the controls in such a way that the outputs are stable within the assigned periods and deadlines, then select and purchase the execution hardware and produce a task (code) implementation that realizes the control law. The software code is then analyzed on the selected platform in order to verify that the time assumptions made at the times the controls are designed are satisfied and all tasks complete with the assigned rates within their deadlines.

This model has several issues and, even in the best case, it portrays an approximate (and sometimes inaccurate) view of the schedulability problem as a binary decision process (go/no go). The first issue is the possibility of validating the selection of the hardware platform, the resource management policies and the software implementation only late, at testing time. The second problem is related to the assumption that all control problems are of type hard real-time. In reality, several systems may miss deadlines without losing stability, and indeed, several systems (including fuel injection [1]) actually operate in spite of deadline misses, at the boundary of overload conditions.

An evaluation of the impact of computation (scheduling) and communication delays on the performance of controls in a virtual environment (at design time) requires a toolset that allows to model the controlled system or plant, the controller logic, and the computation and communication resources, together with the task and message implementation of the controller functionality. Such tools are not readily available from the market, even if several academic projects provide solutions to some or even most of the problems in this context.

A practical solution to the problem cannot ignore the use of commercial standards, such as, for example, the MATLAB/Simulink toolset and the Modelica standard, or the OMG Model-Driven Architecture (MDA). Simulink allows the modeling of continuous-, discrete-time, and hybrid systems, and allows to verify the system functionality against a dynamic model of the controlled system (plant). However, it lacks the capability of modeling physical computing architectures (and to some degree tasks and resources), as well as computation and communication delays that depend on the platform. Model-Driven Engineering (MDE) and Architecture Description Language (ADL) are very good at representing architectural aspects and are designed for extensibility. Also, they typically provide mechanisms to transform models expressed in a language into another. MDE and ADL languages may support the modeling of the execution platform [2], but the tools supporting these languages seldom allow for simulation and the automatic generation of the behavioral code.

The analysis of computation and communication delays can be performed using the Truetime blockset in Simulink. TrueTime [3] is a freeware Matlab/Simulink-based simulation tool that has been developed at Lund University since 1999. It provides models of multi-tasking real-time kernels and networks that can be used in simulation models for networked embedded control systems. TrueTime is used by many research groups worldwide to study the (simulated) impact of lateness and deadline misses on controls. The TrueTime *Kernel* block simulates a *computer node* with a generic real-time kernel, A/D and D/A converters, external interrupt inputs and network interfaces. The *Network* block simulates network scheduling. Kernel and Network blocks are configured by an initialization script (usually written in Matlab code), where a specific API is used by the designer to create tasks, messages, timers and interrupt handlers and define the scheduling and network management policies and the communication resources.

In TrueTime, the model of task code is represented by *code functions* that are written in either Matlab or

C++ code. Data connections among Simulink models are implemented in code using a purposely offered API and the application of a TrueTime Scheduler to an already existing Simulink model of controls requires substantial rewriting, mixing the controller functionality, and models of the task and message set, the scheduler, and the physical execution platform. Because of the monolithic architecture and the number of code artifacts that are needed for system configuration (e.g., initialization script and code functions), the current TrueTime implementation is hardly compatible with an automatic model generation and a M2M transformation flow.

In this paper, we provide a short introduction to a project for the realization of an open toolset, based on the Simulink platform and several open source projects and modeling environments, for the modeling of execution platforms, task and message implementations in SysML/MARTE (Modeling and Analysis of Real-Time and Embedded Systems [4]). The framework allows the definition of the mapping of the Simulink model into the execution platform components and the definition of the task and message model. The project includes the development of Simulink custom blocks for a modular representation of tasks, message and schedulers, and a framework that allows to co-simulate real-time scheduling together with the hybrid Simulink models of the controller and the plant. Finally, we are implementing a set of transformation rules to automatically generate an extended Simulink with blocks representing the execution of the Simulink controller model into a set of real time tasks, with their execution times, under the control of a scheduler, and the exchange of messages over a network. The modularity of the framework allows to easily build a set of examples by simple extensions to existing Simulink models, with only incremental modifications.

The project is entirely open source. A web site (http://retis.sssup.it/tres) is dedicated to it. The current development stage is the following. A full implementation for the single and multicore scheduling part is available. Support for networking is still work in progress (in the design stage). Examples of applications ar available on the web site, including a model for the control of three electrical motors (adapted from the TrueTime examples), and a model of a quadcopter (from [5]). We describe here the main architectural concept of the system as a whole, integrating the description of the parts that are already completed with the design concepts that apply to the part under development.

State of the Art

In the model of complex (cyber-physical) systems, separation of the functional and platform models is advocated by many. The OMG (a standardization organization) in its Model-Driven Architecture (MDA) [6] defines a three stage process in which a Platform-Independent Model or PIM is transformed in a Platform-Specific Model or PSM by means of a Platform Definition Model (PDM). Finally, the automotive industry AUTOSAR standard [7] defines a virtual integration environment for platform-independent software components and a separate model for the (distributed) execution architecture, later merged in a deployment stage (supported by tools). The TIMMO/TIMMO2 [8] projects focus on the modeling infrastructure and the capability of modeling timed events in AUTOSAR. Unfortunately, AUTOSAR does not have a formal model for the behavior of the functions and especially the dynamics of the plant. Therefore, an external tool or the actual code is needed for functional modeling and simulation. Raghav et al. [9] and Hugues et al. [10] proposed two methods for describing the functional behavior according to a reference architecture and then comparing the deployed system with respect to the reference to check whether the performance (delay) target is guaranteed. Multi-paradigm modeling deals with the challenges in the integration of heterogeneous models of computation representing hardware and software. Examples are [11] and [12]. In our work, we focus on the integration of schedulability analysis tools and the commercial Simulink platform.

The development of a platform model for (large and distributed) embedded systems and the modeling of concurrent systems with resource managers (schedulers) requires domain-specific concepts. The OMG MARTE standard is general, rooted on UML/SysML and supported by several tools. MARTE has been applied to several use cases, including on automotive projects [13]. GeneAuto [14], ProjectP [15], the Rubus Component Model [16] and AADL [17] put emphasis on the modeling of task sets and their interactions and the code generation infrastructure, without including simulation capabilities or an explicit formal metamodel for the internal behavior of tasks.

A very large number of projects target the evaluation of scheduling policies and the analysis of task implementations (more than 6 million hits when searching the keywords *real time scheduling simulator* in Google). A necessarily incomplete list includes Yartiss [18], Storm [19], ARTISST [20], Cheddar [21], Stress [22]. Finally, several research works investigate the consequences of computation (scheduling) and communication delays on controls from the standpoint of control theory. An overview on the subject can be found in [23]. Recent works on this subject include [24].

II. Methodology and Flow

The development flow considered in our work is summarized in Figure 1. The *Simulink functional model* is the starting point. Once the simulation results are satisfactory, the designer uses the model exporter to generate an abstract view of functional model. The abstract view is an exported XML file that conforms to an Ecore meta-model for SR systems. The Ecore view preserves all the structural properties of the Simulink model, such as the types and interfaces of the blocks and the connections among the blocks, and also accounts for the information related to the timed execution events, including rate and partial order of execution constraints. Next, the Ecore representation of the Simulink model is translated using QVT into a SysML model in Papyrus (leveraging a profile definition). Here, it is extended with the platform and mapping models.

The *Platform model*, as well as the model of the tasks and messages, is generated using SysML [25], [26]. A specialized profile, built on top of the OMG standard

MARTE (Modeling and Analysis of Real-Time and Embedded systems) [4] profile, is used for modeling embedded platforms and systems, including multicore computation nodes, networks, scheduling and resource management policies on nodes, and arbitration policies for message transmission on networks.

The *mapping model* associates functional elements to tasks, tasks to processing (HW) elements, signals to messages and messages to networks. The mapping model is defined in SysML, by leveraging and extending the standard concept of Allocation. When the SysML model of the functionality mapped onto the platform is complete, Matlab code is generated from the SysML model using the Acceleo [27] open model-to-text generator. The generated code operates on the original Simulink model and adds to it a set of custom blocks (with connections), representing the implementation of the Simulink subsystems of the controller in tasks, executing under the control of a scheduler.

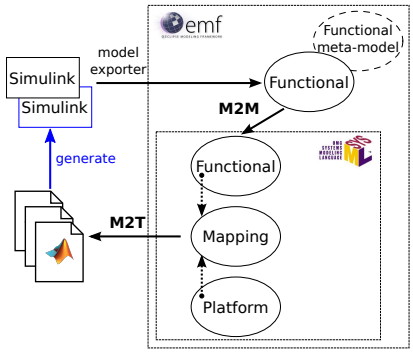

Figure 1. The development flow for the proposed approach.

III. Architecture Overview

The architecture of the co-simulation environment for the evaluation of the impact of scheduling and communication delays on the performance of controls is summarized in Figure 2.

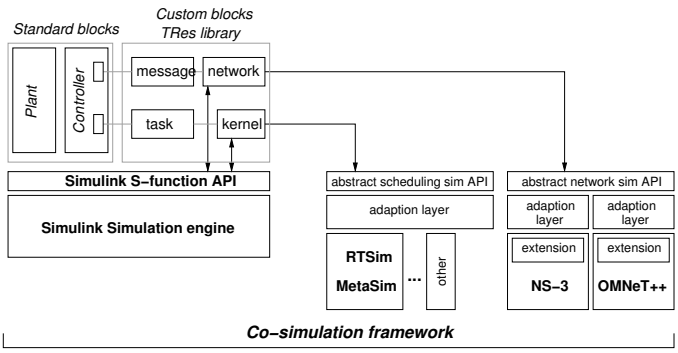

Figure 2. The system co-simulation of the plant and functional controls with the task and network scheduling parts.

The master simulation engine is Simulink. At simulation time, the Simulink engine computes the model update in an outer loop, in which *major steps* are evaluated. A major step is a point in time in which the inputs and outputs of the model blocks are computed and updated. Inside each major step, an inner loop on *minor steps* is used to update the continuous parts of the model.

Our real-time scheduling simulator is implemented as a set of custom blocks that execute at all major steps and interact with the Simulink main engine (capturing the relevant events from the simulated environment).

Every time a major step occurs, the blocks implementing the kernels and networks are invoked and process (if there is any) the task and message arrival events and any other event that is active at the same time. These events are forwarded to the real-time and network scheduling simulators, respectively, and cause an update of their internal structures. The kernel and network Simulink blocks will then query the scheduling and network simulators to determine future relevant events and then use the Simulink API to define major steps in the simulation at all the points in time in which a scheduling event (for one of the system tasks or messages) occurs.

In our project, the real-time scheduling and the network simulator engines are not implemented directly in the custom kernel and network blocks (as in TrueTime). The scheduling simulator is accessed through an abstract interface that mediates between the code of the custom kernel block and a generic real-time simulator. This abstract interface allows to use any scheduling simulator provided that the user writes an adaption layer that consists of a concrete implementation of three generic classes for scheduler, tasks and events. To provide an example, and allow for self-contained use of the project, such adaption layer has already been written and made available for the open source RTSim project (rtsim.sssup.it). RTSim supports multi-core architectures with global scheduling policies. A similar layer will abstract the network simulation engine and allow for the reuse of existing network simulators. Currently, we are defining the abstract interfaces (and adaptation layers) that allow for the (re)use of the very well known projects OMNet++ and NS-3.

For the execution and scheduling of tasks, our framework assumes the same model as in TrueTime (which is also suited to the typical code generation process for Simulink models). The execution of a task is split in units called *segments*, informally corresponding to the execution of a function called by the task main code. Each segment is identified by an execution time (possibly according to a given distribution) and all segments in a task are executed according to a pre-defined sequence. The time duration of each segment corresponds to the execution time of the code implementing one subsystem in the Simulink model.

In more detail, the real-time task execution of Simulink models on single- and multi-core platforms is realized through two custom blocks: `Kernel` and `Task` (shown in Figure 3). The interactions between the Simulink simulation engine and our custom blocks occur through the standard set of Simulink API functions that allow to set inputs and outputs and force a simulation event.

The block `Kernel` models a real-time kernel and the scheduler inside it on a single- or multi-core node according to a given scheduling policy. Each task is modeled with one instance of the block `Task` and consists of the serialized execution of segments/subsystems.

Each block `Task` is a triggered subsystem, executed

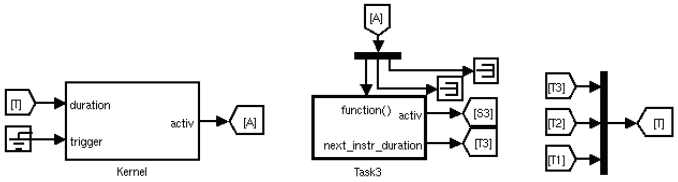

Figure 3. `Kernel` and `Task` blocks.

on the occurrence of a function call event received on its port `function()`. Its output interface consists of two ports: `activ` and `next_instr_duration`. The first one is an array of function call used to issues activation and termination events to the Simulink subsystems executed in the the task segments. The second port outputs a scalar signal representing the duration of next segment executed by the task. Each time `Task` is triggered, it issues the termination signal for the previously executed segment (if any), outputs the activation signal for the current segment, and transmits the execution time of the new segment to the block `Kernel`. The duration of segments executed by `Task` is set by a variable in the Matlab workspace.

The block `Kernel` has two input ports: `duration` and `trigger`. On the `duration` port receives an array of values, one for each `Task` block, with the indication of the duration of the next segment to be executed. On the second port, it receives the array of activations signals of aperiodic tasks (from external sources). The block has one output port, named `activ`, which is used to signal to each task the execution of the current segment. The (simulated) kernel is characterized by the scheduling policy (Deadline Monotonic - DM, Fixed-Priority - FP, and Earliest Deadline First - EDF), with its deadline miss recovery option, and the number of cores it manages.

The start and completion times of the task segments correspond to the times in which the corresponding subsystems reads or sample their inputs and produce their outputs. To guarantee this execution semantics, the activation of the (formerly periodic) subsystem blocks defining the control laws are changed from periodic to function activated (Figure 4) and a latch barrier is added on all their outputs. The signals activating the subsystem (and its input sampling) and the output latch are generated by the task blocks upon the beginning of the execution and the completion of the corresponding segment.

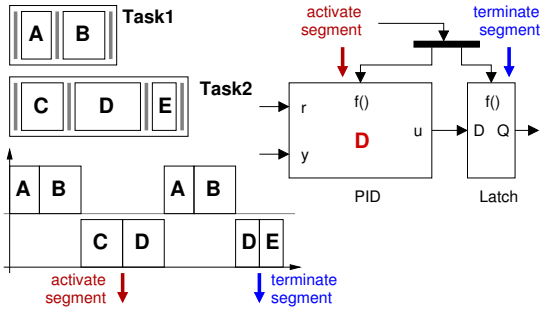

Figure 4. The execution of subsystems modeled through segments.

Similar to tasks, the interaction with the network scheduling simulation will be realized (this part is currently under design and development) using a network block for each network, and a Message block for each message in the system (Figure 5). Similar to the Kernel block, the Network block receives a specification of all the periodic messages queued in the system. It forwards the message arrival event to the Network simulator and queries the network simulator for the start of transmission and end of transmission events of the messages. When a start of message transmission event is active, the Network block triggers the corresponding Message block for sampling the signal values that are transmitted with the message. When the message arrives, another signal is sent to the Message to indicate that the signal values are now available at their destination and ready to be used by the reading subsystems. The availability of the signal values at destination is indicated by the message block by ativating a set of latches in correspondence to all the signal values that are mapped onto the message.

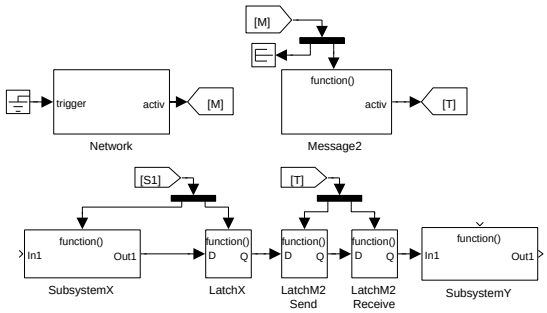

Figure 5. The transmisison of information signals packed into messages under the control of the network block.

IV. Modeling and Generation Overview

The ***functional model*** is created by importing in EMF a Simulink model that includes the controller part and the model of the plant. The Simulink model must comply with the restriction that there is a decomposition level in which the controller part consists of a collection of subsystems, in which each subsystem only contains periodic blocks with the same period (each subsystem has a single rate). Also, subsystems must have a unique name that is used as a reference in the following backannotation stage. A Matlab script uses the Simulink modeling API (programming interface) to parse the model structure and export an XML view of the controller subsystems. The XML conforms to a schema created in accordance with an Eclipse Ecore metamodel, defined for representing the execution constraints that apply to the Simulink subsystems, not too dissimilar from the one proposed in the GeneAuto project [14]. Also, custom functionality (for example in C or C++ code can be wrapped in a custom Simulink wrapper using the S-function mechanisms and included in the simulation.

For the modeling of the physical (HW) part of the ***execution platform*** we rely on the concepts provided in MARTE, and we define our own taxonomy of stereotypes for Basic Software (BSW) components and for the deployment of BSW modules onto the HW. The execution-platform meta-model concepts are organized in two packages: `HwResources` and `BswRTOS`.

The `HwResources` package introduces an element representing a HW board (`HwBoard`) and other HW mod-

eling entities. The `HwProcessor` stereotype provided in the `HW_Computing` sub-package matches the concept of CPU and provides the attribute `nbCores` to specify the number of cores, thus enabling the modeling of multi-core architectures. Other stereotypes are under definition for the representation of networks and network adapters with their attributes.

The ***mapping model*** represents the execution of functional elements by tasks and the allocation of tasks on cores. The two concepts that are central to the definition of a mapping model are: *threads*, represented by the stereotype `Thread`, and *signal variables*, denoted as `ComImpl`. A `Thread` is a unit of concurrent execution that runs on one of the system cores under the control of an RTOS. Each `Thread` is contained in a `Process` and is characterized by a `priority` value. Concrete specializations of `Thread` are `AperiodicThread` and `PeriodicThread` (with its `period`).

The concept of *allocation* completes the specification of mapping meta-model. The `FunctionToThreadMap` denotes the mapping of a functional subsystem into a `Thread`. When multiple subsystems are mapped into the same `Thread`, the attribute `mapOrder` defines how the execution of their `step()` methods will be serialized in the generated thread code. The *mapping order* must be consistent with the partial order of execution imposed by the Simulink model semantics (currently, the designer is responsible for checking compliance. We are investigating the use of OCL constraints for this purpose). The `ThreadToCPUMap` models the deployment of a `Thread` to an `HwProcessor`. The attribute `coreAfn` enables the binding of the thread to a physical processor core (processor affinity).

A set of Acceleo Model-to-text ***generation templates*** processes the SysML model and generates automatically the Simulink blocks for the task and scheduler implementation. The current set of scripts (and the Simulink custom blocks) handle the case of single-core and multicore execution under global scheduling policies. The Acceleo scripts are invoked from a common main template that performs the following sequence of operations: 1) The Simulink custom library of tasks and scheduler blocks is opened. 2) The functional model is saved and a new model is created for its backannotated version. 3) A Matlab script is generated, that creates the initialization variables for the kernel and the task attributes. 4) Another Matlab script is generated for the generation of the kernel and the task blocks. 5) Finally, another set of .m files is created to modify the input model by changing the subsystem blocks to triggered, adding latches on the output links and rerouting the connections

V. Examples and Project availability

The project files are in part already available on a dedicated web site as open source. The part that is available is the code implementing the Simulink custom blocks for the interaction with the node scheduling simulator. The SysML profiles and the Acceleo tranformations will follow shortly. The project is meant to be distributed with a very large possible audience, that includes real-time researchers and students, control developers and embedded systems designers. It also impacts the modeling of embedded systems. Therefore, it could be of interest for SysML modelers and MDE practitioners, and especially, useful as a teaching tool for a large set of disciplines, including simulation, control theory, real-time scheduling, modeling, model-to-model and model-to-text transformations. Together with the custom block code, some examples have already been adapted or ported to our framework to show the applicability of the blockset, to assist in the demonstration of the tool capability and to provide an initial nucleus for use in courses. The experience in developing these initial examples shows that adaption of an existing Simulink model takes only a few hours. We are therefore confident that this initial set will grow very quickly.

The initial examples include a three-servo example (from TrueTime) and a quadcopter model. The three servo Simulink example includes three PID controllers mapped for execution onto three tasks, scheduled on a single-core platform. The result of the mapping is the structure of tasks in Figure 6. The model has been automatically generated starting from an initial model of the controls (without tasks and scheduler) and processing a SysML model of the implementation with Acceleo M2T transformation scripts. The model has one kernel block (`Kernel1`) and three instances of task blocks with three latches and the connections. Task blocks manage the activation and termination signals of the PID subsystems/segments.

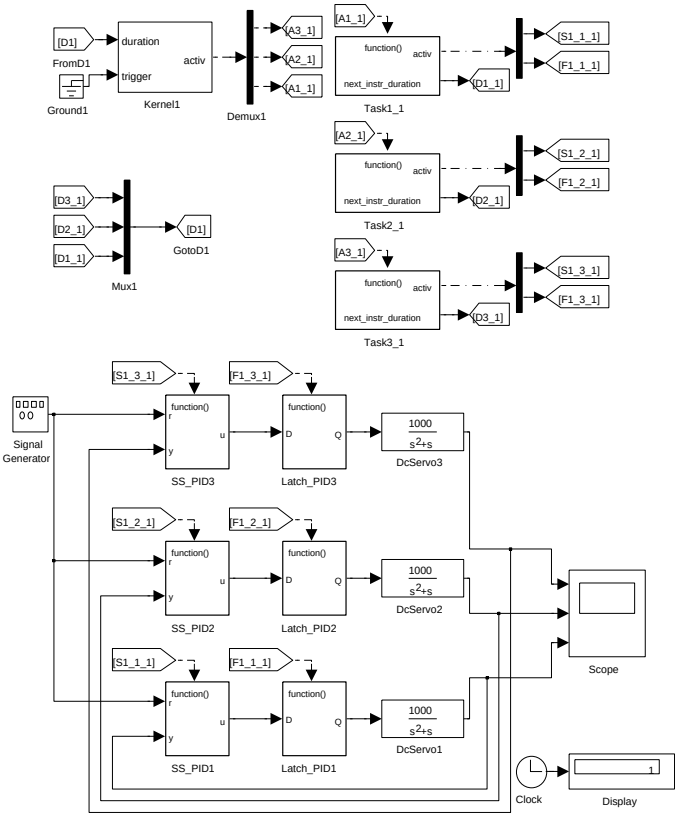

Figure 6. The three servo example.

Figure 7 shows the output of the DC-servos with respect to the reference signal. when a Rate Monotonic (RM, on the left) or EDF scheduling policy (on the right) is used. In both cases, task `Task1_1` (on top) has the lowest priority. In

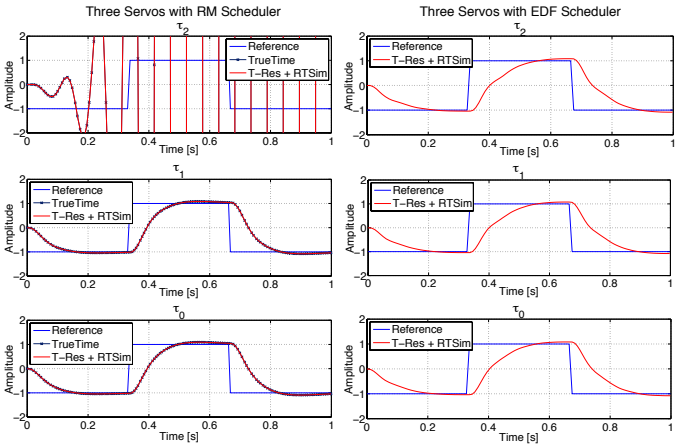

Figure 7. Verification by simulation on the back-annotated model.

this example, the CPU utilization factor is $U \simeq 1.23$. This overload condition induces some performance degradation of controls with respect to the simulation results obtained from the Simulink model without back-annotations. This difference is clear in the results on the left and right side of Figure 7. In the case of RM, the task with the lowest priority cannot guarantee a stable control, because of too many deadline misses. In the case of EDF, the delay due to scheduling and tasks' execution times tends to be spread among the three tasks, and after an initial transient all tasks miss their deadlines. However, the motion of the DC-servos is still controlled with a reasonable error, and the overall control performance is still satisfactory.

VI. Conclusions and Future Work

We present a framework for the definition of execution platforms and task implementations of Simulink functional models. The platform makes use of a purposely developed co-simulation environment for real-time scheduling and network communication, and allows to obtain Simulink models that evaluate the impact of computation and scheduling delays on the performance of the controls. The (open source) project is currently being extended with the modeling of networks, messages and communication delays.

References

[1] D. Buttle, "Real-time in the prime-time," *Keynote presentation, Euromicro ECRTS Conference 2012*, July 2012.

[2] EAST Architecture Description Language (ADL). [Online]. Available: http://www.east-adl.info/

[3] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. E. Årzén, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Syst. Mag.*, vol. 23, no. 3, pp. 16–30, June 2003.

[4] Object Management Group. (2011) A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems. OMG Document Number formal/2011-06-02. [Online]. Available: http://www.omg.org/spec/MARTE

[5] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[6] Object Management Group. Model Driven Architecture (MDA). [Online]. Available: http://www.omg.org/mda/specs.htm

[7] AUTomotive Open System ARchitecture (AUTOSAR). Specifications 4.0. [Online]. Available: http://www.autosar.org/

[8] TIMing MOdel – TOols, algorithms, languages, methodology, and USE cases (TIMMO-2-USE). [Online]. Available: https://itea3.org/project/timmo-2-use.html

[9] G. Raghav, S. Gopalswamy, K. Radhakrishnan, J. Delange, and J. Hugues, "Model Based Code Generation for Distributed Embedded Systems," in *Proceedings of the European Congress on Embedded Real Time Software and Systems, ERTSS*, 2010.

[10] J. Hugues, B. Zalila, L. Pautet, and F. Kordon, "From the prototype to the final embedded system using the Ocarina AADL tool suite," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 4, pp. 42:1–42:25, Aug. 2008.

[11] E. S. A. Canedo and M. A. A. Faruque, "Context-sensitive synthesis of executable functional models of cyber-physical systems," in *4th International Conference on Cyber-Physical Systems (ICCPS '13)*, 2013.

[12] H. Vangheluwe and J. de Lara, "Foundations of multi-paradigm modeling and simulation: computer automated multi-paradigm modelling: meta-modelling and graph transformation," in *35th conference on Winter simulation: driving innovation (WSC '03)*, 2003.

[13] E. Wozniak, C. Mraidha, S. Gerard, and F. Terrier, "A Guidance Framework for the Generation of Implementation Models in the Automotive Domain," in *Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA*, 2011, pp. 468–476.

[14] Automatic Software Generation for Real-Time Embedded Systems (Gene-Auto). [Online]. Available: http://gforge.enseeiht.fr/projects/geneauto

[15] Project P. [Online]. Available: http://www.open-do.org/projects/p/

[16] K. Hänninen, J. Mäki-Turja, M. Nolin, M. Lindberg, J. Lundbäck, and K.-L. Lundbäck, "The Rubus Component Model for Resource Constrained Real-Time Systems," in *Proceedings of the IEEE International Symposium on Industrial Embedded Systems, SIES*, 2008.

[17] Architecture Analysis & Design Language (AADL). [Online]. Available: http://standards.sae.org/as5506b/

[18] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, and M. Qamhieh, "YARTISS: A Tool to Visualize, Test, Compare and Evaluate Real-Time Scheduling Algorithms," in *Proceedings of Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems, WATERS*, 2012, pp. 21–26.

[19] R. Urunuela, A.-M. Déplanche, and Y. Trinquet, "STORM a Simulation TOol for Real-time Multiprocessor scheduling evaluation," in *Proceedings of IEEE Int. Conference on Emerging Technologies and Factory Automation, ETFA*, 2010, pp. 1–8.

[20] D. Decotigny and I. Puaut, "ARTISST: An Extensible and Modular Simulation Tool for Real-Time Systems," in *Proceedings of IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing, ISORC*, 2002, pp. 365–372.

[21] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *Proceedings of the ACM International Conference on Ada (SIGAda)*, 2004, pp. 1–8.

[22] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "STRESS: a Simulator for Hard Real-time Systems," *Softw., Pract. Exper.*, vol. 24, no. 6, pp. 543–564, 1994.

[23] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*. Dover Publications, Jan. 2011.

[24] A. Cervin, M. Velasco, P. Martí, and A. Camacho, "Optimal on-line sampling period assignment: Theory and experiments," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 902–910, 2011.

[25] Object Management Group. (2011) The Unified Modeling Language (UML). [Online]. Available: http://www.omg.org/spec/UML/2.4.1/

[26] ——. (2012) The Systems Modeling Language (SysML). [Online]. Available: http://www.omg.org/spec/SysML/1.3/

[27] Obeo. Acceleo. [Online]. Available: http://www.eclipse.org/acceleo/

Towards a model-based approach guiding the scheduling analysis of real-time systems design

Yassine Ouhammou, Emmanuel Grolleau, Michael Richard, Pascal Richard
ISAE-ENSMA and University of Poitiers
LIAS
Futuroscope, France
Email: {ouhammoy, grolleau, richardm}@ensma.fr and pascal.richard@univ-poitiers.fr

***Abstract*—Real-time systems need to be analyzed at an early stage of the development life-cycle in order to check if all the timing requirements are met. One of the main difficulties that the system designers face is to find the appropriate analysis tests helping to validate and/or to dimension properly their designs. In this paper, we propose an approach to improve the way designers check their system designs. Our approach unifies modeling and schedulability analysis efforts. It has been used with a modeling language in order to orient designers for choosing the most suitable analysis models and tests. Furthermore, this work is also dedicated to research teams to share their researches and prototypes. Hence, we aim at increasing the usability of the real-time scheduling theory.**

I. Introduction

The life-cycle of hard real-time systems (RTS) can last several years, where the system developers have to cope with new hardware/software/standards. To reduce the time-to-market, the design phase of RTS is treated as one of the earliest phase where the validation of timing properties can reduce the cost of the development process. Nevertheless, expert designers in both design and analysis of RTS are uncommon.

Model-driven engineering (MDE) becomes increasingly used to propose solutions and tools for modeling and analyzing RTS. Recently, a set of standard design languages has been proposed for RTS (e.g. UML-MARTE[1], AADL[2]), by modeling different systems artifacts and non-functional requirements so as to support different analysis kinds. The implementation of the schedulability analysis techniques has also taken advantage of MDE. In the past few years, several academic and industrial tools have been proposed as providers of the well-known analysis techniques (validation and dimensioning). They offer the possibility to apply some subsets of schedulability tests during the analysis stage. Some examples of those tools are: Rt-Druid [1], MAST [2], ASIIST [3], SymTA/S [4] and Cheddar [5]. These tools allow real-time designers, especially modelers, to operate during the analysis phase. Since the analysis tools are based on different input analysis models, they facilitate the utilization of the schedulability analysis tests.

However, the analysis tools, as they are used currently, are also driven by the real-time designer's experience. Therefore, determining what type of the analysis technique or the analysis model to use for a given analytical situation may be difficult. Then, extracting the relevant information to define the analysis model may be laborious. Indeed, the following motivating example shows that yet the utilization of the current solutions still requires deep knowledge of the real-time scheduling or the presence of an expert in the schedulability analysis domain.

Motivating example. We consider an application composed of four periodic independent tasks where each task is defined by a set of properties (See Table I). Tasks are preemptive and ordered by decreasing priority. *Task1* is the highest priority task and *Task4* is the lowest priority task. The task-set is executed on a uniprocessor architecture with priority-based scheduling. We assume that the model is done via a design framework and the transformation is done correctly. Although, after launching the analysis process through two different analysis tools which are Rt-Druid [1] and MAST [2], two different results have been provided for the same input model. Table II shows the response-time values provided by MAST and Rt-Druid. While the Rt-Druid result shows that the system is not schedulable because the response-time of *Task4* exceeds the deadline, the Mast result shows that the system is schedulable and provides more accurate response time values (i.e. the result provided by Rt-Druid is pessimistic).

Task	Worst-case execution time	Deadline	Period	Release time
Task1	3 ms	15 ms	20 ms	2 ms
Task2	4 ms	8 ms	23 ms	0 ms
Task3	5 ms	13 ms	23 ms	5 ms
Task4	9 ms	23 ms	23 ms	7 ms

TABLE I. Values of task characteristics

Task	**Worst-case response-time** (MAST)	**Worst-case response-time** (Rt-Druid)
Task1	3 ms	3 ms
Task2	7 ms	7 ms
Task3	8 ms	12 ms
Task4	21 ms	33 ms

TABLE II. Worst-case response times calculated by two analysis tools

The results provided by Rt-Druid are more pessimistic than those provided by MAST on this example.

Explanation. The difference is not related to a wrong implementation of the analysis methods, but to the input analysis model. The example shown in Table I is considered by MAST as a transaction model [6]. Then, the mathematical formula calculating the response-time takes the release-times into consideration. Whereas, the analysis functionality chosen via Rt-Druid masks the release-times and considers the analysis model as a simple periodic model [7]. Then, the mathe-

[1] www.omgmarte.org
[2] www.aadl.info

matical formula calculating the response-times considers that tasks are non-concrete (i.e. the release-times are unknown). Consequently, the tool considers the worst-case behavior of the analysis model by supposing that tasks are simultaneously released. In fact, the result provided by Rt-Druid does not mean that the tool does not support the transaction analysis model, but the analysis functionality chosen via the tool does not provide adequate analysis. In some cases, choosing a very abstract model is not a wrong choice. For instance, the analysis functionality chosen via Rt-Druid (i.e. the analysis test) offers the possibility to use a sensitivity analysis like the test proposed in [8] (which is a dimensioning technique) enabling to tune the system design. To the best of our knowledge, no similar analysis test (i.e. similar to the sensitivity analysis tackled in [8]) exists for the transaction model.

This research proposes a novel strategy leading to get several analysis repositories playing the role of "decision supports". This latter helps designers during the analysis phase in order (i) to detect the analysis situation corresponding to the system design and (ii) to choose the most suitable analysis tests. Thus, an analysis repository model is proposed in order to be instantiated and to customize their instances (i.e. a set of analysis repositories) as needed. Therefore, the consistency and the accuracy of the final system design obtained following the choices proposed by an analysis repository depend on the content of this latter. To enhance the applicability of the real-time scheduling analysis theory, this work can be used as showcase and teaching-aid allowing the research teams to show their results (e.g. analysis models, tests, home tools) and also to share them with other teams of the RTS community.

Following this introduction, Section II presents the relevant elements of our analysis repository approach. Section III is devoted to present the capabilities of the analysis repository and its ability to be used with a modeling design language, a brief proof of concept is also presented. Finally, Section IV summarizes and concludes this article.

II. Analysis Repository Model

This section is devoted to detail our approach in order to unify modeling and analysis efforts. Our idea is to propose a model of analysis repository, which can be instantiated and enriched continuously by analysts. Then, the content of each analysis repository (i.e. instance of the analysis repository model) plays the advising role towards designers by identifying which analysis model(s) and test(s) are fitting the system design.

The analysis repository model is based on a set of notions, which are presented and described in the following subsections. Most of those notions are related to the real-time scheduling theory.

A. Real-Time Context and Identification Rule

The notion of "real-time context" represents the cornerstone of the analysis repository model. In the literature, an analysis model represents the formalization of information extracted from a design model. This information is related to the execution requirements, the tasks behavior, the communication protocols, etc. Every analysis model is characterized by a set of assumptions representing the context χ_i of the analyzed system. Each assumption ϵ_j is related to the software architecture, the timing behavior or the hardware architecture.

Let $\mathcal{X}$ = $\{\chi_1, \chi_2, .., \chi_m\}$ be a set of real-time contexts, and every real-time context χ_i = $\{\epsilon_1, \epsilon_2, .., \epsilon_n\}$. For instance, if χ_i represents a context of independent periodic tasks, with arbitrary deadlines, executed on homogeneous multicore processor, and scheduled by a global fixed-job priority policy [9], then χ_i is characterized by the the following assumptions:

- ϵ_1 = Hardware execution platform is multicore;
- ϵ_2 = Processing cores are homogeneous;
- ϵ_3 = Tasks are independent;
- ϵ_4 = Tasks are periodic;
- ϵ_5 = Tasks are preemptible;
- ϵ_6 = There is no self-suspension;
- ϵ_7 = Priorities are assigned following a fixed-job policy;
- ϵ_8 = Full migration is authorized;
- ϵ_9 = Deadlines are arbitrary.

For factoring the number of assumptions and to guarantee the scalability and the incremental enrichment of the analysis repositories, we suggest that analysis repository model proposes to instantiate a set of identification rules $\mathcal{R}$.

Let $\mathcal{R}$ = $\{r_1, r_2, .., r_n\}$ be a set of identification rules. Every rule r_i represents an interrogation checking the system design and leading to an assumption characterizing one or several real-time contexts. It is common to find several real-time contexts characterized by the same subset of assumptions, or the opposite subset of assumptions. For example, let:
$-$ r_a be a rule checking if "all tasks are preemptible"
$-$ r_b be a rule checking if "all tasks are <u>not</u> preemptible"
We consider the following real-time contexts:
χ_1 supports "only preemptible tasks";
χ_2 supports "only non-preemptible tasks";
χ_3 supports "preemptible and non-preemptible tasks".

If the evaluation of r_a referring to a system design M is equal to:

- True (we note M $\models r_a$), hence M is close to χ_1 and χ_3;
- False (we note M $\models \overline{r_a}$), hence M is close to χ_2 or χ_3;
- Undefined, hence M is close to χ_3. In this case, the "undefined" means that we can not conclude about the satisfaction of the checked characteristic. That can be due to the absence of sufficient elements in the design M.

If the evaluation of r_b referring to a system design M is:

- True, hence M is close to χ_2 or χ_3;
- False, hence M is close to χ_1 or χ_3;
- Undefined, hence M is close to χ_3.

Therefore, we define a function "$Spec$" as a relationship between the set of contexts $\mathcal{X}$ and the set of identification rules $\mathcal{R}$, where:

$Spec$: $\mathcal{X}$ x $\mathcal{R} \rightarrow$ {True, False, Undefined}. In this case, the "Undefined" means that among the rules defining a context, we can find many rules and their opposites.
For instance: $Spec(\chi_1,\ r_a)$ = True, $Spec(\chi_1,\ r_b)$ = False, $Spec(\chi_3,\ r_a)$ = Undefined, $Spec(\chi_3,\ r_b)$ = Undefined.

Meta-modeling. Figure 1 shows the formalization of $\mathcal{X}$ and $\mathcal{R}$ and their relationships by using concepts of UML class diagram. Indeed, every identification rule r_i is an instance of `IdentificationRule` class defined by an `id`, a `description` of the rule in natural language helping users to understand the goal of the rule, and a `formal expression`. This latter depends on the modeling language expressing the design which requires analysis. For instance, 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>. Consequently, the formal expression property of an identification rule aiming to check if the deadlines of all tasks are constrained will be expressed as follows: $\forall\ \tau_i \in Sys,\ T_i \leq D_i$.

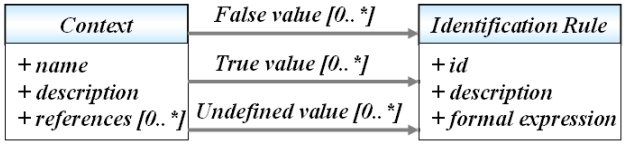

Fig. 1. real-time context and identification rule classes and their relationships

Every real-time context is an instance of `Context` class. This latter is characterized by a unique `name`, an optional `description` explaining the real-time context in native language, and `references` attribute showing research papers which have introduced the real-time context. Each `Context` instance should be specified by a set of identification rules via three relations. $Spec\ (\chi_i, r_j)$ has been translated to three kinds of relations as mentioned in Figure 1.

B. Analysis tests and their characteristics

Scheduling analysis tests have two major purposes: validation (like the response time analysis [10], [7]) and dimensioning (like the sensitivity analysis [8] or the processor allocation).

$\mathcal{T}$ = $\{t_1, t_2, .., t_n\}$ denotes a set of analysis tests contained in the repository.

Every analysis test t_i is related to one or many real-time contexts. When applied to a system, the conclusion provided by the test is correct if all the assumptions of the context are valid for the system. Moreover, validation tests can be considered by a set of characteristics related to the feasibility and the sustainability.

The feasibility characteristic permits to conclude about the accuracy of the test referring to the applicant system. In other words, a schedulability test is defined to be a sufficient condition if all of the task-sets that are deemed schedulable according to the test are in fact schedulable. A test can also be referred as a necessary condition if the failure of the test will indeed lead to miss the deadline at some points during the execution of the system. Schedulability test that is both sufficient and necessary is labeled as exact condition.

In most cases, the parameters of analysis models considers the worst-case values (e.g. worst-case execution time, minimal period, maximal release jitter). During the on-line execution of the system, the task parameters are often better than those considered. Then, the analysis should be sustainable [11] with the new parameter values. So, a schedulability test with respect to a system is sustainable, if the task-sets deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual jobs are changed in any, some, or all of the following ways: (i) decreasing execution times, (ii) increasing periods or inter-arrival times, (iii) decreasing jitters and (iv) increasing relative deadlines. Furthermore, when a change is only related to:

- decreasing the execution-time, then the test ensures the C-sustainability (also known as predictability).
- decreasing the jitter, then the test ensures the J-sustainability.
- increasing the periodicity, then the test ensures the T-sustainability.
- increasing the deadline, then the test ensures the D-sustainability.

Meta-modeling. Every `Context` instance χ_i can be analyzed by a set of the analysis tests (see Figure 2). Every test t_i is an instance of `Analysis Test` class, and it has a set of properties like the `id` the `description`, and the `references` of papers proposing the analysis test. Moreover, feasibility and sustainability characteristics (i.e. instance of `Test Characteristics` class) may vary depending on the context.

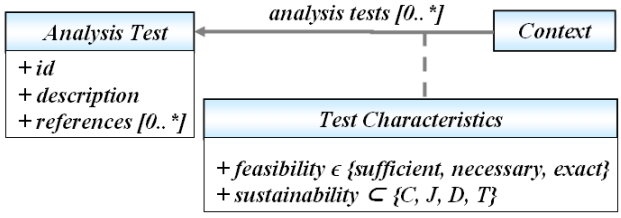

Fig. 2. Analysis test class and its properties

For instance, the mathematical formula of the response time analysis presented in [7] is a validation test representing a sufficient condition for any real-time context corresponding to analysis model with offset (i.e. the release-times are known). However, the same mathematical formula is an exact condition (sufficient and necessary) for contexts corresponding to the periodic analysis model with non-concrete tasks (i.e. the release times are unknown).

C. Analysis tools

As our objective is not to propose a new analysis tool, we aim to take advantages of the existing ones. We give a special attention to classify analysis tools capabilities. By "analysis tool" we do not mean the whole tool like Rt-Druid, but only the analysis functionality provided by the tool. We consider each analysis engine inside an analysis tool as an independent

functionality. Hence, the same analysis functionality inside another analysis framework is considered as another engine. Indeed, a specific analysis test t_i may be related to different analysis tools that are implementing the test. We note $E = \{e_1, e_2, .., e_n\}$ a set of analysis tools inside the analysis repository and which are automatizing various analysis tests.

Meta-modeling. Figure 3 shows the `Analysis Tool` class and its properties, like `id` and `description`. Different analysis tests are implemented by different tools. That allows to compare different output results related to the same test. For this purpose, the `Analysis Tool` class provides the `transformation_model_To_tool` property. This property is devoted to mention the location of an external file (e.g. it can be an executable program or a web service, etc.) permitting the transformation of the system design to the input formalism of the analysis tool. The `transformation_tool_To_model` property is dedicated to contain the location of a program ensuring the restitution after the analysis process (i.e. the reverse transformation).

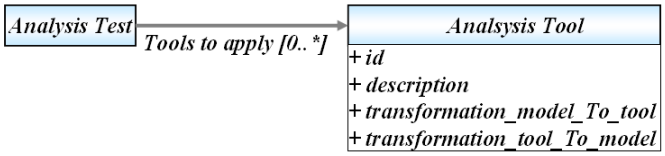

Fig. 3. Analysis tool class and its properties

D. Analysis repository

Every instance of the analysis repository model *Ar* consists of a set of identification rules , a set of real-time contexts $\mathcal{X}$, a set of analysis tests $\mathcal{T}$ and a set of analysis tools *E*. We note, $Ar = <\mathcal{R}, \mathcal{X}, \mathcal{T}, E>$.

Meta-modeling. Every *Ar* is an instance of the `AnalysisRepository` class. It is the root element of other analysis repository elements (contexts, tools, tests, etc.). Then, it is possible to instantiate analysis repositories for every company/organism/research-team to orient their designers and to share their recent results concerning schedulability analysis.

E. Identification process

In order to be manipulable by designers and integrate the design process, the analysis repository model proposes two services. (1) Once the instance of the analysis repository is chosen, the first service loads the system design which needs to be analyzed (e.g. an AADL model or MARTE model). Besides, all the identification rules contained in the chosen analysis repository are evaluated based on the system that needs analysis. (2) The second service loads the results summarizing the evaluation done by the first service, then it compares them with the real-time contexts existing in the analysis repository in order to find the appropriate real-time context if it exists. These two services represent the identification process, which its correctness and its accuracy depend on the content of the chosen analysis repository.

III. Prototyping and integration into a design framework

Since our goal is to assist designers to conclude about the schedulability of their designs, an instance of the analysis repository model has to be invoked by a system design. Consequently, the analysis repository model has to be linked to the design language, which is used to get the system design. Thanks to the `formal expression` attribute of `Identification Rule` class (presented in Section II-A), the analysis repository model can be linked to any design language based on Ecore or UML. In other words, the current implementation enables the analysis repository model to be linked to any design language supporting OCL (Object Constraint Language) [12]. Indeed, the `formal expression` attribute is being the constraint expression formalized depending on both, the OCL notation and the design language used.

In our case, we have chosen to attach the MoSaRT design language (Modeling Oriented Scheduling Analysis of Real-Time systems) to an analysis repository instance. The next subsection gives a brief description of MoSaRT language.

A. MoSaRT design language

MoSaRT language contains several concepts which are very close to the schedulability analysis. It enables to have different kinds of models, each kind focuses on a specific part of the real-time system. Thus, the MoSaRT design language provides the hardware model, the software architecture model, the behavioral model and the functional model. The implementation of MoSaRT design language is based on Ecore [13]. For more details about the MoSaRT language, readers can see the related papers [14], [15].

B. Proof of concept

1) Instantiation of the analysis repository model: First of all we start by illustrating the analyst's tasks. Hence, we provide an analysis repository containing a real-time context corresponding to the Liu and Layland model [16]. This context is based on several identification rules (some of them are shown in Figure 4). Each rule is mapped to a formal expression implemented as an OCL constraint related to the design language of the system that needs analysis (i.e. the MoSaRT design language). Figure 4 shows the formal expression of the identification rule called "`UniprocessorArchitecture`". Moreover, we have chosen the response time analysis test presented in [7] as an analysis test for the context corresponding to the analysis model of [16]. This test is implemented by several tools like Rt-Druid.

In order to ease the use of the analysis tools implementing the analysis tests, we provide two transformation programs. The first one is based on ATL and transforms the system design from MoSaRT language to Rt-Druid formalism. The second transformation program transforms the system design from MoSaRT language to MAST formalism. Since MAST is based on a textual format, the transformation is based on Acceleo[3].

[3] www.acceleo.org

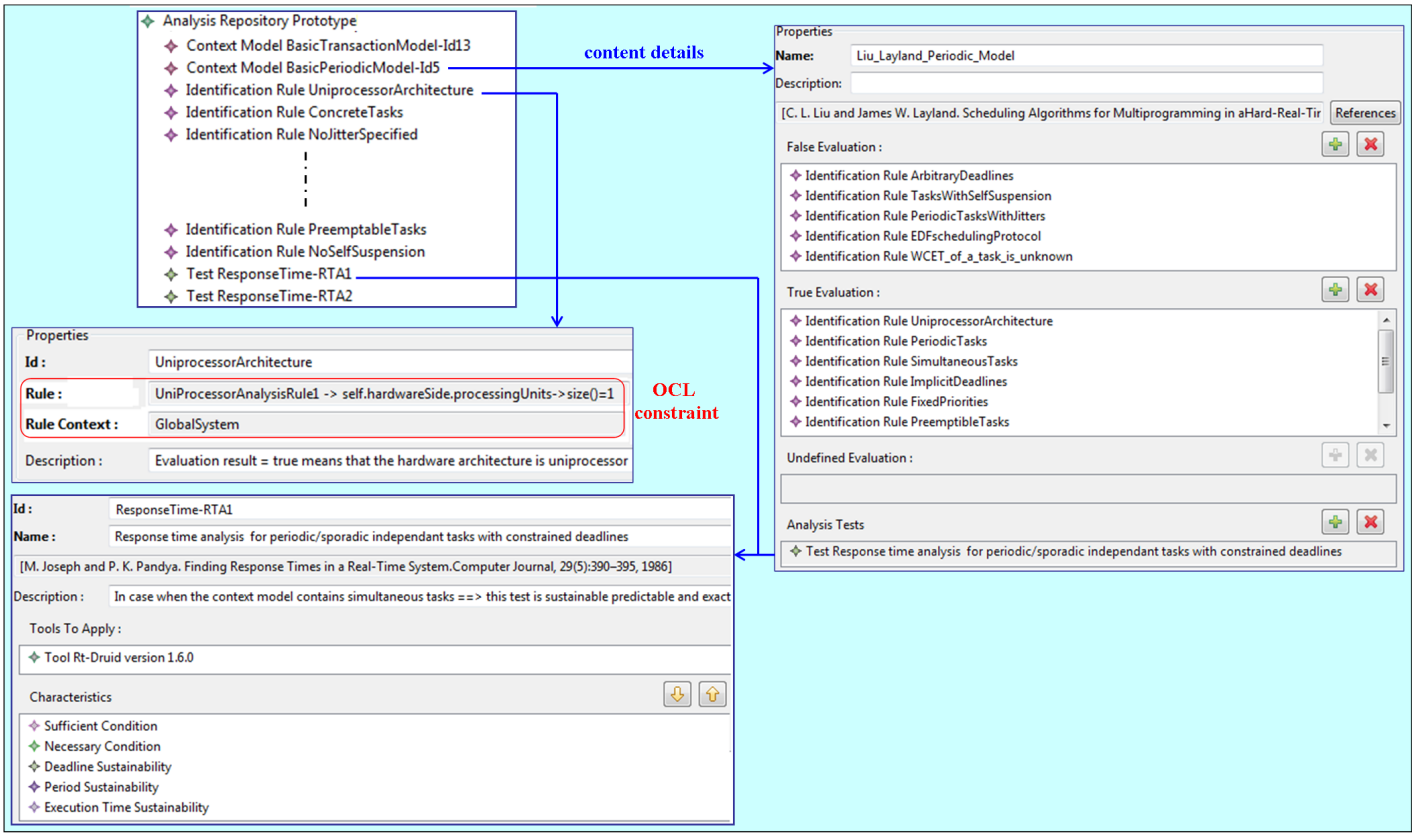

Fig. 4. Part of an analysis repository highlighting the details of a context

2) Utilization of the MoSaRT language for modeling: Figure 5 shows the design corresponding to the motivating example (see also Table I). Parts (A) and (B) represent the architecture layer of the system (software architecture mapped to the hardware architecture). Part (C) shows the software behavioral model of the system, where every task of the software architecture is represented by a `taskActivity` element. The software behavior model shows also the trigger of each task and its timing properties.

3) Utilization of the instantiated analysis repository and its impact: Once the system design is done and becomes ready for analysis, we can select the analysis repository previously instantiated. Figure 6 shows the result that is found after calling the identification process. Indeed, the basic periodic model proposed in [16] is the real-time context which corresponds to the treated example. The result window (See Figure 6) contains also the "Context Characteristics" tab which summarizes the assumptions related to the found context, and the "Analysis Test" tab for orienting designers to the appropriate tools. Furthermore, Figure 6 shows that the design can be analyzed by the response time analysis (`rta1`), and the Rt-Druid tool is one of the tools implementing this test. So, we can generate a model respecting the Rt-Druid formalism after launching the transformation. **Note that the obtained result depends on the content's consistency and the richness of the analysis repository.**

The right column of Table II evinces the analysis result provided by Rt-Druid. The result means that the system is not schedulable and it is pessimistic because the test provided

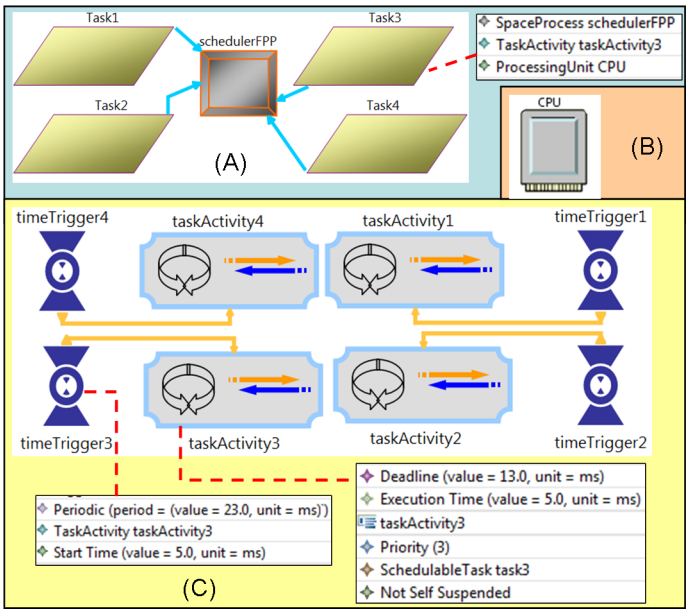

Fig. 5. Different views of a design model that corresponds to the example of Table I. (A): Software architecture model. (B): Hardware architecture model. (C): Software behavior model.

by the analysis repository is only sufficient. In this case, a designer can try to change the task set characteristics or to modify the hardware architecture by using a faster CPU or a multicore processor for example. Nevertheless, this design

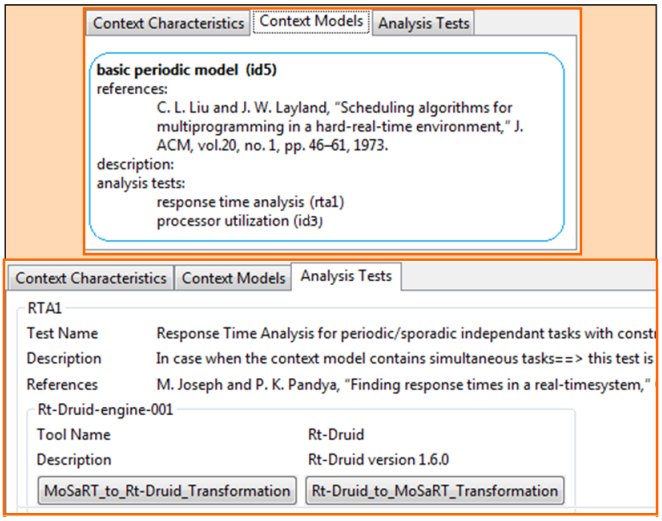

Fig. 6. screenshot of the identification result provided by the analysis repository

model can be analyzed differently in order to avoid the over-dimensioning. Thanks to the identification process's result, the designer should be informed that the design model can be considered as a transaction model [6]. Indeed, the transaction model takes tasks with different offsets into consideration. Consequently, the identification process shall propose the response-time analysis test (`rta2`), which is more appropriate to analyze the transaction model. The test is provided by several tools, like MAST, which was used for the analysis of the treated system, and led to the results illustrated in the left column of Table II.

IV. Conclusion and Perspectives

In this paper, we have presented the MoSaRT analysis repository as a helpful modeling support avoiding wrong design choices at an early design phase. Thus, it helps designers to cope with the scheduling analysis difficulties and to be more autonomous during the analysis stage.

The presented contributions are also dedicated to academic researchers in order to increase the usage of the real-time scheduling theory. So, our contribution can be used as a teaching-aid and also as a way of transferring the scheduling theory results from academia to industrial practice by making it more accessible to industrial users, and also to be compared with other researches.

We are working to improve the utilization of MoSaRT analysis repository by adding relationships between "real-time contexts" and also relationships between "identifications rules". For example, the "generalization" between real-time contexts is one of the relationships that may be taken into account.

We are also working on a special use of the analysis repository, that is when designers know the context of their system, hence they can set their choice at the beginning of the design. In this case the utilization of the analysis repository seems like a design pattern. Then, at each design change (e.g. adding tasks, removing processor, etc.) different analyses will be recalculated interactively at run-time to provide results within the modeling.

In this paper, we have used the analysis repository model with the MoSaRT design language, but it can also be easily used with MARTE or AADL. The use of the analysis repository with these languages may help users to set their own methodologies related to the standard language (like Optimum, a methodology related to MARTE [17]).

References

[1] P. Gai, M. D. Natale, N. Serreli, L. Palopoli, and A. Ferrari, "Adding timing analysis to functional design to predict implementation errors," SAE Technical Paper 2007-01-1272, 2007.

[2] J. L. Medina, J. L. M. Pasaje, M. G. Harbour, and J. M. Drake, "Mast real-time view: A graphic uml tool for modeling object-oriented real-time systems," in *IEEE Real-Time Systems Symposium (RTSS)*, 2001, pp. 245–256.

[3] M.-Y. Nam, R. Pellizzoni, L. Sha, and R. M. Bradford, "Asiist: Application specific i/o integration support tool for real-time bus architecture designs," in *International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE Computer Society, 2009, pp. 11–22.

[4] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis–the symta/s approach," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005.

[5] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *ACM SIGAda*, 2004, pp. 1–8.

[6] J. C. Palencia and M. González Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *IEEE Real-Time Systems Symposium (RTSS)*, 1998, pp. 26–37.

[7] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[8] E. Bini, M. Di Natale, and G. Buttazzo, "Sensitivity analysis for fixed-priority real-time systems," *Real-Time Systems*, vol. 39, pp. 5–30, 2008.

[9] S. K. Baruah and T. P. Baker, "Global EDF schedulability analysis of arbitrary sporadic task systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2008, pp. 3–12.

[10] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

[11] A. Burns and S. K. Baruah, "Sustainability in real-time scheduling," *Journal of Computing Science and Engineering (JCSE)*, vol. 2, no. 1, pp. 74–97, 2008.

[12] OMG, "Object constraint language, omg available specification, version 2.0," www.omg.org/spec/OCL/2.0/, 2006.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.

[14] Y. Ouhammou, E. Grolleau, M. Richard, and P. Richard, "Model driven timing analysis for real-time systems," in *IEEE International Conference on Embedded Software and Systems (ICESS)*, 2012, pp. 1458–1465.

[15] ——, "Towards a simple meta-model for complex real-time and embedded systems," in *International Conference on Model and Data Engineering (MEDI)*. Springer LNCS, 2011, pp. 226–236.

[16] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[17] C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard, "Optimum: a marte-based methodology for schedulability analysis at early design stages," *ACM SIGSOFT Software Engineering Notes*, vol. 36, pp. 1–8, 2011.

Model-driven Deployment Optimization for Multicore Embedded Real-time Systems: the *OptimAll* Approach

Federico Ciccozzi, Juraj Feljan
School of Innovation Design and Engineering (IDT)
Mälardalen University
Västeras, Sweden
E-mail: federico.ciccozzi@mdh.se, juraj.feljan@mdh.se

***Abstract*—The power of modern embedded systems is continuously increasing together with their complexity, thereby making their development more challenging. In the specific case of the adoption of multicore solutions, while processing power is heavily increased, the issue of allocating software tasks to specific cores on the target platform arises. In this paper we introduce *OptimAll*, an automated model-driven approach that aims at providing support in the delicate phase of task allocation at design time. Besides introducing the entire approach, in this work we focus on the automatic generation of a suitable input to the task allocation optimization mechanism from a UML–MARTE system design model, as well as on the actual optimization mechanism and its outcomes in relation to the design model elements.**

I. Introduction

Nowadays most computer systems that connote our everyday life are embedded and characterised by real-time properties. One of the main traits that affects the development of this kind of systems is their ever-increasing performance demand, as they include more and more complex functionality. In most cases, the higher performance needs are tackled by increasing the processing power through, e.g., the adoption of multicore and manycore solutions; in this work we focus on the former. A multicore processor is a single chip that contains two or more processing units that are tightly coupled together in order to preserve energy-efficiency.

Adopting a solution with multiple processing units introduces the challenge of how to deploy software components to the available cores to best utilize the hardware platform. In *OptimAll* we address deployment of software components as a two-step phase: (i) allocation of software components to software schedulable entities (i.e., tasks) and (ii) allocation of these entities to specific cores. In this work we focus on the latter step, namely the allocation of tasks to processing units, with respect to extra-functional properties (EFPs) relevant for real-time systems.

In our solution, we exploit Model-Driven Engineering (MDE) [1] for providing automation and aiding the developer in the delicate phase of deployment optimization at design time. One of the main goals of MDE is to shift the focus of the development from hand-written code to models that represent an abstraction of the problem at hand, and from which early analysis, simulation and testing are made possible through the exploitation of model transformations [2].

More specifically, we exploit MDE for modelling the system under development and in particular the details related to the task allocation problem. Model transformations are defined in order to automatically generate a suitable representation of the task allocation, which can be simulated in order to obtain performance predictions. This in turn enables task allocation optimization — we asses a number of task allocation candidates in search for one that satisfies the constraints on EFPs. We also support visualization of the resulting EFPs for an allocation candidate.

Contribution. In this work we introduce *OptimAll*, a model-driven deployment optimization approach for multicore embedded soft real-time systems. In the long run *OptimAll* is meant to offer automatic full code generation capabilities as well as back-propagation features for optimization based on monitored system runs (as described in Section IV). In this paper we lay the foundations of the approach and we focus on the modelling and prediction-based optimization phases. Starting from a deployment model, we show how *OptimAll* automatically generates a suitable input for the task allocation optimization mechanism [3] through a set of model transformations, and then iterates the simulation runs in search for a good allocation candidate.

The remainder of the paper is organized as follows. Section II provides an overlook on related approaches documented in the literature, while Section III describes the basic concepts on which our approach is built upon. In Section IV we describe the *OptimAll* approach in all its intended features. In Section V a running example is exploited to show *OptimAll* at work. The paper is concluded with Section VI where we provide a short recall of the paper's contribution and an outlook on the coming planned activities.

II. Related work

MDE for embedded systems has a general goal of lifting the level of abstraction from code to models thus simplifying software development. Models are used both to reason about the EFPs of the system under development, and as a specification from which the implementation can be automatically generated. Regarding the former, models enable obtaining performance predictions already at an early stage of development, prior to the implementation, via model-based analysis and model simulation, in line with what software

performance engineering advocates [4]. These model-based performance predictions make it possible to quickly asses a large number of system configurations (e.g., deployment of software components to hardware nodes), thus enabling system optimization at an early development stage. In the remainder of this section, we present related work by discussing several approaches for model-driven performance prediction and optimization. However, in contrast to *OptimAll*, none of the approaches are specifically tailored for soft real-time multicore embedded systems, and cover the complete cycle of automatically going from models, through an optimization engine to code and back.

ProCom [5] is a component-based and model-based approach for developing automotive embedded systems. It has a notion of a rich component, which is a set of models, documentation and code. Through different modelling formalisms, ProCom can be used to analyse worst-case execution times, end-to-end response times, and resource usage. It also provides support for automatic synthesis of code from the models, however it does not enable optimization or back-propagation of information from code to models.

ArcheOpterix [6] is a framework for optimizing embedded systems modelled using the Architecture Analysis and Description Language (AADL). It supports several categories of EFPs, such as reliability and performance. The optimization mechanism employs various general purpose heuristics including genetic algorithms, Bayesian learning and hill climbing. The approach can account for uncertainties of design time parameter estimations, through its extension called Robust ArcheOpterix [7] — it proposes architectures that reduce the impact of the uncertainties.

Additional approaches (not limited to embedded systems) can be found in the survey of component-based approaches for performance evaluation by Koziolek [8], and in the survey of architecture optimization approaches by Aleti et. al. [9].

III. Background

In this section we set the domain of the work and describe techniques and technologies employed for defining the solution.

A. Task allocation problem

The domain of our work is represented by embedded soft real-time systems, where accurate timing behaviour is crucial for the correct functioning of the system, but occasional deadline misses are tolerated (as opposed to hard real-time systems where the absence of deadline misses must be guaranteed beforehand). Therefore, the EFPs we are interested in are related to timing, and include *end-to-end response time*, *deadline misses* and *core load*. These properties depend heavily on the allocation of tasks to cores. An intuitive example is allocating too many tasks to the same core, which will become overloaded and the tasks will therefore miss their deadlines.

It is desirable to identify a good allocation early in the development process, already prior to the implementation. The earlier design faults that lead to performance issues are caught, the cheaper and simpler it is to correct them [10]. Therefore we base our work on MDE concepts, and strongly rely on model-based analysis at design time. Since we focus on the average-case behaviour (as opposed to worst-case behaviour in hard real-time systems), and since the aforementioned EFPs depend heavily on the dynamic interplay between tasks, they cannot be derived analytically from task parameters. Rather, we obtain the property values by performing simulation of an allocation model.

B. Task model

In our approach we support *tasks* of two kinds: *periodic* and *event-triggered* (triggered by other tasks finishing their execution). Each task is assigned a number of parameters: *priority*, *affinity* (specifying which core the task is allocated to), *best-case execution time* (BCET), *worst-case execution time* (WCET). Moreover, in the specific case of periodic tasks, two additional parameters, namely *deadline* and *period*, are defined. An event-triggered task is considered to have missed its deadline if it is triggered again while its previous instance has still not finished executing. Currently, only a uniform distribution of task execution times is implemented. However, additional distributions are planned to be added. A *task chain* represents the flow of execution, and it is defined by a periodic task starting the chain and a set of event-triggered tasks triggered in ordered sequence. *End-to-end response times* are EFPs that are defined at chain level, and represent the duration between the point in time when the periodic task at the start of the chain begins its execution, until the point in time when the last task in the chain finishes its execution. During their execution tasks do not move between cores, as they are statically allocated. Each core has a *scheduler* in charge of running the tasks assigned to it and we currently support *preemptive* and *non-preemptive fixed priority-based* schedulers.

Next we formalise the aforementioned notions of task, periodic task, event-triggered task and chain.

Definition 1: A task T is a non-instantiable tuple $T = \langle B, W, pr, a \rangle$, where B represents T's BCET, W represents the WCET, pr represents T's scheduling priority and a represents the affinity parameter identifying the core to which T is allocated.

Definition 2: A periodic task PT is an instantiable specialization of T defined as the tuple $PT = \langle T, pe, d \rangle$, where T represents the tuple $\langle B, W, pr, a \rangle$, pe represents PT's period and d represents T's deadline.

Definition 3: An event triggered task ET is an instantiable specialization of T defined as the tuple $ET = T$, where T represents the tuple $\langle B, W, pr, a \rangle$.

Definition 4: A chain C is a non-empty ordered set of tasks $\{PT, T_1, T_2, \ldots, T_n\}$ with $|C| \geq 1$ and where the first element is always represented by a periodic task PT.

C. Modelling language

The reference modelling language exploited in the *OptimAll* approach is represented by UML [11] for functional descriptions, and by its profile for Modeling and Analysis of

Real Time and Embedded systems (MARTE) [12] for extra-functional as well as deployment modelling. Moreover, the approach is implemented and runs on top of MDT Papyrus [13], an open source integrated environment for editing EMF [14] models and particularly supporting UML and related profiles, on the Eclipse platform.

Concerning the functional modelling of the system we follow the component-based pattern [11] where each component is equipped with provided and required interfaces realised via ports and with state-machines and other standard UML diagrams to express functional behaviour. Moreover, the Action Language for Foundational UML (ALF) [15] is meant to be exploited for defining complex behaviours. Functional models are decorated with extra-functional information either through MARTE stereotypes or through specific annotations defined appositely for the purpose. For describing deployment information we exploit specific concepts provided by MARTE through which the modeller defines allocation of software components first to schedulable tasks and then to processing units.

D. Model transformations

Following the MDE paradigm, a system is developed by designing models and refining them starting from higher and moving to lower levels of abstraction until code is generated; refinements are performed through transformations between models. A *model transformation* translates a source model to a target model while preserving their well-formedness [2]. More specifically, in *OptimAll* we exploit the following kinds of model transformation:

- **Model-to-model (M2M):** which translates between source and target models that can be instances of the same or different languages;
- **Model-to-text (M2T):** which is a particular case of M2M where the target artefact is represented by text;
- **Text-to-model (T2M):** that operates in the opposite direction as the M2T, generating a model from a textual representation.

Moreover, any of these types of model transformations can be defined as *in-place*, meaning that source (or one of the sources) and target are represented by the same model; in this case, the transformation provides as output an updated version of (one of) the model(s) in input. Except for the in-place transformations which are by nature *endogenous*, the other transformations entailed in *OptimAll* are *exogenous* meaning that they operate between artefacts expressed using different languages [2]. M2M transformations are implemented with the Operational QVT[1] language, M2T transformations with Xpand[2], and T2M transformations with Java.

IV. The *OptimAll* approach

The goal of the *OptimAll* approach is to provide support to the developer in optimizing the deployment, already at design phase, by iteratively exploiting simulation based on (i) predicted performance-related EFPs, as well as on (ii) actual runtime performance-related EFPs gathered by monitoring

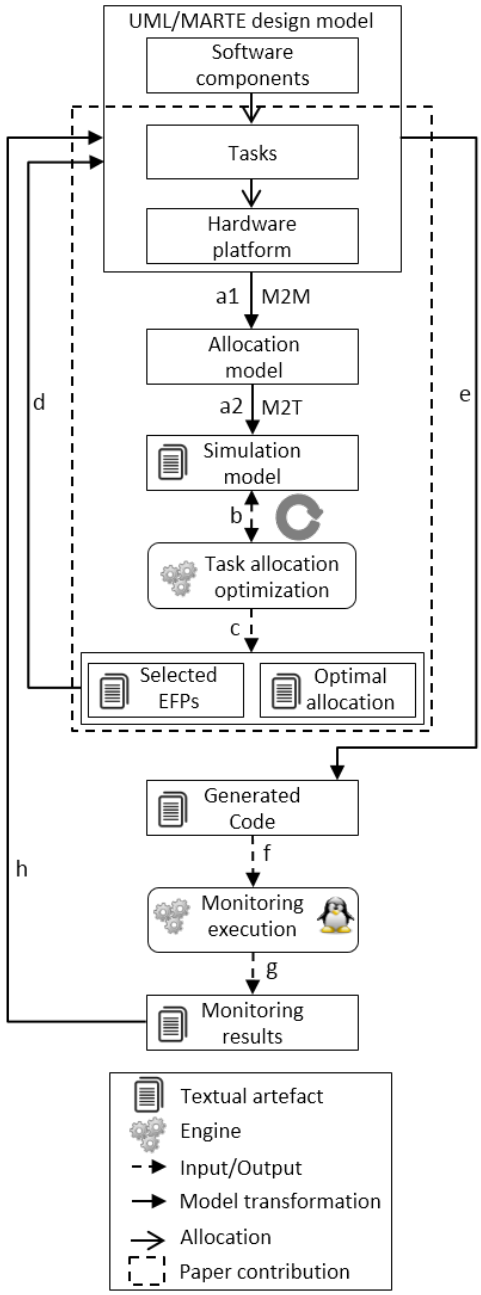

Fig. 1. The *OptimAll* Approach

the execution of automatically generated code. In Fig. 1 the *OptimAll* approach is depicted; note that the contribution of this paper provides a solution for the steps grouped in the dashed box. The approach is meant to operate at design level starting from a design system model defined in UML describing functional aspects. Deployment is modelled using MARTE and encompasses allocation of (i) software entities to tasks and (ii) tasks to specific processing units. From the design model, an M2M transformation (Fig. 1.a1) is in charge of generating an *allocation model* from which an M2T transformation (Fig. 1.a2) generates a *simulation model* to be fed as input to the *task allocation optimization* mechanism. This mechanism performs an iterative search process, where in each step the simulation model is executed, EFPs are derived

[1] http://www.eclipse.org/mmt/?project=qvto
[2] http://www.eclipse.org/modeling/m2t/?project=xpand

from the simulation results, and a new allocation candidate is generated for testing in the next iteration (Fig. 1.b). After the cycle finishes, it outputs the best allocation it was able to find (Fig. 1.c). Together with the allocation, the mechanism outputs the corresponding values of the EFPs and a visual trace of how the EFPs changed during the simulation (see Section V).

Simulation and optimization results are meant to be back-propagated (Fig. 1.d) through specific in-place M2M transformations to the design model in a similar way to what was proposed in [16]. *Selected EFPs* would be shown as extra-functional decorations of the functional model elements they pertain to, while the *optimal allocation* is meant to be shown in the design model through hints to the user by means of suggested optimal allocation links.

The first run of the task allocation optimization is based on predicted performance-related EFPs (expert estimations) and aims at providing a fair approximation of the actual optimal allocation. In order to reach a solution even closer to the theoretically optimal, *OptimAll* will have to exploit actual runtime values rather than predictions. In order to do that, the approach will first automatically generate target code from the design model (Fig. 1.e) taking into account the *best allocation* based on predictions (Fig. 1.d). Such an ability is pivotal in order not to jeopardize the consistency between modelling artefacts, as well as the validity of simulations and optimizations run on them, and the final implementation of the system. In this respect, the generated code is not meant to be edited by hand. Possible optimizations are indeed not performed directly through code editing, but rather by re-iterating the code generation process once the task allocation has been refined according to the optimization mechanism.

Actual runtime values of EFPs to be exploited by the task allocation optimization mechanism are gathered by monitoring the execution of the generated code (Fig. 1.f, 1.g), similarly to what is proposed in [17]. Once gathered, EFPs are back-propagated to the design model (Fig. 1.h), and finally the task allocation optimization mechanism can be re-run leveraging the back-propagated values. The whole approach is iterated until the user is satisfied with the identified allocation.

V. Solution

In this section we describe the various steps of the approach and show how they operate on a running example.

A. Modelling the system

Since in this work we address task allocation to cores, we focus on the modelling artefacts describing the deployment. In *OptimAll* the deployment of software components to the processing nodes is achieved through two intermediate layers: (1) a software component is allocated through a one-to-one connection to a specific schedulable task (stereotyped as ≪*swSchedulableResource*≫), (2) a task is allocated through a one-to-one connection to a core (stereotyped as ≪*hwComputingResource*≫). An excerpt of the deployment model we will exploit for showing the proposed solution at work is depicted in Fig. 2.

As mentioned in Section III, we support two kinds of tasks: (i) periodic (defined as *PeriodicTask* in the model), and

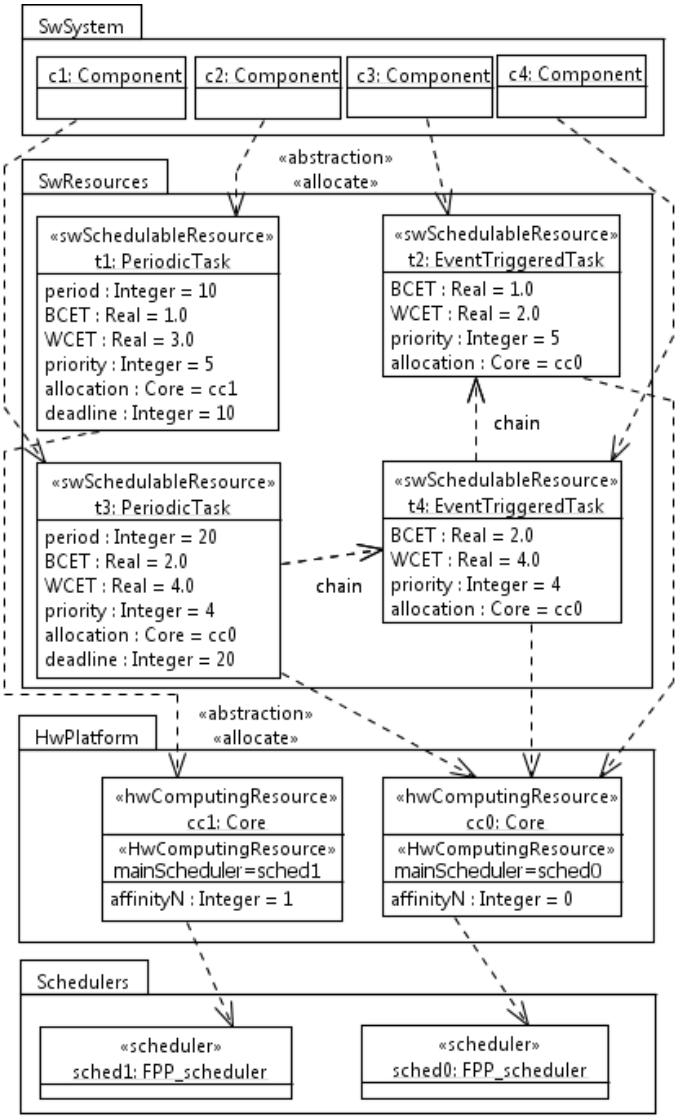

Fig. 2. Design model

(ii) event-triggered (defined as *EventTriggeredTask*). Cores are defined as *Core* in the model and they have a main scheduler, defined in the stereotype's property *mainScheduler* that refers to a scheduler instance (stereotyped as ≪*scheduler*≫). In the example two instances `sched0` and `sched1` of a fixed priority preemptive scheduler (defined as *FPP_scheduler*) are shown.

The allocation links are stereotyped as ≪*allocate*≫. In Fig. 2 we can see, e.g., that component `c1` is allocated to task `t3`, which is in turn allocated to core `cc0` with scheduler `sched0`. Additionally, since the task allocation optimization mechanism is based on task chains, we model this information through directed dependency links called *chain*; in the coming enhancements of the approach we plan to introduce a specific MARTE's stereotype providing the needed chain-related attributes for this purpose. In Fig. 2 we can see two chains, one constituted of periodic task `t3`, event-triggered task `t4` and event-triggered task `t2` and the other one constituted of the sole periodic task `t1`. In the next increment of *OptimAll* we plan to automatically derive this information from the

connections among software components in order to relieve the developer from a manual definition of chains which might be laborious and error-prone for complex systems.

As aforementioned, the model portion depicted in Fig. 2 does not represent the complete running example which in fact consists of the following elements:

- **Event-triggered tasks:** $t2 = \langle 1, 2, 5, cc0\rangle$, $t4 = \langle 2, 4, 4, cc0\rangle$, $t6 = \langle 3, 6, 3, cc1\rangle$, $t8 = \langle 4, 6, 2, cc0\rangle$, $t10 = \langle 5, 8, 1, cc1\rangle$;
- **Periodic tasks:** $t1 = \langle 1, 3, 5, cc1, 10, 10\rangle$, $t3 = \langle 2, 4, 4, cc0, 20, 20\rangle$, $t5 = \langle 3, 5, 3, cc0, 20, 20\rangle$, $t7 = \langle 4, 7, 2, cc0, 25, 25\rangle$, $t9 = \langle 5, 8, 1, cc0, 50, 50\rangle$;
- **Chains:** $c1 = \{t1\}$, $c2 = \{t3, t4, t2\}$, $c3 = \{t5, t6\}$, $c4 = \{t7, t8\}$, $c5 = \{t9, t10\}$.

B. Generating the simulation model

From the system model, the approach automatically extracts the information needed for running the task allocation optimization, that is to say tasks, chains, schedulers, cores and the allocation of tasks to cores. The generation of the simulation model that will be fed as input to the task allocation optimization, is a two-step process. First an M2M transformation generates an *allocation model*, conforming to the *allocation metamodel* depicted in Fig. 3. From the allocation model, an M2T transformation generates the actual *simulation model*, i.e. the input to the task allocation optimization mechanism. The reason for such a multi-step approach resides in our goal to maximise independence of the approach from the entailed modelling language. In fact, employing the approach for similar purposes starting from a non-UML system model would be possible just by redefining the M2M transformation generating the allocation model, while the rest of the approach would remain unchanged. Nevertheless, in order to have a full UML-complaint approach, we aim at bypassing the allocation model and generate the Java simulation model directly from the UML–MARTE system model, possibly leaving the allocation model as an optional generated artefacts.

The allocation metamodel depicted in Fig. 3 represents the allocation of tasks to cores and it is defined through Ecore in the Eclipse Modelling Framework[3]. The main element is `Configuration` that contains tasks, chains, schedulers and cores. Tasks are represented through the abstract metaclass `Task` that defines the common properties of a task. `Task` is specialised by (i) `EventTriggeredTask`, which represents event-triggered tasks, and (ii) `PeriodicTask`, representing periodic tasks. Any `Task` can trigger a number of `EventTriggeredTask` and is allocated to, at most, one `Core`. Each `Core` may have a scheduler (`Scheduler`) of type `NonpreemptivePriorityScheduler` or `PreemptivePriorityScheduler`. Chains are represented by the metaclass `Chain` which points to an ordered set of `Task` elements. Once the allocation model[4] is generated, an M2T transformation takes it as input to generate the *simulation model*. The simulation model can be seen as an executable textual representation of the allocation model. As the optimization mechanism is implemented using Java,

[3]https://www.eclipse.org/modeling/emf/

[4]Due to space limitation and to its straightforwardness once defined the allocation metamodel, we do not show the allocation model.

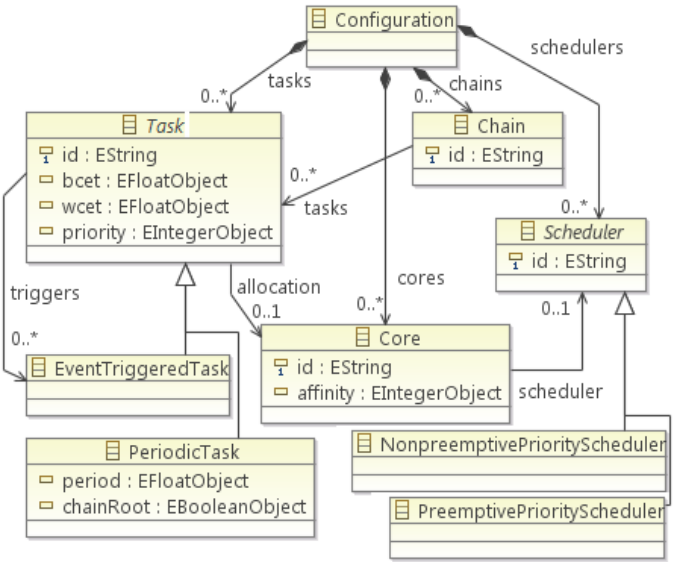

Fig. 3. Allocation metamodel

the simulation model is a Java class that reflects the allocation model — it defines the existing tasks, their groupings into chains and their allocation to the available cores.

C. Task allocation optimization

Task allocation optimization is performed with respect to end-to-end response times. The goal of the optimization is to keep the number of deadline misses in the system below a desired boundary, while minimizing the average response time for a selected task chain. Generally, the optimization mechanism is envisioned to be extended for other EFPs.

The simulation model serves as input to the task allocation optimization. In each step the optimization mechanism executes the simulation model, and then derives relevant EFPs from the obtained simulation results. The EFPs are used in order to quantify the current allocation candidate against the best allocation candidate found thus far. If the current allocation is better than the best one, it becomes the new best allocation. As last step in each iteration, the optimization mechanism proposes a new allocation candidate to be tested in the following iteration. This is done using our custom heuristic, which takes the best allocation candidate as basis, and then identifies a task to be relocated to a different core. The more a task delays other tasks and the more it is itself delayed by other tasks, the bigger the chance it will be picked for relocation. When choosing a new core for the picked task, core load is taken into account — the lower the load of a particular core, the bigger the chance that the picked task will be allocated there. More details and an evaluation of the heuristic can be found in [3]. Having relocated a task means that the simulation model is updated, and the updated version will be assessed in the next iteration of the optimization mechanism. After having performed the desired number of iterations, the optimization mechanism outputs the best allocation candidate it was able to find.

Next we illustrate one optimization run, using the running example. The optimization is set to run for 100 iterations. In

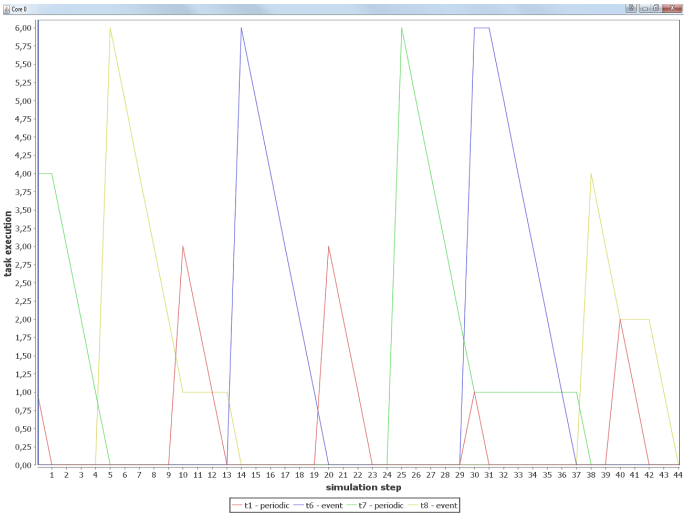

Fig. 4. Task execution trace

each iteration, the simulation is performed for 1000 time units, which corresponds to 10 hyperperiods in the running example. The chain we are optimizing (i.e., whose average response time is to be minimized) is $c2$. The limit of allowed deadline misses is set to 0. The starting allocation results with no deadline misses for chains $c1$ and $c2$, 1 deadline miss for $c3$, 30 deadline misses for $c4$ and 19 deadline misses for $c5$, so the heuristic will first try to find an allocation with no deadline misses. Task $t7$ is selected for relocation from core $cc0$ to core $cc1$. This new allocation candidate is assessed in the second iteration — $c3$ now has no deadline misses, $c4$ has 11, while $c5$ has 5. Since there are now less deadline misses in the system, this becomes the new best allocation candidate, and the one used as basis for proposing the next candidate. The optimization continues in the same way.

The best allocation is found in step 71: it has no deadline misses and an average response time for chain $c2$ of 7.46. A more detailed illustration and assessment of the optimization mechanism can be found in [3]. Other than the best allocation specification, as mentioned before, the optimization mechanism outputs a visual trace of the relevant performance metrics for any desired allocation. The ones currently supported are: task execution (an excerpt for the best allocation for core $cc0$ is shown in Fig. 4), core load, task deadline misses and chain deadline misses.

VI. Outlook

In this paper we introduced the *OptimAll* approach for model-driven deployment optimization of multicore embedded soft real-time systems. Besides an overall description of the foundations of the approach, we described in more details the core modelling and prediction-based optimization phases. From a deployment model, we showed how *OptimAll* automatically generates an input model for the task allocation optimization mechanism and then iteratively runs the simulation in search for a good allocation candidate. In the ongoing work we started to address the steps of the approach that were not covered by this contribution:

- **Automatic code generation**: from system models we aim at generating instrumented code tailored for multicore, which can be executed and monitored to gather the information needed to optimise deployment;
- **Monitoring**: when executing code, we want to be able to observe and gather selected EFPs through specific extensions to the platform (e.g., monitoring routines at OS-level);
- **Back-propagation of EFPs**: the values resulting from model-based simulation described in this work as well as the values gathered at runtime through monitoring shall be propagated back to the system models, in the form of both extra-functional decorations as well as computed textual/graphical allocation hints, for user's investigation.

Moreover, regarding the phases described in this paper, future enhancements will cover: (i) the automated generation of task chains from message passing and function calls among software components instead of manually modelling them, (ii) the entailment of multi-branch chains meant as the possibility for one task to be triggered by or to trigger several tasks.

Acknowledgment

This research was supported by Swedish Foundation for Strategic Research through grant IIS11-0060 (Ralf 3 project) and by ARTEMIS grant 333053 (CONCERTO project).

References

[1] S. Kent, "Model driven engineering," in *Procs of IFM*. Springer-Verlag, 2002, pp. 286–298.

[2] K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," in *Procs of OOPSLA*, 2003.

[3] J. Feljan and J. Carlson, "Task Allocation Optimization for Multicore Embedded Systems," in *Procs of SEAA*, 2014.

[4] M. Woodside, G. Franks, and D. C. Petriu, "The Future of Software Performance Engineering," in *Procs of FoSER*, 2007, pp. 171–187.

[5] S. Sentilles, A. Vulgarakis, T. Bureš, J. Carlson, and I. Crnković, "A component model for control-intensive distributed embedded systems," in *Procs of CBSE*, 2008.

[6] A. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, "ArcheOpterix: An extendable tool for architecture optimization of AADL models," in *Procs of MOMPES*, 2009.

[7] I. Meedeniya, A. Aleti, I. Avazpour, and A. Amin, "Robust ArcheOpterix: Architecture optimization of embedded systems under uncertainty," in *Procs of SEES*, 2012.

[8] H. Koziolek, "Performance evaluation of component-based software systems: A survey," *Performance Evaluation, Special Issue on Software and Performance*, vol. 67, no. 8, 2010.

[9] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, 2013.

[10] J. Bézivin, "On the unification power of models," *Software & Systems Modeling*, vol. 4, no. 2, pp. 171–188, 2005.

[11] Object Management Group (OMG), "UML Superstructure Specification V2.3," http://www.omg.org/spec/UML/2.3/Superstructure/PDF/, Last Accessed: July 2013.

[12] S. Taha, A. Radermacher, S. Gérard, and J.-L. Dekeyser, "MARTE: UML-based Hardware Design from Modelling to Simulation," in *Procs of FDL*, 2007, pp. 274–279.

[13] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "Papyrus: A UML2 Tool for Domain-Specific Language Modeling," in *Model-Based Engineering of Embedded Real-Time Systems*, 2007, pp. 361–368.

[14] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose, *Eclipse Modeling Framework*. Addison Wesley, 2003.

[15] Object Management Group, "Action Language For FoundationalUML - ALF," http://www.omg.org/spec/ALF/, Oct 2010.

[16] F. Ciccozzi, A. Cicchetti, and M. Sjödin, "Round-Trip Support for Extra-functional Property Management in Model-Driven Engineering of Embedded Systems," *Information and Software Technology*, 2012.

[17] F. Ciccozzi, M. Saadatmand, A. Cicchetti, and M. Sjödin, "An Automated Round-trip Support Towards Deployment Assessment in Component-based Embedded Systems," in *Procs of CBSE*, June 2013.

A Tool for the Optimal Design of Soft Real-Time Systems

Luigi Palopoli, Luca Abeni, Daniele Fontanelli
Dipartimento di Scienza e Ingegneria dell'Informazione
University of Trento, Trento, Italy
{*luigi.palopoli,luca.abeni,daniele.fontanelli*}*@disi.unitn.it*

***Abstract*—In recent years a series of important achievements have paved the way for the introduction of probabilistic analysis in the area of soft real–time systems design. In this paper, we present an extensible design tool that facilitates the access to this technology for a potentially large number of researchers and industrial practitioners. Although the tool is currently limited to single–task applications, it enables probabilistic analysis of the temporal performance of real–time tasks under fixed priority and reservation based scheduling. In case of resource reservations, the tool also supports the synthesis of scheduling parameters that optimise a quality metric related to the probabilistic behaviour of the tasks.**

***Keywords*-Soft real–time systems, Probabilistic Guarantees**

I. Introduction

In the classic context of hard real–time systems, a system consists of a set of tasks; each one generates a stream of jobs that have to terminate *before* a deadline. For soft real–time systems this requirement is too strict and can be revisited in a probabilistic sense. One of the possibilities is to specify a probability for the termination of a task within a deadline, or more generally, a desired distribution of the delays.

It is argued that the Quality of different type of industrially relevant applications can be easily related to such probabilistic metrics. This consideration, along with the ever-increasing variability of modern real–time applications, has stimulated an intense research on probabilistic real–time systems. A very important line of work is on the probabilistic generalisation of analysis technique initially developed in the hard real–time domain such as time demand analysis [1] and computation of the response time of tasks scheduled with fixed priority and with stochastic computation time [2], [3], the inter-arrival time [4] or both [5].

Other papers assume reservation based scheduling [6]. In a reservation-based scheduler tasks enjoy the temporal isolation property, which allows the designer to decouple the behaviour of the different tasks. Based on this, Abeni et al. [7] have proposed an analysis technique based on queueing theory for reservation-based schedulers. Recently, approximated solution techniques have been proposed to decrease the complexity of the algorithm used to compute the probability of meeting the deadline [8], [9].

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n ICT-2011-288917 DALi - Devices for Assisted Living and by the HYCON2 NoE, under grant agreement FP7-ICT-257462.

The time is ripe to make some of the most effective techniques for probabilistic analysis of real–time systems available to a wide community of researchers and industrial practitioners. This is the purpose of the PROSIT (PRObabilistic deSIgn of real-Time systems) tool, which is based on two important concepts: stochastic analysis of soft real–time systems and definition of a quality metric which depends on *probabilistic deadlines*. In the recent literature, it is shown that the analysis of the stochastic behaviour of real–time tasks can be reduced to that a of particular class of discrete–time Markov Chains (DTMC) called Quasi-Birth-Death process [10]; this holds both for fixed priority scheduling [2], [3] and for resource reservations [8]. This property allowed us to use some of the most efficient numeric and analytic algorithms [11] as a basis for the implementation of the system analysis. The definition of the Quality as a function of the probability of meeting deadlines (or more generally of the distribution of the delays) allowed us to revisit in a probabilistic framework the ideas proposed in such frameworks as QRAM [12], and to develop algorithms for the optimal choice of the scheduling parameters.

II. Problem Presentation

In this paper, we consider a set of independent applications each one composed of various real–time tasks. The applications share the same computing resources, but do not interact in other ways. In this paper, we restrict our focus to single-task applications, reserving extensions to multi-task applications for our future work.

A real-time task τ_i is a stream of jobs $J_{i,k}$; job $J_{i,k}$ arrives (becoming executable) at time $r_{i,k}$, finishes at time $f_{i,k}$ after executing for a time $c_{i,k}$, and is characterised by a deadline $d_{i,k} = r_{i,k} + D_i$, (where D_i is the *relative deadline* of τ_i). We consider a generalisation of the traditional notion of deadline, called *probabilistic deadline* [7]. A probabilistic deadline is a pair (δ_i, p_i), where p_i is the steady-state probability of respecting a relative deadline δ_i ($d_{i,k} = r_{i,k} + \delta_i$). The probabilistic deadline is respected if $\mathbf{Pr}\{f_{i,k} \leq r_{i,k} + \delta_i\} \geq p_i$. It is possible to specify a sequence of probabilistic deadlines for τ_i, $\left(\delta_i^{(1)}, p_i^{(1)}\right)$,

$\ldots$, $\left(\delta_i^{(H_i)}, p_i^{(H_i)}\right)$, where $p_i^{(h)} \geq p_i^{(f)}$ if $\delta_i^{(h)} > \delta_i^{(f)}$, formulating in this way a specification on the distribution of the delays.

The computation time of each job $c_{i,k}$ is a stochastic process $\mathcal{U}_i$ described by the probability mass function (PMF) $U_i(c) = \mathbf{Pr}\{c_{i,k} = c\}$. Likewise, the inter-arrival time (the distance between two subsequent activations) is assumed to be a stochastic process $\mathcal{I}_i$ with PMF $I_i(t) = \mathbf{Pr}\{r_{i,k+1} - r_{i,k} = t\}$. As a special case, when the $I_i(t)$ distribution is given by a *Kronecker* delta centred in T_i ($I_i(t) = \delta(t - T_i)$) the task is periodic of period T_i.

A. Application Quality

While probabilistic deadlines effectively capture the real–time constraints, the quality perceived by the user of the application is not immediately expressed by such quantities. Much more useful is a quality index μ_i, which is necessarily application specific and is related to the design choices. This index is related to the probabilistic deadlines by a functional dependence. In the simplest case, we can consider a single probabilistic deadline where δ_i is equal to the relative deadline D_i. In this case the index μ_i is a monotone non decreasing function of probability p_i. More generally, if we consider a sequence of probabilistic deadlines, μ_i is a function of the vector $\left[p_i^{(1)}, \ldots, p_i^{(H_i)}\right]$.

Our notion of a Quality function is inspired to the QRAM framework [12]. However, QRAM assumes a functional dependence between scheduling parameters and μ_i, which can be difficult to identify. On the contrary, we relate the quality to the probabilistic temporal behaviour of the task, which depends on the scheduling parameters in non-obvious ways. This allows the designer to reason about the system quality in a natural conceptual framework without committing to any particular scheduling policy.

As an example, in the domain of control applications, a quality index of this kind could be the steady state covariance of the controlled plant state, which is known to be a function of the distribution of the delays [13]. Likewise, for a media processing application, we can define μ_i as the PSNR or SSIM between the original media and the one reproduced by the application [14]. Under the assumption that video frames decoded by jobs finishing late are not displayed, when a deadline is missed the PSNR or SSIM is computed using the last frame decoded in time [15] and degrades with the frequency of this event.

B. Scheduling

Although fixed priority scheduling [16] is probably not the best possible choice for soft real–time systems, its enormous popularity suggested us to consider it within the PROSIT framework.

A more reputable scheduling choice for this domain are the so called *CPU Reservations* [6]: each task τ_i is associated with a reservation (Q_i^s, T_i^s), meaning that τ_i is guaranteed to execute for Q_i^s (*budget*) time units in every interval of length T_i^s (*reservation period*). The PROSIT tool operates with any reservation scheduler as long as it respects the *temporal isolation* property: the amount of computation time Q_i^s is reserved to τ_i regardless of the behaviour of the other tasks. The ratio $B_i = Q_i^s/T_i^s$ is termed *bandwidth* and corresponds to the fraction of CPU allocated to the task.

C. The Analysis and the Synthesis problem

The first problem we address, called the *Analysis Problem*, can be shortly described as follows: *given a set of applications, each one characterised by its probability distributions, a quality function, a scheduling algorithm, a set of scheduling parameters* par_i *and a sequence of probabilistic deadlines, decide if the steady state probabilities* $\left[p_i^{(1)}, \ldots, p_i^{(H_i)}\right]$ *are respected and compute the Quality* μ_i *for each of the applications*. The scheduling parameters par_i are given by the priorities prio_i for a fixed priority scheduler and by the pair (Q_i^s, T_i^s) for a resource reservation scheduler.

In the analysis problem the scheduling parameters are assumed chosen by the designer. The synthesis problem is different since it requires the computation of optimal scheduling parameters. The idea is that a choice of scheduling parameters determines a set of steady state distribution and hence a different quality for each of the applications. The different qualities can be collected to form a global quality function $\mathbf{f}(\mu_1, \ldots, \mu_n)$. The synthesis problem can be expressed as $\max_{\text{par}_1, \ldots, \text{par}_n} \mathbf{f}(\mu_1, \ldots, \mu_n)$ subject to $\mu_i \geq l_i$ and to a schedulability condition. An example of schedulability condition for resource reservations is CPU time allocated to the different tasks cannot exceed a specified value $\sum_{i=1}^{n} B_i \leq U^{lub}$ [7].

III. Overview of the Tool

The typical workflow in using the PROSIT tool are sketched in Figure 1, which represent two use cases related to the solution of the problems described above.

A. The Analysis Problem

For the analysis problem (top half of the figure), the designer is required to provide the definition of the task set that he/she wishes to analyse.

If the chosen scheduling algorithm is resource reservation, in order to analyse one task the user only needs to provide its timing requirements (distribution of the inter-arrival time $I_i(t)$ and of its computation time $U_i(c)$) and its scheduling parameters (Q_i^s and T_i^s). This is because the temporal isolation property allows decoupling the analysis of the different tasks. On the contrary, if fixed priority scheduling is used, the analysis is possible only for periodic tasks and requires the knowledge of the temporal requirements of the task and of all those having a higher priority [2], [3]. The distributions can either be chosen from a library (e.g., uniform, beta, Gaussian, etc.) customising the required

parameters or specified in a file. In addition to specifying the task set, the user also chooses a solution algorithm for the probabilistic analysis and the Quality evaluator (i.e., the function μ_i relating the application quality to the probabilistic deadlines). This information can be inserted using command line parameters or a system specification file.

As a result of the tool invocation, a C++ object is generated that captures all the information required for the analysis. One of the methods of this object (`solve`) computes the probabilistic deadlines. This is a virtual method, which is overloaded with the specific solution strategy chosen by the user. The quality corresponding to the distribution of the probabilities for the probabilistic deadlines is computed by a different method (`Quality`), which is itself virtual to allow for different quality models. The tool prints the result and the computation time to screen (and/or to a file).

B. The Synthesis Problem

The solution of the synthesis problem is currently available only for the resource reservations. The tool requires a system specification file containing a description of the temporal information of all the tasks (see Figure 1). For each task, the user specifies the temporal parameters, the server period T_i^s, the solution algorithm, the Quality function and the minimum required value for the quality. The budget Q_i^s is computed by the optimisation algorithm. One could legitimately argue over the choice of fixing T_i^s and leaving Q_i^s as decision variable. The motivation is rooted in the philosophy of the resource reservations, in which T_i^s is used to control the granularity of resource allocation, while Q_i^s is used to control the bandwidth.

In the system specification file, the user also specifies the global quality function $\mathbf{f}$. This function composes the quality associated with each application, which is a non-negative real number. Possible choices are the infinity norm - $\mathbf{f}(\mu_1, \ldots, \mu_n) = \max_{i=1}^{n} \mu_i$ - and the one norm - $\mathbf{f}(\mu_1, \ldots, \mu_n) = \sum_{i=1}^{n} \mu_i$. After the tool execution is started, the parser generates C++ object instances for each task and hands them over to the optimiser (which is itself a C++ object). The optimisation algorithm iteratively calls the `quality` method of the task (and indirectly the `solve` method) to compute the quality associated to a choice of decision variables or to estimate the gradient and eventually produces the optimal choice of parameters.

A possible workflow to derive the information required by the PROSIT tool for its operations is shown in Figure 2: the application is executed on a real (instrumented) kernel (for example, the Linux kernel with `ftrace`) to extract information about the tasks' inter-activation and execution times, etc. Such information is then inserted in a system specification file, which is used by PROSIT to define the type of problem (analysis or synthesis), describe the taskset, specify the scheduler, and provide a Quality model. Each task is associated with a symbolic name and a type, that can be chosen from a library of existing types. A task type is associated with a solution algorithm for the probabilities (in the example the analytic bound [9]) and with the quality computation. In the tasks definition it is possible to specify inter-arrival times and execution times (both inter-arrival and execution times can be constant or described by a Probability Mass Function). The user-provided Quality model is specified by a type (chosen in a library) and by a set of parameters. For example, it is possible to adopt a simple linear model (see Section VI for more details) with a lower bound for the quality of service. An important problem is how to derive the computation requirements of the task (distribution of computation time, period, distribution of the inter-arrival time). This is relatively easy if the designer is in control of the source code and can instrument it with probes pinpointing the start and the termination of each job. The task is much harder for legacy applications and can be accomplished using two external modules: a tracer that operates inside the kernel and notes all the events related to the application and an event analyser, which detects periods and estimates the distribution of the computation time.

C. The Application Programming Interface

The tool is based on a C++ library designed to be flexible and extensible. Most of the features of the tool are exposed to the software developer through an API for the possible benefits of using some of the library components outside the PROSIT tool. For instance, one could use the library for an admission test based on probabilistic deadlines.

The library can be easily extended in several directions by: 1) the definition of new solution algorithms for probability computation, 2) the definition of new quality metrics, 3) the definition of different distribution types, 4) the definition of different optimisation algorithms. In some cases, such extensions are made by sub-classing and redefinition of a few methods. In others (e.g., for the quality metrics), the user has to define her/his own function objects using a *factory* design pattern [17], along with a couple of auxiliary functions for XML parsing, which make the newly defined extensions readily accessible within the tool.

IV. Algorithms Analysis and Synthesis

A. Probabilistic Analysis

The cornerstone of our tool is the probabilistic analysis. For the sake of brevity, let us focus on periodic tasks using reservation-based scheduling. As shown in the literature [7], a stochastic process $v_{i,j}$ can be introduced to model the amount of time to be executed after the arrival of the j^{th} job $J_{i,j}$. The initial backlog $v_{i,0}$ is obviously equal to $c_{i,0}$ and $v_{i,j+1}$ can be easily computed as

$$v_{i,j+1} = \max\{0, v_{i,j} - N_i Q_i^s\} + c_{i,j+1} \qquad (1)$$

where $N_i = T_i/T_i^s$ is an integer number describing the number of reservation periods contained in the task

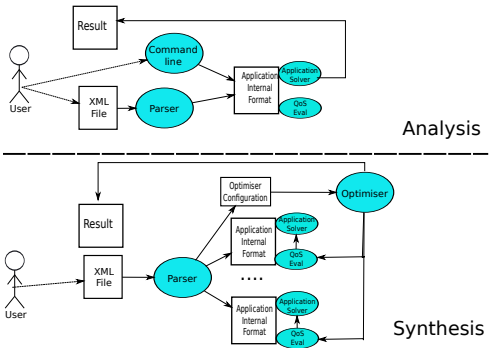

Figure 1. Use cases of the PROSIT tool

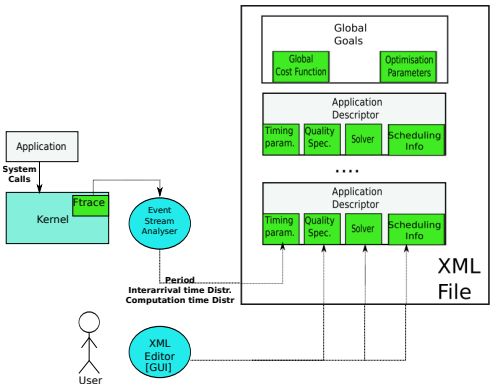

Figure 2. Definition of the optimisation problem with the PROSIT tool

period T_i. This model is easily generalised to the case of aperiodic tasks (making the conservative simplification that the activation of each job is considered only at the beginning of a reservation period) [8] and describes the evolution of a discrete–time Markov Chain (DTMC). A very important feature of this model is its recursive structure, leading to which qualifies the DTMC as a Quasi-Birth-Death (QBD) process. As discussed by Diaz et al. [2], if one considers the probabilistic a set of tasks scheduled by fixed priority, the analysis reduces to one of a DTMC having the same structure. This consideration are of the greatest importance because very effective numeric solutions exists for QBD processes. In particular, in the PROSIT tool we have implemented logarithmic reduction [18] and cyclic reduction [11]. The two implementations can serve as a template for the implementation for any of the plethora of algorithms documented in the literature [10].

Given the specific structure of the QBD, it is also possible to compute an analytical bound for the probability of deadline miss. The algorithm is shown in [9]; as discussed in the paper and shown in the experimental section this bound is particularly useful in the synthesis problem for the computation of suboptimal solutions of acceptable quality with substantial savings of time.

Procedure 1 Solve Optimisation

Input: taskList; /*List of the tasks */; $\underline{\mu}_i$; /* Lower bounds for quality*/
Output: μ; /*optimal value */ Q_i^s; /* Optimal budgets */

1: $B_A = 1.0$;
2: $\forall_i \ \overline{\mu}_i = \tau_i$.reservationPeriod();
3: **if** $\sum_{\tau_i \in \text{taskList}} \frac{\tau_i.\text{invQuality}(\underline{\mu}_i)}{\tau_i.\text{reservationPeriod}()} > 1.0$ **then return** UNFEASIBLE;
4: **end if**
5: $\underline{\mu} = \min_{i=1,\ldots,n} \underline{\mu}_i$; $\overline{\mu} = \min_{i=1,\ldots,n} \overline{\mu}_i$
6: **for** $\tau_i \in$ taskList **do**
7: $\quad Q_i^s = \tau_i.\text{invQuality}(\overline{\mu})$;
8: $\quad$ **if** $Q_i^s < \tau_i.\text{invQuality}(\underline{\mu}_i)$ **then** $Q_i^s = \tau_i.\text{invQuality}(\underline{\mu}_i)$;
9: $\quad\quad$ taskList$-= \tau_i$; $B_A -= \frac{Q_i^s}{\tau_i.\text{reservationPeriod}()}$;
10: $\quad$ **end if**
11: **end for**
12: **if** $\sum_{\tau_i \in \text{taskList}} \frac{Q_i^s}{\tau_i.\text{reservationPeriod}()} < 1.0$ **then**
13: $\quad \mu = \overline{\mu}$;
14: **else**
15: $\quad$ **while** $\overline{\mu} - \underline{\mu} > 0$ **do**
16: $\quad\quad \mu = \left(\overline{\mu} + \underline{\mu}\right)/2$;
17: $\quad\quad$ **for** $\tau_i \in$ taskList **do**
18: $\quad\quad\quad Q_i^s = \tau_i.\text{invQuality}(\mu)$;
19: $\quad\quad\quad$ **if** $Q_i^s < \tau_i.\text{invQuality}(\underline{\mu}_i)$ **then** $Q_i^s = \tau_i.\text{invQuality}(\underline{\mu}_i)$;
20: $\quad\quad\quad\quad$ taskList$-= \tau_i$; $B_A -= \frac{Q_i^s}{\tau_i.\text{reservationPeriod}()}$;
21: $\quad\quad\quad$ **end if**
22: $\quad\quad$ **end for**
23: $\quad\quad$ **if** $\sum_{\tau_i \in \text{taskList}} \frac{Q_i^s}{\tau_i.\text{reservationPeriod}()} < B_A$ **then** $\underline{\mu} = \mu$
24: $\quad\quad$ **else** $\overline{\mu} = \mu$
25: $\quad\quad$ **end if**
26: $\quad$ **end while**
27: **end if**

B. Synthesis

As discussed in Section II, the synthesis amounts to the solution of optimisation algorithm, which in the current version of the tool is possible only for resource reservations. The solution is very efficient when the infinity norm is used and the following assumption can be made: *the quality increases if the budget reserved to the application, and hence the probability of meeting the deadline increases*. In all cases of practical relevance that we have examined, this assumption is easily verified.

The monotonicity of the function allows us to apply the efficient solution algorithm reported in Procedure 1. The first lines of the algorithm (3 through 5) are to verify if the total bandwidth required to attain the lower bounds of the specification exceeds 100%, the problem being unfeasible in this case. The search for the optimal solution is reduced to within two bounds (line 6); the lower one ($\underline{\mu}$) derives form the lower bound constraints, while the higher one is obtained by assigning 100% of the bandwidth to the task ($Q_i^s = T_i^s$). The method `invQuality` computes the budget required to attain a specified level of quality; since the Quality is assumed monotone increasing, the inversion can be carried out by a simple dichotomic search. The operation can be expensive because it entails repeated calls to the `solve` method. The code segment between line 7 and 15 computes the budget required to each task for the upper bound $\overline{\mu}$. If

some of the tasks is constrained to a lower bound $\underline{\mu}_i$ higher than $\overline{\mu}$, this task is removed from the subsequent search phases and it is allocated a bandwidth sufficient to attain its lower bound. The execution of the algorithm is terminated if the total required bandwidth is lower than 100% (meaning that $\overline{\mu}$ is attainable and is therefore the optimum). In the opposite case, a binary search is carried out, in which the same steps applied to $\overline{\mu}$ between line 7 and 15 are applied to the midpoint between $\underline{\mu}$ and $\overline{\mu}$. The search is stopped when the two extreme coincides. It can easily be shown that this algorithm converges to the optimum.

V. External Modules

As shown in Figure 2, the distributions of the execution and inter-arrival times $U(c)$ and $I(t)$ can be extracted by analysing the execution traces of the task. As an example, the external tool TrcUtils [19] can be used for this purpose. The tool uses the Linux `ftrace` functionality to capture such traces and is organised as a pipeline of trace filters: the first stage of the pipeline (the import filter) transforms the text traces generated by `ftrace` (which contain redundant information) into an internal binary format, which only contains the relevant information and can be used by later stages of the pipeline. The next stages of the TrcUtils pipeline consist of a second set of filters that export traces in different formats, parse the internal format to gather various statistics about tasks execution, display the schedule, etc. In this context, the interesting functionality is the generation of PMFs of the execution and inter-arrival times, which can be exported by a new filter into the PROSIT XML format. Currently, this mechanism operates correctly only for self-suspending tasks (that is, real-time tasks for which a job never blocks, and the task blocks only at the end of each job). Work is being done to overcome this limitation taking inspiration from other techniques [20].

Another important activity supported by external tools is the generation of the mapping between probabilistic deadlines and applications' quality. Work is in progress to use the PSNRTools to evaluate the quality of media processing tasks, based on the PROSIT outputs. PSNRTools is a set of video processing tools that can encode an original video stream according to some specified parameters (using ffmpeg) and remove some parts of the video that have been lost or corrupted by some processing application to generate an output stream. A list of lost frames can be generated, for example, by considering the probabilistic deadlines computed by PROSIT. PSNRTools can then evaluate the differences between the output stream and the original uncompressed video stream, by computing the PSNR or SSim between them; when a frame is lost, the quality index is computed by using the latest correct output frame (a behaviour similar to the one of a real player).

VI. Application examples

In order to show the application of the tool, we considered a scenario where four periodic real–time applications are scheduled through a reservation-based scheduler. In Table I we report for each task period (T_i), server period (T_i^s), a triplet describing its computation time (where the first two elements are the best and worst case execution times and the third element is the distribution), the lower bound for the QoS $\underline{\mu}_i$ and the quality of service function. By $\beta_{a,b}$ we denote the beta distribution $\beta_{a,b}(c) \propto r(c)^{a-1}\left(1-r(c)\right)^{b-1}$ defined over the range $[c_{min}, c_{max}]$, where $r(c) = (c - c_{min}) / (c_{max} - c_{min})$. The Quality is, in this example, a very simple function of the probability p of meeting a deadline set equal to the period. Specifically, $\textbf{linear}_{p_{min}, p_{max}, \alpha}(p)$ denotes a function equal to 0 if $p \leq p_{min}$, to $\alpha\,(p - p_{min})$ if $p_{min} < p < p_{max}$ and $\alpha\,(p_{max} - p_{min})$ for $p \geq p_{max}$. We have executed the synthesis procedure using the infinity norm as the global cost function. This was done using as solvers both the analytic bound [9] and the cyclic reduction (CR) [11]. The former produces a conservative bound on the probability of meeting the deadline. Thereby, an optimisation algorithm based on the analytic bound simply produces suboptimal solutions. On the contrary, the CR solver produces an exact solution for the probability (within the limits of a numeric solution) and hence the application of the optimisation algorithm described in the previous section produces the optimal solution.

The results of the optimisation using analytic bound and CR are reported in the second and in the third group of columns of Table I. In particular, we report the optimal value of the budget, the corresponding probability of meeting the deadline and the value of the quality function. In order to make a fair comparison, we have re-evaluated the probability for the optimal budgets using an exact solver (CR) also for the analytic solution. The CR solution clearly produces a closer approximation of the optimal as compared to the suboptimal produced by the analytic solution. Indeed, the infinity norm of the quality is 0.45 for CR and 0.36 for the analytic bound. But, the computation time was 15744 μs for the analytic bound and 8963983 μs for CR. These computation times have been measured on a MacBook Air equipped with an Intel Core I7 dual processor operated at 1.7Ghz and with 8GB of RAM; the tool was compiled with a gcc 4.7.3 and `-O8` optimisation switch. As a preliminary observation, the analytic approach is viable if a quick computation is required (e.g., if the tool is used on-line), while the exact approaches are preferable when an offline execution of the tool allows for a more precise solution.

VII. Conclusions

We have described a software tool for probabilistic design of real–time systems called PROSIT, shown concrete use cases and discussed its algorithmic foundations. An intense

Table I
EXPERIMENTAL RESULTS

Task Description						Analytic Sol.			CR Sol.		
Task	T	T_i^s	$[c_{min}, c_{max}, U_i(c)]$	$\underline{\mu}$	Quality func.	Q_i^s	p_i	μ	Q_i^s	p_i	μ
τ_1	400	200	[10, 370, $\beta_{1.5,\,15.1}$]	0.1	linear$_{0.01,\,0.85,\,0.7}$	31	0.67	0.46	30	0.67	0.46
τ_2	600	100	[20, 560, $\beta_{1.5,\,10.1}$]	0.1	linear$_{0.01,\,0.92,\,0.5}$	27	0.86	0.42	30	0.92	0.45
τ_3	300	100	[20, 260, $\beta_{1.5,\,8.1}$]	0.1	linear$_{0.01,\,0.7,\,1.1}$	26	0.69	0.74	24	0.56	0.61
τ_4	300	100	[20, 200, $\beta_{1.5,\,8.1}$]	0.1	linear$_{0.01,\,0.95,\,0.5}$	23	0.73	0.36	28	0.92	0.46

development activity covering a significant number of features of the tool, such as a full support for fixed priority, and different optimisation algorithms, is currently ongoing. From the modelling point of view, we are looking at different types of real–time applications (e.g., multi-task and distributed applications); the analysis of real applications will extend the library of available quality functions.

REFERENCES

[1] M. K. Gardner and J. Liu, "Analyzing stochastic fixed-priority real-time systems," in *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS99)*. Springer, March 1999, pp. 44–58.

[2] J. Diaz, D. Garcia, K. Kim, C.-G. Lee, L. Lo Bello, J. Lopez, S.-L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, 2002, pp. 289–300.

[3] J. L. Diaz, J. M. López, M. Garcia, A. M. Campos, K. Kim, and L. Lo Bello, "Pessimism in the stochastic analysis of real-time systems: Concept and applications," in *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*. IEEE, 2005, pp. 197–207.

[4] L. Cucu-Grosjean and E. Tovar, "A framework for the response time analysis of fixed-priority tasks with stochastic inter-arrival times," *ACM SIGBED Review - Special issue: The work-in-progress (WIP) session of the RTSS 2005*, vol. 3, no. 1, pp. 7–12, January 2006.

[5] G. A. Kaczynskit, L. Lo Bello, and T. Nolte, "Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, September 2007.

[6] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia systems," in *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[7] L. Abeni and G. Buttazzo, "Stochastic analysis of a reservation-based system," in *Proceedings of the 15th International Parallel and Distributed Processing Symposium.*, San Francisco, California, April 2001.

[8] N. Manica, L. Palopoli, and L. Abeni, "Numerically efficient probabilistic guarantees for resource reservations," in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE, 2012, pp. 1–8.

[9] L. Palopoli, D. Fontanelli, N. Manica, and L. Abeni, "An analytical bound for probabilistic deadlines," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*. IEEE, 2012, pp. 179–188.

[10] D. Bini, G. Latouche, and B. Meini, *Numerical methods for structured Markov chains*. Oxford University Press Oxford, 2005.

[11] D. Bini and B. Meini, "On cyclic reduction applied to a class of toeplitz-like matrices arising in queueing problems," in *Computations with Markov Chains*. Springer, 1995, pp. 21–38.

[12] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for qos management," in *Proceedings of the IEEE Real Time Systems Symposium*, 1997.

[13] D. Fontanelli, L. Greco, and L. Palopoli, "Soft RealTime Scheduling for Embedded Control Systems," *Automatica*, vol. 49, pp. 2330–2338, July 2013.

[14] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid a framework for video transmission and quality evaluation," in *Computer Performance Evaluation. Modelling Techniques and Tools*, ser. Lecture Notes in Computer Science, P. Kemper and W. Sanders, Eds., vol. 2794. Springer, 2003.

[15] C. Kiraly, L. Abeni, and R. L. Cigno, "Effects of p2p streaming on video quality," in *Communications (ICC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–5.

[16] C. L. Liu and J. Layland, "Scheduling alghorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, 1973.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: Abstraction and reuse of object-oriented design*. Springer, 1993.

[18] G. Latouche and Ramaswami, "A Logarithmic Reduction Algorithm for Quasi-Birth-Death Processes," *Journal of Applied Probability*, pp. 650–674, 1993.

[19] L. Abeni and N. Manica, "Interoperable tracing tools," in *Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS2012)*, Pisa, July 2012.

[20] T. Cucinotta, F. Checconi, L. Abeni, and L. Palopoli, "Self-tuning schedulers for legacy real-time applications," in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 55–68.

SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms

Maxime Chéramy*, Pierre-Emmanuel Hladik* and Anne-Marie Déplanche†
*CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
†IRCCyN UMR CNRS 6597, (Institut de Recherche en Communications et Cybernétique de Nantes), ECN,
1 rue de la Noe, BP92101, F-44321 Nantes cedex 3, France

***Abstract*—In this paper, we present SimSo, a simulator designed for the comparison and the understanding of real-time scheduling policies. This tool is designed to facilitate the implementation of schedulers in a realistic way. Currently, more than twenty-five scheduling algorithms are available in SimSo. A particular attention is paid to the control of the computation time of the jobs therefore introducing more flexibility, for instance by taking into account cache-related preemption delays. In addition, SimSo offers an easy way to generate the tasksets, to perform simulations and to collect data from the experiments.**

I. Introduction

Davis and Burns referenced more than thirty real-time multiprocessor scheduling algorithms in 2011 [11] and more than a dozen of new algorithms have emerged since then, e.g. [26], [22]. Such a large number of scheduling algorithms makes their evaluation and comparison difficult. The evaluation generally comes from theoretical analysis, simulation or an actual implementation, according to criteria that can include utilization bounds, success rates, number of preemptions, migrations, and/or algorithm complexity.

Our long-term objective is to compare the various schedulers with ease while taking into account the capacity of the hardware architecture (e.g. caches, dynamic frequency scaling, or system overheads) to have an effect on their performance. This effect is currently very difficult to evaluate using theoretical analyses such as schedulability tests or resource augmentation. On the other hand, while using a real system would seem to be a better approach, the effective implementation of a scheduler as an operating system component requires a substantial amount of time and the results are too specific to the system. As a consequence, we think that simulation could be a good compromise to efficiently evaluate scheduling algorithms.

This paper deals with SimSo, our tool to simulate multiprocessor real-time schedulers and that aims at facilitating the design of experimental evaluations. In a prior publication, some design choices regarding the simulation kernel have been presented [7]. More recently, we showed using SimSo how the use of the WCET could bias the evaluation of scheduling algorithms and how the impact of the caches could be integrated in the simulation [8]. As a consequence, the concept of *execution time model* was introduced.

Contribution. This paper presents SimSo and the main novelties that now enable to conduct large scheduling evaluations using it. It is indeed possible to automate the simulation of scheduling algorithms from the generation of the systems to the collection of the resulting data. The main task generators are now included and the number of available schedulers increased from five to more than twenty-five. Our methodology to automate the evaluation of multiple scheduling algorithms is described through an example.

Paper organization. The remainder of this paper is organized as follows: in Section II, related work is summarized. Section III presents SimSo, and Section IV shows how it can be used through an example. Finally, Section V provides some concluding remarks and envisages future work.

II. Related Work

Our work addresses the evaluation of the performance of scheduling algorithms using empirical measures. Empirical evaluations of scheduling algorithms focus on the overheads involved in scheduling decisions. The main studied causes of overheads are context switches, preemptions, migrations and computational complexity. Two approaches are typically considered to evaluate them. The first one is based on measured performance on a real platform with a dedicated operating system, e.g. the experiments done with LITMUS^{RT} [4], an extension of the Linux Kernel developed at the University of North Carolina, or the experimental work of Lelli et al. [20] on a dedicated implementation of Linux with RM and EDF multiprocessor schedulers. This method could also be conducted on a cycle-accurate simulated architecture with a real operating system as in [31]. The second approach is to use tools dedicated to the simulation of real-time systems. Most of these tools are designed to validate, test and analyze systems. MAST [16] proposes a set of tools to model and analyze distributed real-time systems with, for instance, feasibility tests or sensitivity analyses. MAST also includes a simulator, JSimMAST. Cheddar [28] proposes a GUI comprising a simulator, many feasibility tests and it is also used to simulate AADL models. RTSIM [5] is a collection of programming libraries for the simulation of real-time control systems. It is used in particular for experimenting new scheduling algorithms. The last version was published in 2007.

STORM [30] and YARTISS [6] are the closest tools to what we aim. They offer a simulator to conduct evaluation on scheduling algorithms with the possibility to easily join new scheduling policies. However, due to its time triggered simulation engine, STORM does not provide an efficient way to model the unit of time below a tick of simulation which is a significant limitation for us. YARTISS is certainly the most suitable tool to evaluate scheduling algorithms by considering overheads or hardware effects. However, we began the implementation of our tool in 2011, before YARTISS was published. Moreover, its design is focused on the study of energy consumption and customizing it for our needs would have been difficult.

III. SimSo

To facilitate the experimentation of scheduling algorithms, we thus propose a dedicated tool: SimSo[1], a real-time scheduling simulator designed to be easy to use as well as extend. This software is freely available under an open source license.

The design of SimSo has been driven by the components available in real systems so that practical issues regarding the implementation can be taken into consideration. Such issues would have been hard or even impossible to integrate into theoretical studies.

A. Architecture

The core of SimSo relies on SimPy[2], a process-based discrete-event simulation framework. The use of discrete-event simulation allows it to deal with short and long durations at the same cost. Its process-based nature offers a convenient way to express the behavior of the simulated components.

The characteristics of a system are modeled by a *Configuration* object that contains all the information about the system (tasksets, processors, duration, scheduler, etc). This object provides some methods to configure the system but also to save it into an XML file.

Figure 1 shows the main classes of SimSo and their mutual interactions. The design of SimSo is inspired by real systems: there are processors, tasks, jobs, timers, etc. Each of these objects simulates the behavior of the corresponding part on the system: *Tasks* release the jobs; *Jobs* emulate the execution of the task's code; *Timers* can launch a method on a processor at a given time; etc. The instances of *Processors* are actually the central part of the simulation because they simulate both a processor and the operating system executing on it. Each processor can execute a job or be interrupted to execute a method of the scheduler. Finally, the *Scheduler* object is not an active process. It could be considered as a part of the operating system and as a consequence, its methods are only called by the *Processors*.

The *Model* object is the conductor of the simulation. It takes as a parameter the *Configuration* object. When the *run_model* method is called, the objects described above are created and launched.

[1]SimSo: http://homepages.laas.fr/mcheramy/simso/
[2]SimPy: http://simpy.readthedocs.org/

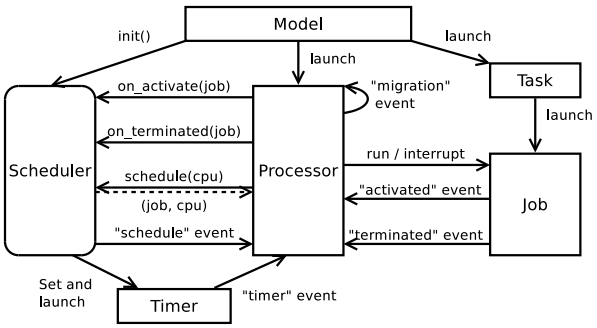

Fig. 1. Interactions between main class instances. *Processor*, *Task*, *Job* and *Timer* are *Process* objects and can have multiple instances.

The design of SimSo allows it to take into consideration various time overheads that occur during the life of the system. This includes direct overheads such as context-switches and scheduler calls (with fixed time penalties) but also indirect overheads with a simplified system of locks to forbid the parallel execution of a scheduler if needed. Such overheads are applied on the processor they are supposed to occur (e.g. the time spent in the scheduler is taken into account on the processor that called the scheduler).

We would also like to draw attention to the fact that the above-mentioned overheads only consume extra-time without changing the time used to execute the jobs. Indeed, as an example, these overheads do not take into account the possible cache misses that could slow down a job and increase its duration. This important aspect can also be taken into account by SimSo and is explained in section III-D.

B. Writing a Scheduler

The first requirement for the experimentation of a real-time scheduling policy is, undoubtedly, a way to specify the algorithm. This should be able to deal with any kind of online scheduler: global, partitioned, semi-partitioned, etc. Moreover, the implementation of a scheduler in a simulator should also be realistic in the sense that it should rely on mechanisms available on a real system. For instance, the choice of which processor should run the scheduler may have an impact on the performance or even the schedulability. Another example is the finite precision of the timers: this may introduce a tiny difference compared to the theoretical schedule and cause a major issue.

One of the advantages of using a simulator is to simplify the experimentation. Writing a scheduler should therefore be as easy as possible and rely on useful methods. We decided to use Python, a high-level language that benefits from a growing interest from the scientific community (e.g. the SciPy project). In practice, most of the schedulers that we have implemented contain less than 200 lines of code. The language is different to the one that would be used on a real implementation, however, this does not change the underlying algorithms and logic.

A scheduler for SimSo is a Python class that inherits from the *Scheduler* class and is loaded dynamically into the

simulator. The following methods must be implemented:

- **init:** The *init* method is called when the simulation starts, it is used to initialize the scheduler.
- **on_activate:** This method is called whenever a job is activated.
- **on_terminated:** This method is called when the execution of a job is done or when a job is aborted.
- **schedule:** This method returns the scheduling decisions. This method is called when a processor has been requested to take a scheduling decision. This request is usually done during a job activation, a job termination or by a timer.

As an example, figure 2 shows the source code of a global multiprocessor Earliest Deadline First scheduler[3].

```
from simso.core import Scheduler

class G_EDF(Scheduler):
    def init(self):
        self.ready_list = []

    def on_activate(self, job):
        self.ready_list.append(job)
        # Send a "schedule" event to the processor.
        job.cpu.resched()

    def on_terminated(self, job):
        # Send a "schedule" event to the processor.
        job.cpu.resched()

    def schedule(self, cpu):
        decision = None # No change.

        if self.ready_list:
            # Look for a free processor or the processor
            # running the job with the least priority.
            key = lambda x: (1 if not x.running else 0,
                             x.running.absolute_deadline if x.running else 0)
            cpu_min = max(self.processors, key=key)

            # Obtain the job with the highest priority within the ready list.
            job = min(self.ready_list, key=lambda x: x.absolute_deadline)

            # If the selected job has a higher priority
            # than the one running on the selected cpu:
            if (cpu_min.running is None or
                    cpu_min.running.absolute_deadline > job.absolute_deadline):
                self.ready_list.remove(job)
                if cpu_min.running:
                    self.ready_list.append(cpu_min.running)
                # Schedule job on cpu_min.
                decision = (job, cpu_min)

    return decision
```

Fig. 2. Code of a global multiprocessor Earliest Deadline First scheduler.

C. Available Schedulers

In order to check the ability to express a wide range of algorithms, we have already implemented more than 25 schedulers. The main uniprocessor schedulers, RM, DM, FP, EDF and M-LLF [24] are available. The DVFS schedulers Static-EDF and CC-EDF [25] are also available.

The library of schedulers provided with SimSo also includes a large variety of multiprocessor real-time scheduling algorithms, from partitioning to global ones.

The *partitioned* approach forbids migrations and necessitates a static allocation of the tasks to the processors. The schedulers P-EDF and P-RM are available (they use the Decreasing First-Fit assignment algorithm). Moreover, a dedicated class is provided in SimSo to offer the possibility to choose any uniprocessor scheduler and one of the available assignment algorithms (First-Fit, Next-Fit, Best-Fit, Worst-Fit, with or without an initial sorting). This class is intended to ease the development of a partitioned scheduler, but it is not mandatory.

On the other side, when migration is permitted, scheduling algorithms are referred to as *global*. A first category of global schedulers use a single list of active tasks and assign a priority to each task. For an architecture with m processors, the m jobs with the highest priority run in parallel. The following algorithms belonging to that category are available in SimSo: G-RM, G-EDF, G-FL [13], EDF-US [29], PriD [18], EDZL, M-LLF [24] and more recently U-EDF [22].

Baruah introduced the concept of fairness as a way to achieve optimality in terms of schedulability. SimSo provides such PFair schedulers with PD^2 and its work-conserving variant ER-PD^2 [1]. Subsequently, it was demonstrated that the fairness constraint could be released to only apply at the job boundaries and thus could reduce the number of preemptions and migrations. This led to the BFair and DP-Fair techniques. We have implemented such schedulers: LLREF [9], LRE-TL [15], DP-WRAP [21], BF [31] and NVNLF [14].

In order to reduce the number of migrations, some hybrid approaches, termed *semi-partitioned* approaches, combine the advantages of global and partitioned scheduling. At the present time, SimSo proposes three semi-partitioned schedulers: EDHS [19], EKG [2] and RUN [26].

D. Execution Time Model

When simulation is used to study the schedulability of a system, it is usual that the tasks meet their worst-case execution time at each job. However, the use of the WCET is in fact very pessimistic: the worst-case is an upper-bound that is hardly reached by the jobs, and it is even less likely that the jobs of all the tasks meet their WCET at the same time. As a consequence, we believe that the WCET approach should not be the only way to compare policies in terms of performance. It is non-realistic and gives an advantage to some scheduling policies that highly depend on the WCET. Relatedly, schedulers capable to take benefits from shorter computation times cannot be fairly evaluated. In [8], we give some experimental results that illustrate this fact.

Also, many scheduling evaluations only focus on the number of preemptions and migrations because they are the source of overheads. A preemption induces a system overhead due to the context-switching, but it may also increase the computation time of a job by causing extra cache misses. In fact, Mogul and Berg have shown that the Cache-Related Preemption Delays (CRPD) are more important than the system overheads. To increase realism, it is essential to integrate CRPD within the computation time of the jobs.

As a consequence of the two previous remarks, it is desirable to have the possibility to simulate a system with customized durations of jobs, depending on the purpose of the simulation. In SimSo this point is achieved with the *Execution Time Models* (ETM). An ETM is a class that determines the duration of the jobs during the simulation. Figure 3 shows

[3] A minor modification to this code would reduce the number of migrations by executing a job in the same processor than its previous execution.

the communication between a job and the ETM object (there is a single ETM object for all the jobs). The ETM object is informed by the jobs of any scheduling event. The job will use the *get_ret* method to get a lower bound of its remaining execution time and, when that time is up, the job calls that method again until it returns 0.

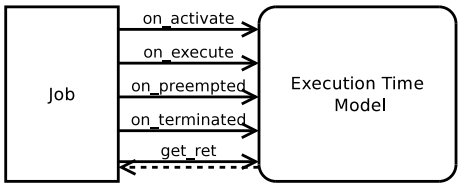

Fig. 3. Interface of any execution time model.

Several *Execution Time Models* are already available in SimSo. The simplest model consists of using the WCET of the tasks for their execution time. A second one uses a random duration for each job to meet a given average execution time (ACET). The ACET model uses a normal distribution defined by its mean, its standard deviation and is bounded by the WCET. Another model detects the preemptions and migrations and extends the WCET[4] of the job using fixed time penalties. Finally, a more complex model tries to simulate the state of the caches. In this latter model, the execution time of the jobs depends on the events that happen while they are active. This ETM is also interesting because it simulates the impact of shared caches and, as a consequence, it is impossible to know in advance when a job will end since it depends on external events.

These models can also deal with Dynamic Voltage and Frequency Scaling (DVFS). Indeed, when the speed of a processor is changed, the job that was running on it is preempted and resumed in order to inform the ETM and to reevaluate its remaining execution time. The current DVFS model simply considers that a job consumes its computation time proportionally to the speed of the processor. This is obviously a simplified assumption, but it is possible to implement more realistic ETM models to deal with DVFS.

Similarly, it should also be possible to add an energy consumption model.

E. Generation of Tasksets

A taskset is defined by the number of tasks, their utilization factor, periods, deadlines and the total utilization. Bini and Buttazzo showed how the random generation of the tasksets can bias the experimental results of some scheduling algorithms on uniprocessor [3].

For the multiprocessor case, several methods are used by the researchers to generate the tasksets. The most common algorithms are implemented in SimSo:

- Kato et al. use an approach inspired by the algorithm described by Ripoll et al. where tasks are appended to the taskset until the targeted total utilization is reached [19], [27]. The number of tasks is therefore variable.
- The algorithms *UUniFast-Discard* and *RandFixedSum* generate a taskset with a given number of tasks and a given total utilization [12]. At the present time, this methods seem to be the most efficient in generating tasksets with a weak bias.

These algorithms only generate a set of utilization rates and must thus be combined with a period generator. The following algorithms are made available in SimSo:

- *Uniform distributions in various fixed ranges:* Most evaluations use it and this is certainly an interesting way to study the influence of the periods, but it may not be relevant for realistic cases.
- *Log-uniform choice of periods [10]:* For a period range of 1-1000ms, the log-uniform distribution generates an equal number of tasks in each time band (1-10ms, 10-100ms, 100-1000ms) whereas a uniform distribution would generate 90% of the periods in the range 100-1000ms.
- *Random draw among a fixed set of values:* One could argue that in an industrial system, the periods are derived from the specifications, which are partly written by humans. Task periods are therefore more likely to be rounded.

Other period generators could also be added in the future. For instance, Goossens [17] suggested a method to reduce the hyper-period of the system by using periods that can be decomposed in a limited number of prime numbers.

F. Collecting Simulation Results

In order to evaluate scheduling algorithms, some data must be collected from the simulation. The literature proposes many measures, here is a non-exhaustive list of data that could be recovered:

Success rate: The ratio between the number of jobs that have exceeded their deadline and the number of jobs. It gives a performance indicator on the schedulability of a taskset.

Preemptions and migrations: Preemptions and migrations are a factor of overhead and many recent schedulers are focusing on their reduction. A distinction is made between job migration and task migration since they may have not the same implications.

Scheduler calls: The algorithm of a scheduler requires some time to determine which jobs should run on the processors. Some scheduling policies are known to make many scheduling decisions, and some require a significant amount of time to compute. Therefore, it is interesting to keep track of the number of calls to the various methods of the scheduler.

Normalized laxity: Lelli et al. proposed to measure the performance of a scheduler by computing the normalized laxity [20]. The laxity of a job is its relative deadline minus its response-time. The laxity of each job of each task is divided by the task period in order to obtain a normalized laxity. A greater normalized laxity is synonym of a better safety and better reactivity.

[4]In this case, the WCET is defined as the worst-case execution time without any interruption.

During the simulation of a system with SimSo, every significant events are traced. At the end of the simulation, a *Results* object is built to store these events and could be post-treated to compute measurements. Whereas this approach is actually heavier than just counting events such as the preemptions and migrations during the simulation, this provides more flexibility. Indeed, it is not necessary to modify the code of the simulator to add the computation of new measurements one did not think about. A set of methods are also available to ease the retrieval of usual metrics such as the ones mentioned above.

SimSo provides a graphical user interface that helps to configure a system and run it. That GUI is capable of displaying common measures such as preemptions, migrations, or execution times. It is also possible to display a gantt chart, which is very useful during the development of a scheduler. However, this GUI only shows the results for a single simulation.

G. Conducting an Evaluation Campaign

To conduct a large evaluation campaign, it is possible to use SimSo as a Python module. This way, a Python script can be written to automate the creation of systems, their simulation and the collection of the results. This choice was motivated by the fact that the studies can be very specific and a graphical user interface would be necessarily too frozen or too complex. On the other hand, using a script is much more flexible. Everything that is possible using the graphical user interface is also possible from a script.

IV. Example

This section illustrates the use of SimSo in conducting an experiment on scheduling policies. SimSo is used as a module for a Python script and the steps described below have been programmed.

This experiment focuses on the number of preemptions and migrations in function of the number of tasks, for various numbers of processors and load. The objective is to compare five schedulers: G-EDF, NVNLF, EKG[5], RUN and U-EDF.

A. Generation of the Configurations

The first step is to define the characteristics of the simulated systems. For this example, we have selected the following parameters:

- Number of tasks: 20, 30, 40, 50, 60, 70, 80, 90, 100
- Number of processors: 2, 4, 8
- System utilization: 85%, 95%

For each configuration (tasks, processors, utilization), twenty tasksets are generated using the methods offered by SimSo, leading to a total of 5400 systems ($9 \times 3 \times 2 \times 20 \times 5$). The RandFixedSum algorithm was used to determine the task utilizations and the periods were chosen randomly within a log-uniform distribution between 2 and 100 ms. The ACET Execution Time Model is used and, for each task, the expected value is set to 75% of the WCET and the standard deviation to 10% of the WCET. Each system is simulated on the interval of time 0-1000ms[6].

The *Configuration* objects were saved into XML files for potential reuse (it is interesting to repeat simulations on systems with atypical results in order to obtain a better understanding.).

B. Simulation and Collection of the Results

SimSo executed 5400 simulations which took approximatively 2 hours on an Intel Core i7 processor.

When a simulation is done, the number of preemptions and job migrations are extracted from the *Results* object built by the *Model* object. Preemptions caused by the system (e.g. the scheduler is called but no decision is taken) are not taken into account.

In order to facilitate the analysis, we stored the data in an SQLite3 database.

C. Analysis

From that database, another script draws the charts using matplotlib, a plotting library for Python. Each point is the mean of the twenty tasksets sharing the same parameters. The results for 8 processors and a system utilization of 95% are shown on Figure 4.

A few comments on the results are provided here as a complement to the figure. EKG generates a lot of migrations that could be easily avoided with a better choice of the parameter K or other improvements [23]. The results for NVNLF are getting better with more processors unlike the others. U-EDF could probably do better combined with clustering. With more than 20 tasks, RUN acts as a partitioned scheduler most of the time. G-EDF provides better results in terms of preemptions and migrations but a few jobs were aborted as a consequence of deadline misses. U-EDF and RUN could probably catch up with G-EDF with a work-conserving variant.

V. Conclusion

In this paper, we have presented SimSo, a simulation tool to evaluate the multiprocessor schedulers. Its objective is to facilitate the comparison of the numerous scheduling policies. To this end, we will conduct large campaigns of experiments with many scheduling algorithms using the same tasksets. This should allow us to reproduce numerous experiments in order to confirm or invalidate results. At the present time, more than twenty-five schedulers are available, showing that SimSo is capable of handling partitioned, global and hybrid scheduling approaches.

The architecture of SimSo, in particular the scheduling interface, was briefly explained. Particular care has been taken to keep a realistic scheduling interface so that practical decisions are not eluded. This has also enabled SimSo to take into consideration direct overheads such as the context-switches or scheduling decisions. Moreover, the computation time of the jobs is determined by a model that can be selected depending on the purpose of the simulation. Hence, the computation time

[5]The parameter K has been set to the number of processors.

[6]Unfortunately, the hyper-period for a set of 100 tasks with random periods is far too long to be considered (in years).

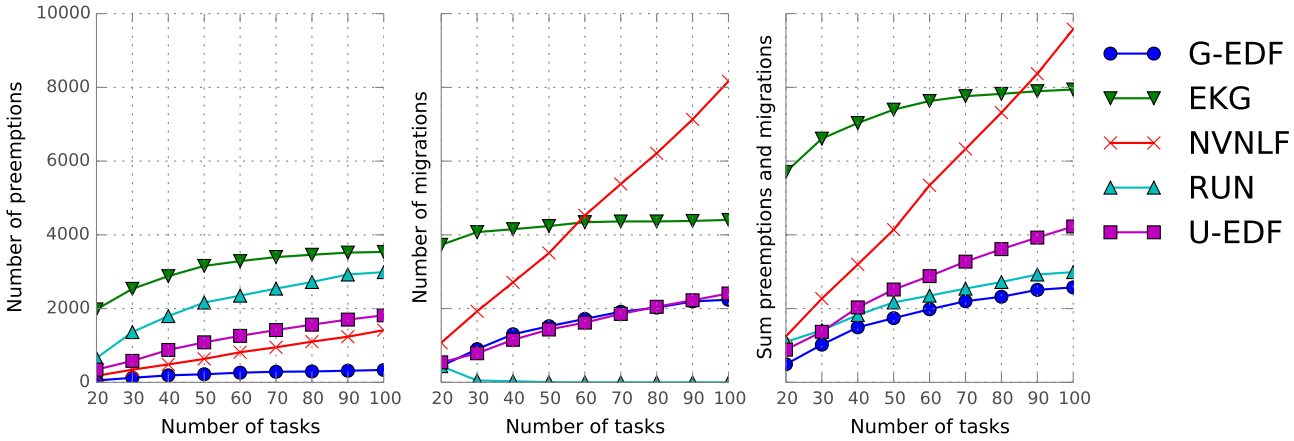

Fig. 4. Number of preemptions and migrations for a system with 8 processors and a (worst-case) total utilization of 95%. The simulation used random durations for the job computation time.

of a job can either be a static duration, a random duration, or even take into account cache-related preemption delays. Additionally, a small example shows the capability of SimSo to produce concrete results.

Future work includes an improvement of SimSo by introducing cache interferences in the simulation and introducing more complex task behaviors such as shared resources and precedence relations.

Acknowledgment

The work presented in this paper was conducted under the research project RESPECTED (http://anr-respected.laas.fr/) which is supported by the French National Agency for Research (ANR), program ARPEGE.

References

[1] J. Anderson and A. Srinivasan, "Early-release fair scheduling," in *Proc. of ECRTS '00*, 2000.

[2] B. Andersson and E. Tovar, "Multiprocessor scheduling with few preemptions," in *Proc. of RTCSA*, 2006.

[3] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, no. 1-2, 2005.

[4] J. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. Anderson, "LITMUS^{RT}: A testbed for empirically comparing real-time multiprocessor schedulers," in *Proc. of RTSS*, 2006.

[5] A. Casile, G. Buttazzo, G. Lamastra, and G. Lipari, "Simulation and tracing of hybrid task sets on distributed systems," in *Proc. of RTCSA*, 1998.

[6] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, and M. Qamhieh, "YARTISS: A Tool to Visualize, Test, Compare and Evaluate Real-Time Scheduling Algorithms," in *Proc. of WATERS*, 2012.

[7] M. Chéramy, A.-M. Déplanche, and P.-E. Hladik, "Simulation of real-time multiprocessor scheduling with overheads," in *Proc. of SIMUL-TECH*, 2013.

[8] M. Chéramy, P.-E. Hladik, A.-M. Déplanche, and S. Dubé, "Simulation of real-time scheduling with various execution time models," in *Proc. of the WiP session of SIES*, 2014.

[9] H. Cho, B. Ravindran, and E. Jensen, "An optimal real-time scheduling algorithm for multiprocessors," in *Proc. of RTSS*, 2006.

[10] R. Davis and A. Burns, "Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," in *Proc. of RTSS*, 2009.

[11] ——, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, 2011.

[12] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *Proc. of WATERS*, 2010.

[13] J. Erickson and J. Anderson, "Fair Lateness Scheduling: Reducing Maximum Lateness in G-EDF-Like Scheduling," in *Proc. of ECRTS '12*, 2012.

[14] K. Funaoka, S. Kato, and N. Yamasaki, "Work-conserving optimal real-time scheduling on multiprocessors," in *Proc. of ECRTS '08*, 2008.

[15] S. Funk and V. Nanadur, "LRE-TL: An Optimal Multiprocessor Scheduling Algorithm for Sporadic Task Sets," in *Proc. of RTNS*, 2009.

[16] M. Gonzalez Harbour, J. Gutierrez Garcia, J. Palencia Gutierrez, and J. Drake Moyano, "MAST: Modeling and analysis suite for real time applications," in *Proc. of ECRTS '01*, 2001.

[17] J. Goossens and C. Macq, "Limitation of the hyper-period in real-time periodic task set generation," in *Proc. of the 9th International Conference on Real-Time Systems (RTS)*, 2001.

[18] J. Goossens, S. Funk, and S. Baruah, "Priority-driven scheduling of periodic task systems on multiprocessors," *Real-Time Systems*, vol. 25, no. 2-3, 2003.

[19] S. Kato and N. Yamasaki, "Portioned EDF-based scheduling on multiprocessors," in *Proc. of EMSOFT*, 2008.

[20] J. Lelli, D. Faggioli, T. Cucinotta, and G. Lipari, "An experimental comparison of different real-time schedulers on multicore systems," *Journal of Systems and Software*, vol. 85, no. 10, 2012.

[21] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, "DP-FAIR: A Simple Model for Understanding Optimal Multiprocessor Scheduling," in *Proc. of ECRTS '10*, 2010.

[22] G. Nelissen, V. Berten, V. Nelis, J. Goossens, and D. Milojevic, "U-EDF: An Unfair But Optimal Multiprocessor Scheduling Algorithm for Sporadic Tasks," in *Proc of ECRTS '12*, July 2012.

[23] G. Nelissen, S. Funk, and J. Goossens, "Reducing Preemptions and Migrations in EKG," in *Proc. of RTCSA*, Aug 2012.

[24] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in *Proc. of RTCSA*, 1998.

[25] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proc. of SOSP '01*, 2001.

[26] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, "RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor," in *Proc. of RTSS*, 2011.

[27] I. Ripoll, A. Crespo, and A. Mok, "Improvement in feasibility testing for real-time tasks," *Real-Time Systems*, vol. 11, no. 1, 1996.

[28] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: A flexible real time scheduling framework," *Ada Lett.*, vol. XXIV, no. 4, 2004.

[29] A. Srinivasan and S. Baruah, "Deadline-based scheduling of periodic task systems on multiprocessors," *Inf. Process. Lett.*, vol. 84, no. 2, 2002.

[30] R. Urunuela, A.-M. Déplanche, and Y. Trinquet, "STORM a simulation tool for real-time multiprocessor scheduling evaluation," in *Proc. of ETFA*, 2010.

[31] D. Zhu, D. Mosse, and R. Melhem, "Multiple-resource periodic scheduling problem: how much fairness is necessary?" in *Proc. of RTSS*, 2003.

Simulating real-time and embedded networks scheduling scenarios with ARTEMIS

Olivier Cros, Laurent George, Frédéric Fauberteau, Xiaoting Li
ECE Paris
37, quai de Grenelle
75015 Paris, France
Email: {cros, lgeorge, fauberte, xiali}@ece.fr

***Abstract*—Real-time industrial domains are subject to strong constraints in terms of performance and reliability that directly increase the costs of their infrastructures. In order to build these infrastructures and to test them, we propose to implement ARTEMIS: Another Real-Time Engine for Message-Issued Simulation. Its aim is to manage all real-time networks like CAN or AFDX and to simulate their behaviors in terms of scheduling and performance delay. To implement this tool, we use a modular way of development, building modules on a two-parts kernel. This architecture allows our software to be generic. Moreover, many interfaces can be easily integrated for several network implementations.**

I. Introduction

A. Network Infrastructure

Industrial domains such as aircraft or public transports rely on real-time networks to transmit their data. For requirements like ergonomics, comfort or even mechanics, various constraints are necessary in terms of reliability, safety and performances. Respecting these constraints and testing them imply to build high-efficiency network infrastructures. But real-time industrial networks involve very high timing requirements that make their infrastructures very expensive in terms of time, money, materials and human resources. Hence these infrastructures have to be tested by simulation and improved before being implemented. The review of this analysis justifies our goals to develop tools to make building of this test plans easier. Here, we propose a simulation software to model and simulate behaviors on industrial networks infrastructures.

B. Functional Needs

Developing a network simulation tool from scratch is a costly work that needs to be strongly justified. That is why the first point before conception is too focus on the functional needs we want our tool to complete, and to search for already existing tools to check if this needs are not already centralized in a tool.

The first, main objective of ARTEMIS is to be easy to use, and to adapt itself to all kind of profiles : engineers and developers, but also to industrials or reasearchers not necessarily specialized in computer science. So, ergonomy and easyness of configuration is a central point to focus on.

Moreover, we propose a list of functional requirements according to the review of these different simulation approaches. The main points we can emphasize are:

- ARTEMIS is as easy as possible to install independently of the operating system,
- ARTEMIS is flexible to take into account any scheduling context of network messages,
- ARTEMIS is free for research activities and accessibility,
- ARTEMIS is open-source in order to easily implement new modules and to make it well known by the real-time community. Moreover, it allows a larger point of view in terms of development standards and norms to respect,
- In order to keep the genericness in the development and to accept add-ons from any development team, ARTEMIS is extremely modular and respects standard protocols of data model.

C. Related Work

We can notice that many simulators already exist but each one is oriented to a specific context. As a first general classification, we can split all the real-time scheduling simulators in two wide parts : processor-based and network-based.

The processor ones are schedulers which are used in a static machine context, in order to focus on how to schedule a set of given tasks in a computing machine. The network-based simulators are built to focus on message scheduling through several nodes in a network(Ethernet, AFDX [1], CAN, ...). They are either generic or specific to a network architecture. We present here a review of the main simulation softwares used these two contexts:

- *Network Simulator 3* (ns-3) [2] is one of the main open-source network simulator designed for performance evaluation in an industry-based context. Working on virtual devices which allow to model nodes and links, ns-3 is often used as a testing framework.
- OMNET++ [3] is another industrial network simulator for networks adapted to specific industrial needs. Given a functional requirement, OMNET++ [4] allows to adapt the size of a network and to simulate an infrastructure with a precise number of nodes, communication links and peripherals.
- Cheddar [5] and FORTAS [6] are uniprocessor and multiprocessor open-source simulators. They are simulation tools which are the closest ones to the model

we want to use. Unfortunately, they do not provide network implementation we need to simulate tasks and messages in AFDX, CAN or in any other network architecture.

- SetSIM [4] is a mathematical simulator designed to modernise switched Ethernet networks and able to manage many different protocols like FTT-SE and POWERLINK [7]. The cluster-based architecture it provides allows to have either a centralized scheduling policy or a totally distributed one with each node applying its own independent policy.

Of course, we just present here a part of all real-time simulators that exists, and we could also have mentioned others like Rapitime, or either real-time operating systems like RTLinux ou PikeOS [8]. The point is that each of these softwares is designed for a specific kind of requirement : industrial, simulation, dimensioning, ... But alas, each one of these includes major lacks of functionalities or distorsion in the employed approach that implies us to build our own tool : closed sources, lack of modularity and interoperability through different operating systems, difficulties to implement network architectures like AFDX... The main aim of our work was to design a tool to cover all the previously mentioned requirements and to solve these problems.

D. Development guidelines

The point is to make ARTEMIS a powerful and generic tool, but to implement a viable software, we made some decisions concerning its development :

- ARTEMIS is designed to be used before an industrial implementation, to compute the size of a network infrastructure
- ARTEMIS is designed as a simulation tool first, for reasearch and industrial activities in terms of real-time and mixed-criticality
- ARTEMIS is wanted to be generic and to be able to communicate and send frames to real networks.

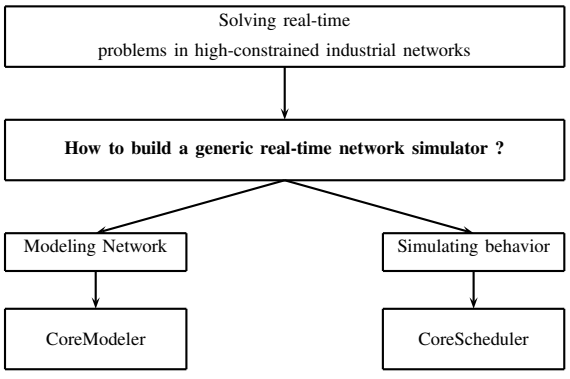

Fig. 1. Design of the ARTEMIS kernel.

II. Main modules

A. Java Structure

In order to implement the strong modularity and to keep a very high portability, we have chosen to implement ARTEMIS with an object language. Portability and free open-source approach automatically excludes corporate languages like C#, and leaves the classical object-languages: C++, Python, Java. We chose Java for its portability and its high compatibility with web-oriented languages.

One of the main points in the development of ARTEMIS was to skip its needs in terms of setup and configuration. To implement this, we needed to model its graphical interface through a comfortable and ergonomic set of development-tools, that is why we decided to implement its graphical interface in web-oriented languages(PHP/HTML/JS). So, to keep high-interactivity with this graphical module, we decided to implement ARTEMIS's kernel in Java.

B. Modular system

The main objective in the development of ARTEMIS is to allow different teams, perhaps without any relation between them, to work on the same tool. That is why the points we needed to specify were the architecture of the software and the standards of intern communication between project parts. We decided to work on a very high-modular approach, with auto-generated specified XML files between modules. We built a complete simulation kernel, which is the center of the software and implements the scheduling and modeling (see Figure 3) and works totally independently from other parts with auto-generated XML files.

The topology of the network, as its behavior during simulation, is represented by a set of XML standardized-files, and that is this XML standards we need to make the whole project reliable.

First, the XML entry file of the kernel, which represents all the network topology and structure (with nodes, links and messages) needed to be specified properly. We decided to represent the network as a set of nodes, each node containing (optionally): links to other machines, messages to generate. A link to another machine symbolizes a direct network connection: if two machines share the same link, they are physically binded (by a wire, for example). A message is always generated by a machine and this machine represents its entry point in the network.

Each message is represented by a 4-tuple $(\mathcal{P}_i, T_i, W_i, O_i)$ as described bellow.

```
<message id="1" destination="4">
        <crit lvl="0">
                <path>1,3,4</path>
                <period>6</period>
                <offset>1</offset>
                <wcet>4</wcet>
        </crit>
</message>
```

Fig. 2. XML representation of a message of source node 1 and destination node 4.

Conceiving a simulator with a standardization approach implies to use a modular paradigm. Indeed, ARTEMIS is developed not as a global simulator software but as a set of

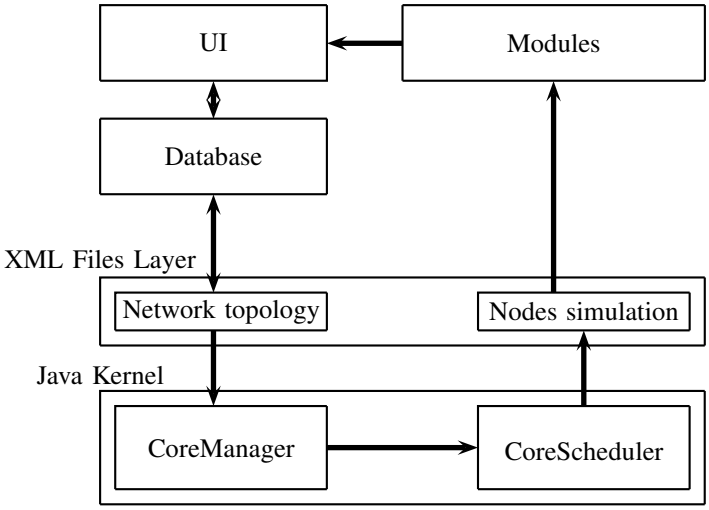

Fig. 3. Modules architecture for ARTEMIS

plug-ins connected to a central simulation kernel. First, we want to focus on this simulation kernel.

It is splitted in two different submodules. Firstly, the CoreManager is a modelization submodule that represents and builds an entire set of objects to model a network topology. Secondly, the CoreScheduler is a running-simulator which dynamically simulates the behavior of the network at each time-cycle.

C. Modelization Kernel

In the modelization module, an XML entry file represents the network topology (this file is generated by the graphical interface of ARTEMIS). Starting from it, the module builds a set of objects for each component in the network. In our network, there is three main kinds of components: nodes, messages and links:

- the nodes represent the network operators (switches, routers, machines, *etc.*). In order to keep the maximum genericness, we groupes all kind of operators behind a common concept : a node. Each operator is represented as a node object and is characterized by a set of input and output buffers. With this approach, we can really simulate the implementation of a network port. For example, in the specific case of switched Ethernet, this allows us to simulate an IP network architecture.
- the links represent the bridges of the network: two machines sharing the same link can directly send informations to each other.
- the messages are the informations sent through the network. Each message is characterized by a sender which is the machine where the messages arrives in the network and by a receiver which is its final destination in the network. In the XML entry file, the path of message is defined statically. Indeed, in order to assure the determinism of a network architecture, defining static network path (and, so, a computable end-to-end delay) is a unavoidable constraint.

As in classical scheduling problems, each message of index i in the network is first characterized by a vector of three components (T_i, W_i, O_i) where T_i is its period (the minimum inter-arrival time of two consecutive messages), W_i is its Worst-Case Transmission Time (WCTT) and O_i is its offset (the activation time on the first node). As we consider also periodic and non-periodic messages, we adapted this model to software needs with defining the following hypothesis : a non-periodic message is a message with an infinite period. So, to represent a one-shot message, we just need to put a message's period to a limit far beyond the simulation time limit.

In ARTEMIS, we add one element to this vector : the path. So then each message of index i is then characterized by a vector of four components $(\mathcal{P}_i, T_i, W_i, O_i)$ where $\mathcal{P}_i$ is the path of the message (starting from the origin machine identifier to destination machine identifier) and the three last components correspond to the previous definition.

D. Simulation Kernel

The simulation module of the kernel is built to implement runtime simulation of the topology by using the multi-threading paradigm. To respect our modular approach, we have implemented this module as two main tools: the *manager* and the *scheduler*. Thus they provide a clear separation between two aspects of the network simulation. The first one consists in the real-time approaches with scheduling requirements (scheduling policies, scheduling analysis [9], timing constraints, *etc.*). The second one corresponds to the network approaches (physical medium implementation, performance problems, *etc.*).

First, the role of the manager is to build a representation of the network components (machines, links, nodes, *etc.*). It has the responsibility to ensure compliance with the link architecture. Indeed, all components have their own manager which builds a structure among them, and then gives it to a global manager. Then, the manager part is a static modeler built to represent a network topology. It is split in different sub-managers, each one dedicated to one type of component: node, link, message. Then, the role of the scheduler is to dynamically simulate the network behavior. It creates a time loop and, for each time, the scheduler makes each node generate, analyze and send the messages through the network(see Algorithm 1).

E. Grapher and Calculator

The grapher and the calculator are two modules which are parts of the kernel but they operate independently. Indeed, in order to exploit the results of the kernel and to make them understandable for users, we need to transform XML files to graphs and to compute end-to-end delays according to node behaviors.

The kernel builds a XML file per node to represent their behavior over time. Using data contained in these files, the grapher builds for each node an image representation of the node state evolution and the currently analyzed message at each time of the simulation.

The calculator is a timing analysis module which computes the end-to-end delay of a given packet from the XML files according to different methods: holistic approach [10], trajectory approach [9].

F. User Interface

The purpose of the user interface is to be very intuitive and easy-to-use. Indeed, the user should be able to build and implement a network topology as fast as possible. In order to respect this approach, we decided to implement a web interface linked to a MySQL database. This way, there is no constraints in terms of installation or configuration (as far as a server is available). The user interface provides two modes: one drag-and-drop mode to quickly build simple topologies without any particular configuration and a more detailed mode to accurately configure the topologies.

In this two modes, the user just adds components (nodes, links and messages) to a central webpage which builds and displays automatically the global topology. Then, the user can configure each component separately (modify the WCTT of a message, or the scheduling policy of a node). Since all settings have a default value, the user can acquaint itself with the simulator without the preliminary setting up of a complicated configuration.

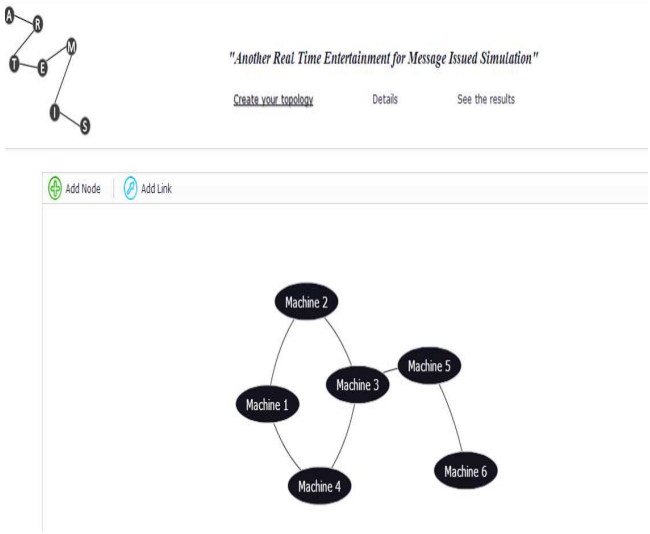

Fig. 4. ARTEMIS User Interface

III. Architecture

A. Path management

Basically, as we are working in a real-time context, the network has to be deterministic, and so to bound end-to-end message delays. One condition to assure this determinism is to manually define each message's path directly via the user interface(UI). So, through the UI, the user can define each message's path through the network, starting from its entry point to its output point.

For ergonomic issues, we built a static Dijkstra calculator(which we integrated in the UI) to allow a default trajectory computation for each message. But this method is just for ergonomic purposes, and does not correspond to an effective dynamic trajectory computation in the network architectures.

B. Scheduling policies

Each node is associated to a specified Java class designed to implement a scheduling policy. Given this, we implement a decentralized scheduling policy: each node has a default policy, which is the one adopted by the main manager, and can implement its own (determined by the user, or at least by a configuration file) at launch time. With this architecture, we decided first to implement the main current scheduling policies of real-time domain:

- First In First Out (FIFO) [11], [12], analyzes and sends the messages in the order they arrived in the input buffers.
- Fixed Priority (FP) [9], assigns a static priority to each message and always analyze the one with the higher priority first.
- FP/FIFO, messages are at first scheduled according to their fixed priorities, and messages with the same fixed priority are then scheduled FIFO.

These policies are given as examples and we can implement any other scheduling policy (EDF [13], [14], FP-RM, FP-DM, *etc.*) in ARTEMIS.

IV. Network Packets Scheduling

A. Applying Scheduling Policy to Nodes

Scheduling in ARTEMIS is split into two main procedures: analyzing and sending. Analyzing consists, for a node, of reading and treating the bytes one by one. In our scheduling, it is the packet transmission time consumption. The second step, sending, is putting a message on the bound of a link to allow the scheduler to transmit it to the node on the other bound.

Each node in ARTEMIS can adopt its own scheduling policy, or adopt the common one given by the CoreManager. Each node applies its scheduling policy to all its input buffers (maximum 500), and given this policy takes the messages one by one and sends it to an output link.

At each time, the scheduler pushes messages in output ports to the input port on the other end of the link. By default, we consider that passing through a link between two nodes takes no time, which means that the network flow is infinite. In real cases, the flow is not infinite, so the latency needed to pass through a link is not null. So, this transmission latency can be modified in the scheduler configuration to simulate a specific flow between two nodes. This latency can be set generally (same for each link) or independently.

At each time, then, the scheduler scans each machine and decides:

- if a message is already analyzed by the machine and its execution time is smaller than its WCTT, then we continue to analyze it,
- if there is no message or it is the end of transmission, the scheduler puts the message in an output port and selects a new message from input according to the scheduling policy.

Indeed, after a modelization cycle by the CoreManager, the scheduling simulation of the network occurs in two times. First, we need to schedule the behavior of each node : either it is an entry point in the network for a message, and then it is

```
Time_sim = 0;
N = {Node_1, Node_2, ..., Node_n};
N_copy = N;
for Time_sim from 0 to limit_time do
    msg = null;
    while N_copy is not empty do
        currentMachine = getNextMachine(N_copy);
        if currentMachine is not busy then
            msg = currentMachine.input.next();
        else
            Keep previous msg;
        end
        if msg ≠ null then
            if analyze(msg) ended then
                StoreToOutputBuffer(msg);
                Mark currentMachine as not busy;
            else
                Mark currentMachine as busy;
            end
        else
            Mark currentMachine as not busy;
        end
        N_copy = N_copy − currentMachine ;
    end
    for currentMachine ∈ N do
        sendMessages(currentMachine);
    end
end
```

Algorithm 1: ARTEMIS Scheduling algorithm

considered as a message generator, either it contains messages in its input ports, and then needs to get it. This first step of the scheduling is then to get these non-treated messages and to put it in the node. Then we analyze it(during a time equal to the WCTT of the message) and, finally, we put the message in an output port.

The second part is contained in a second loop and consists on moving all messages on output ports to input ports on the other side of the link. In order to keep the scheduling clear and to avoid confusions, we can't merge these two loops. So, we can conclude that simulating the network behavior consists on : simulating the nodes behavior, and transmitting the messages through the nodes.

B. Sending Packets

Sending packets in ARTEMIS consists of sending a packet from a node in its path to the next one. For this, we use a special object in our core : the link object. An object link consists of an association between two network addresses, each one linked to a different node. When a packet's analysis finishes on a node, the scheduler proceeds in a 3-step phase : First, the scheduler puts the packet from node's core to an output buffer. It symbolizes the end of its WCTT. Then, periodically, all output buffers are emptied and all packets in it are attached to links corresponding to the next machine in their path. At this time, packets are no more attached to any machine. Finally, a packet is put from the beginning of a link (which corresponds to a network address) to the end of it (which corresponds to the next address in its path).

In real cases, transmission of packets through links takes time, that is why we need to consider a link object as a kind of node, which possess its own treatment time, but without any queuing system or scheduling policies. Each packet sent through a link is immediately taken into account and transmitted.

V. Interfacing with real networks

A. Address Management

One of the main points of ARTEMIS in the context of industrial simulation is its ability to simulate a real network architecture, and to interface it with a real network interface of a given machine. For example, with a simulation server, we can emulate a given switched Ethernet network, and generate or get packets from an external network. To do this, we just need to define some nodes as input or output points of the network (see 5).

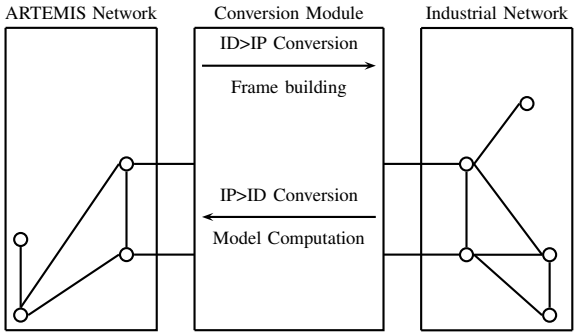

Fig. 5. Communication between ARTEMIS and real networks

To do this interfacing, ARTEMIS must implement an address management architecture to implement a real network behavior. That implies to include two main points. The first one is a network address manager based on IP layer to make our virtual network able to communicate with real infrastructure. The second one is a real message builder which represents messages, not only as a 4-tuples, but also as an array of bytes, composed according to specific protocols (especially Ethernet).

B. Packet modelization

To communicate with real network nodes with our virtually-simulated network, we have to decide which network protocols to implement in our software. First, this kind of modelization has to be implemented in a separate module: added to the kernel, the objective of this module is to transform virtually-simulated messages into real structured Ethernet frames (in case of Ethernet protocol). To do this, we need to compute the Ethernet packets header to a data manager which allows us to build messages as real network frames.

VI. Experimental results

In order to show the simulation results we can manage with ARTEMIS, we decided to model a simple network (see Figure 6) and to test it according to specific parameters:

- all the network is on FIFO policy, single-criticality mode,
- we did simulate the behavior of the network during 20 time-cycles,
- our network is composed of two messages M_1 and M_2 (see Fig 6).

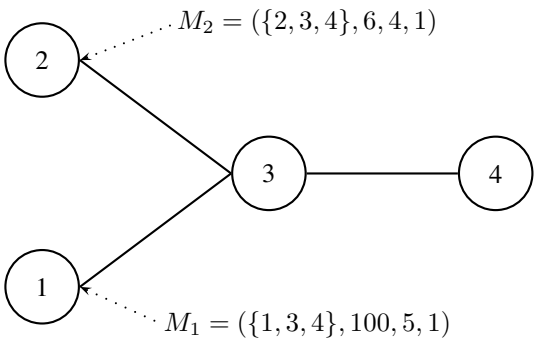

Fig. 6. ARTEMIS network simulation

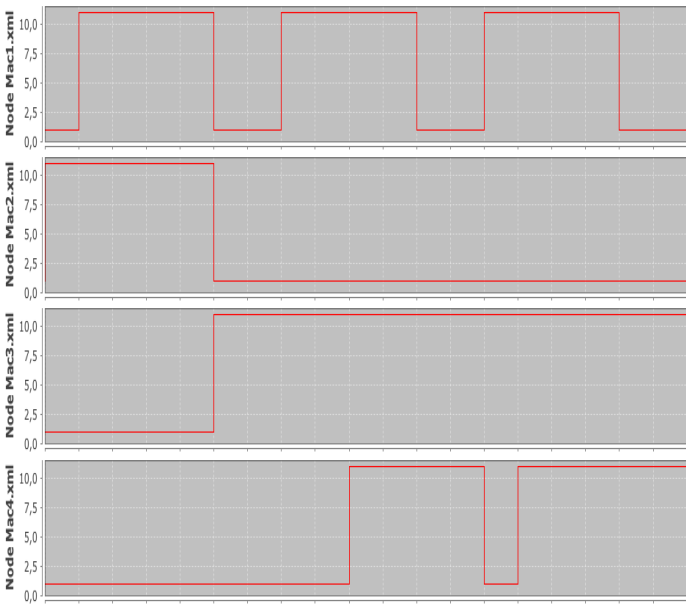

Fig. 7. Network simulation with ARTEMIS

The results computed by ARTEMIS are shown in Figure 7. Indeed, we have a generated png histogram for each node in the network (based on a separate XML file) which represents the behavior of each node during the simulation.

We can see, for each node(1, 2, 3, 4) that the obtained simulated behavior is conform to the hand-built mathematical theory. So, for this small topology, we simulated it correctly in an execution time lower than 15 seconds.

VII. Conclusion

To conclude, ARTEMIS is a generic real-time network simulation tool designed to mix network performance computation and industrial architecture dimensioning. Its modular conception based on XML files allows it to communicate and interface itself with many plug-ins, not necessarily implemented in Java. It is conceived to reduce the need of long configuration and installation difficulties and to centralize many previous simulation approaches in one central software.

As it is mainly designed for network simulation, the point is to manage concepts like mixed-criticality, real networks communication, Ethernet frame generations and many scheduling policies (FP, EDF, *etc.*) with it in order to compare their efficiency on the same topology.

As a future work, we want to focus on building new modules for ARTEMIS, connecting it to other languages and systems. The point would be also to study its genericness by testing simulation scenarios in uniprocessor and multiprocessor contexts.

In order to keep it usable and performing, we also want to focus on the execution time of simulations to improve the efficiency of our time-based simulation model.

References

[1] *ARINC 664*, ACCE Std. 664, 2002-2008.

[2] M. L. T. R. Anderson and G. F. Riley, "Network simulations with the ns-3 simulator."

[3] A. Varga, "The omnet++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*.

[4] T. M. Ashjaei, M. Behnam, "Setsim: A modular simulation tool for switched ethernet networks," in *WATERS-2013*.

[5] L. N. F. Singhoff, J. Legrand and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *International conference on Ada, 2004*.

[6] P. Courbin and L. George, "Fortas : Framework for real-time analysis and simulation," in *2nd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems.*

[7] L. A. R. Marau and P. Pedreiras, "Enhancing real-time communication over cots ethernet switches," in *Workshop on Factory Communication Systems 2008.*

[8] R. Kaiser and S. Wagner, "Evolution of the pikeos microkernel," in *First International Workshop on Microkernels for Embedded Systems*.

[9] S. Martin, P. Minet, and L. George, "End-to-end response time with fixed priority scheduling: Trajectory approach versus holistic approach," *International Journal of Communication Systems*, vol. 18, no. 1, pp. 37–56, 2005.

[10] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 40, no. 2–3, pp. 117 – 134, 1994.

[11] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "Optimistic problem in the trajectory approach in FIFO context," in *Proc. IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2013, pp. 1–8.

[12] S. Martin and P. Minet, "Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class," in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, Apr. 2006, pp. 8–pp.

[13] J. Rivas, J. Gutierrez, J. Palencia, and M. Harbour, "Schedulability analysis and optimization of heterogeneous EDF and FP distributed real-time systems," in *Proc. 23rd Euromicro Conference on Real-Time Systems (ECRTS)*, July 2011, pp. 195–204.

[14] S. Martin, P. Minet, and L. George, "The Trajectory approach for the end-to-end response times with non-preemptive FP/EDF*," in *Software Engineering Research and Applications*. Springer, 2006, pp. 229–247.

Applying Holistic Schedulability Tests to Industrial Systems: Experience and Lessons Learned

Shuai Li*†, Stéphane Rubini†, Frank Singhoff†, Michel Bourdellès*
*Thales Communications & Security, 4 av. des Louvresses, 92622 Gennevilliers, France
*Email: {first-name}.{last-name}@fr.thalesgroup.com
†Lab-STICC/UMR 6285, UBO, UEB, 20 av. Le Gorgeu, 29200 Brest, France
†Email: {last-name}@univ-brest.fr

***Abstract*—Several holistic schedulability tests exist in the literature, but they are not always used in the industry. One possibility to increase the usability of such tests, by system designers, is to implement them in scheduling analysis tools. This paper shows an experience on applying holistic schedulability tests to an industrial TDMA software radio protocol from Thales, by implementing the tests in Cheddar, a scheduling analysis tool. Our experience learned through this experience shows advantages and issues of applying such tests in the industry.**

I. Introduction

A Real Time Embedded System (RTES) has limited resources (e.g. processors) and processing depends on time. Most RTES have several concurrent tasks, with deadlines, scheduled on processors. One aspect of designing a RTES is to verify that task deadlines will be met, under a specific scheduler; otherwise said that the system is schedulable. This can be done through schedulability tests [20].

To apply a particular schedulability test, a system is abstracted with a specific *task model*. Two of the first proposed schedulability tests are the Liu and Layland test [11], based on processor utilization, and the Joseph and Pandya test [6], based on response times of tasks (i.e. time from release to completion). Both of these tests are applied to the periodic task model with constrained deadlines (less than period). Since then, several schedulability tests have been proposed for different task models, including models to describe multiprocessor partitioned systems (communicating tasks allocated on processors in a network) with shared resources. The transaction model [22], [17] is one such task model and in this paper we focus on its holistic schedulability tests. Holistic schedulability tests [17] compute upper-bounds of response times of precedence related tasks, by using the response time of the predecessor(s) to compute the release events of the successor(s). The response time computation is iterative. The system starts in an initial state and task response times are updated at each iteration of the test, until a convergence is reached.

To apply schedulability tests in an industrial context, they must be implemented in tools. The tools generally let the user create a model of their system according to an Architecture Description Language (ADL) [13]. Unfortunately, tools that implement holistic schedulability tests are not common [14]. This non-availability is thus one factor that explains why holistic analysis isn't widely used in the industry, the other being the pessimism [17] of response time upper-bounds if not adapted to a specific system.

In this paper, we investigate the applicability of holistic schedulability tests to industrial TDMA Software Radio Protocols (SRP) [9] developed by Thales, through the implementation of such tests in Cheddar, a real-time scheduling analysis tool [3]. Besides running on a multiprocessor partitioned system, a TDMA SRP is both a time-triggered [8] (TDMA) and event-triggered [8] (tasks handling data/control flows in the radio protocol) system. Numerous works [12] have been done previously to analyze schedulability of TDMA systems, but they only handle the time-triggered aspect of such systems, and they do not consider shared resources. For these reasons, our approach consists to model a TDMA SRP with the transaction model and assess schedulability with holistic tests. Indeed, with the transaction model, both tasks released by other tasks (event-triggered), and tasks released in time (time-triggered), can be modeled [16].

The rest of the paper is structured as follows: Section II compares some schedulability analysis tools. Section III presents Cheddar. Section IV defines the transaction model. In Section V, we expose and discuss our solution to implement holistic schedulability tests in Cheddar. In Section VI, a holistic schedulability test is applied on a TDMA SRP. Finally we conclude with future works.

II. Scheduling Analysis Tools

In this paper we focus on Cheddar, a real-time scheduling analysis tool. There exists of course several other state-of-the-art tools that perform scheduling analysis. These tools propose different ADLs and scheduling analysis methods.

Some scheduling analysis tools are based on equations to assess schedulability of a system. MAST [4] is a modeling and analysis suite for real-time applications. In the MAST toolset, an architecture is modeled with an ADL based on events. Events are sent between tasks that have precedence dependency. Tasks are allocated on processors and they may use shared resources. MAST then transforms the event-based architecture model to transactions for scheduling analysis. Holistic schedulability tests for transactions can then be applied.

SymTA/S [5] is a scheduling analysis tool originally dedicated to the automotive industry. As such, SymTA/S handles

an ADL based on entities found in automotive systems, e.g. OSEK, ECU. The architecture is modeled as components allocated on bus and processors. Components have ports through which they receive and send event streams. SymTA/S thus uses an event stream propagation model for scheduling analysis. and the tool applies a composition approach to assess schedulability. In the compositional approach, local scheduling analysis is first performed on a component and then propagated through the system (using event streams) to reach a global analysis result.

Rubus-ICE [14] is a tool suite for model-driven development of real-time systems, with modelers, code generators and analysis methods. The architecture is modeled in the Rubus Component Model (RCM) language. In RCM, software functions are modeled as components that communicate through a producer-consumer scheme. Time parameters are extracted from the component-based model and scheduling analysis methods can then be applied. Holistic schedulability tests have been implemented as plug-ins in Rubus-ICE.

Other scheduling analysis tools are based on simulation to verify (non-)schedulability. STORM [24] is a simulator for multiprocessor architectures. The architecture is described as software and hardware components. After scheduling simulation is conducted, analysis results can be shown as textual reports or graphical diagrams.

Finally some scheduling analysis tools provide several methods, including equation-based methods, formal methods, and/or simulation. Rapid RMA [23] is a set of modeling and scheduling analysis tools. The architecture is modeled with components, with the support for CORBA (an architecture standard focused on interoperability) compliant architectures. Design scenarios are then modeled for scheduling analysis. Rapid RMA uses the rate-monotonic analysis [11] to determine schedulability but it also provides a simulator.

TimeSquare [2] is a model development kit provided as a set of Eclipse plug-ins. The architecture is modeled with an UML MARTE (a UML profile for RTES) component-based model. The UML MARTE model is then transformed to a logical time model called CCSL. Model simulation can then be performed, as well as formal verification of time constraints, to assess schedulability.

Real-Time at Works (RTaW) [15] is a set of tools for timing analysis of real-time systems. Systems are first modeled in SysML (a modeling language for system engineering). RTaW is composed of several tools, including a simulator and formal methods to compute response times, for different architectures respecting industrial standards. For example, RTaW supports a number of communication buses in real-time systems, e.g. CAN, ARINC, Ethernet.

In conclusion we see that most of the existing scheduling analysis tool have an ADL based on components. We also see that holistic schedulability tests are not wide-spread among tools, which limits their usability by system designers that wish for a ”push-button easy” tool.

III. Cheddar

Cheddar is a GPL-licensed open-source real-time scheduling analysis tool written in Ada. The project was started in 2001 and since 2008 the tool is distributed as a module in AADL Inspector [3]. Fig. 1 illustrates the Cheddar tool.

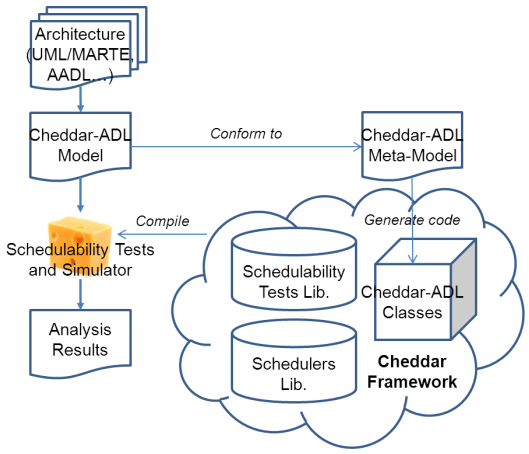

Fig. 1. Cheddar, Real-Time Scheduling Analysis Tool

Users first specify their **architecture** in Cheddar's Architecture Description Language (*Cheddar-ADL*). The GUI provided by Cheddar can be used to generate the **Cheddar-ADL model**, or model transformation [19] can be used to build an architecture in Cheddar-ADL from a standard ADL (e.g. MARTE to Cheddar [9], AADL to Cheddar [3]). A scheduling analysis method provided by the tool (**schedulability test** or **simulation**) is then used to get **analysis results** (i.e. schedulability or simulation trace).

A Cheddar-ADL model is conform to its **Cheddar-ADL meta-model**, which is specified in EXPRESS, a data modeling language. Cheddar-ADL is a language that is close to seminal scheduling analysis methods [11], [6], [21]. For example, entities of tasks, processors and shared resources are defined.

Through a model-driven process,the Cheddar-ADL meta-model is used to generate code of **Cheddar-ADL classes**, a part of the **Cheddar framework**. This ensures that the code of the Cheddar-ADL classes is always conform to the Cheddar-ADL meta-model. The Cheddar framework is composed of generated code from Cheddar-ADL and of manually written code in the **schedulability tests library** and the **schedulers library**.

To extend Cheddar, the general approach is to extend the **schedulability tests** or **schedulers libraries** of the framework. If necessary, the **Cheddar-ADL meta-model** is modified, and code of **Cheddar-ADL classes** is generated. This is the approach we have used to implement holistic schedulability tests in the tool.

IV. Transaction Model

The transaction model, proposed by [22], does not currently exist in Cheddar. This model must be implemented to specify systems on which holistic schedulability tests are applied. Let

us remind the definition of the transaction model, according to [16].

A transaction Γ_i is a group of tasks. A transaction is released by a periodic event. A particular instance of a transaction is called a job. A job of a task in a transaction is released after the event that releases the job of the transaction. Assuming Γ_i is released at t_0, each task $\tau_{ij} \in \Gamma_i$ is defined by the following parameters:

- WCET (C_{ij}) and BCET (C^b_{ij}): A task has a Worst Case Execution Time (WCET) and a Best Case Execution Time (BCET).
- Offset (O_{ij}): The offset of a task is its earliest release time after the time the transaction is released, i.e. a job of τ_{ij} is released at earliest at $t_0 + O_{ij}$.
- Jitter (J_{ij}): A task release is delayed by an arbitrary amount of time between 0 and the maximum jitter, i.e. a job of τ_{ij} is released in $[t_0 + O_{ij}; t_0 + O_{ij} + Jij]$.
- Deadline (D_{ij}): The global deadline [16] of a task is relative to the transaction release time, i.e. a job of τ_{ij} must complete execution before $t_0 + D_{ij}$.
- Blocking time (B_{ij}): Tasks may use shared resources in critical sections [21]. Shared resources access is mutually exclusive so tasks may be blocked. Shared resources are assumed to be protected by a protocol [21] that makes it possible to bound the maximum blocking time of each task, denoted B_{ij}.
- Priority ($prio(\tau_{ij})$): In case of fixed-priority scheduling, a task has a fixed priority. When two tasks want to access the processor, the higher priority task is given access in preference to lower priority task.

Fig. 2 illustrates a transaction Γ_i with tasks τ_{ij} and τ_{ik}.

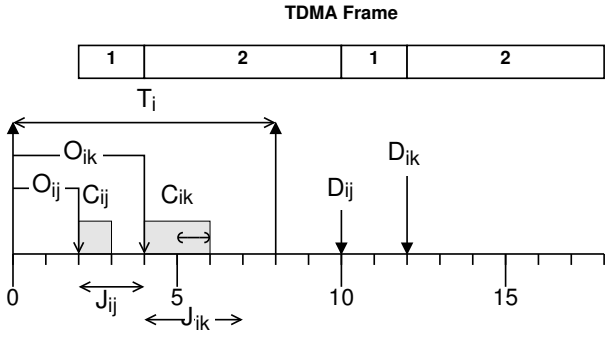

Fig. 2. Transaction Example: Upward arrows are transaction job releases; Double curved arrows are critical sections

Tasks in a transaction are related by precedence dependency. A precedence dependency between a predecessor task and a successor task is a constraint that means that a job of the predecessor task must complete its execution before a job of the successor task can be released [16].

Holistic schedulability tests compute Worst Case Response Times (WCRT) of tasks: the maximum time between a task's earliest release time and its latest completion time.

V. Implementing Transactions and Holistic Schedulability Tests in Cheddar

We now show how Cheddar was extended for transactions and holistic schedulability tests. First our modifications to Cheddar-ADL is shown. Then we discuss how the tests were implemented, focusing on implementation issues and choices we faced.

A. Extending the Cheddar-ADL Meta-Model

To see if Cheddar-ADL is sufficient to implement the transaction model, we focus on a partial meta-model of Cheddar-ADL with task entities, in Fig. 3. In the following sections, reference of entities are those in Fig. 3.

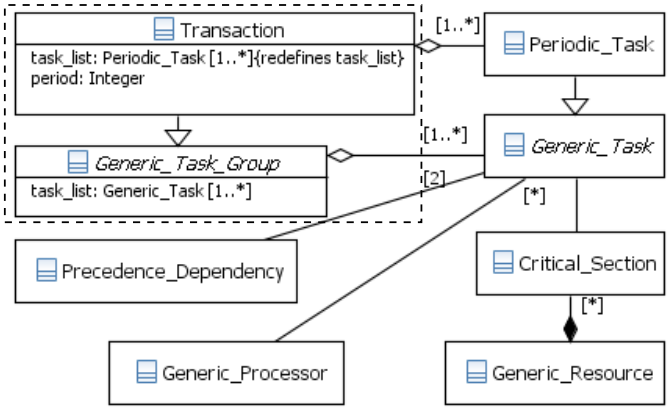

Fig. 3. Cheddar-ADL Partial Meta-Model: Dashed box highlight extensions

1) Re-use of Existing Entities: A number of entities that already exist in Cheddar-ADL can be re-used to implement the transaction model. The `Periodic_Task` entity has attributes of a task in the transaction model. In Cheddar-ADL any task entity can have a `Critical_Section` where it uses a `Generic_Resource` entity, representing a shared resource. Furthermore a task is allocated on a `Generic_Processor` entity. Through the `Precedence_Dependency` entity, a precedence dependency can be specified between two tasks.

The main entity in the transaction model, that cannot be modeled in the current Cheddar-ADL, is the transaction entity itself.

2) New Task Group Entities: To model a transaction entity, we introduce a *task group* concept in the Cheddar-ADL meta-model. A *task group* is modeled by the new entity `Generic_Task_Group` in Fig. 3.

A `Generic_Task_Group` is a set of tasks. Like task entities, any *task group* entity inheriting from `Generic_Task_Group` may have attributes. *Task group* attributes constrain attributes of tasks in the *task group*. The type of tasks that can be in a *task group* are also constrained. For example the `Transaction` entity in Fig. 3 is used to model a transaction with period T_i represented by its attribute `period`. A `Transaction` can only contain `Periodic_Task` entities.

Fig. 4 illustrates the transaction in Fig. 2 modeled in Cheddar-ADL (XML) with the new *task group* entities.

```
<periodic_task id="tau_ij">
 <name>tau_ij</name>
 <capacity>1</capacity>
 <offsets>
  <offset_type>
   <offset_value>2</offset_value>
   <activation>0</activation>
  </offset_type>
 </offsets>
 <jitter>2</jitter>
 <deadline>10</deadline>
 <blocking_time>0</blocking_time>
 <priority>1</priority>
 <period>8</period>
</periodic_task>

<transaction id="tdma_tasks">
 <name>TDMA_Tasks</name>
 <task_list>
  <periodic_task ref="tau_ij"/>
  <periodic_task ref="tau_ik"/>
 </task_list>
 <period>8</period>
</transaction>

<periodic_task id="tau_ik">
 <name>tau_ik</name>
 <capacity>2</capacity>
 <offsets>
  <offset_type>
   <offset_value>4</offset_value>
   <activation>0</activation>
  </offset_type>
 </offsets>
 <jitter>3</jitter>
 <deadline>10</deadline>
 <blocking_time>0</blocking_time>
 <priority>1</priority>
 <period>8</period>
</periodic_task>
```

Fig. 4. Task Group Example

B. Implementation of Holistic Schedulability Tests

After extending the Cheddar-ADL meta-model with transactions, code for the Cheddar-ADL classes was generated. In total 738 lines of code were generated for the new *task group* entities. No extra entities or structures were added to the framework. The holistic tests in [1], [22], [16], [18], [10] were implemented, using the generated code. The main differences between these tests is that they reduce pessimism of response time upper-bounds when considering a specific release pattern in a transaction (e.g. a task can release several tasks immediately [18] and non-immediately [10]).

We now discuss some implementation choices we made and expose issues we faced.

1) Advantages of Implementation Solution: The solution we proposed to model transactions, introduces the *task group* entities but re-uses most of the Cheddar-ADL mechanism for tasks. This has an advantage in terms of meta-model and code maintenance.

The `Transaction` entity we introduced, is generic enough to model any kind of transaction. Indeed, the main difference between different kinds of transactions is their release pattern, i.e. tasks can have more or less successors and predecessors [16], [18], [7]. Since the `Precedence_Dependency` entity in Cheddar-ADL is used to determine precedence between tasks, the order in which tasks are grouped in a `Transaction` does not determine their precedence dependencies. Any task precedence dependency can be represented with the `Precedence_Dependency` entity.

2) Drawbacks of Implementation Solution: Our implementations of holistic schedulability tests use most of the entities that are already present in Cheddar. The main issue from this implementation choice is the time performance of the schedulability tests. Indeed, like stated previously, the `Transaction` entity and `Precedence_Dependency` entity are enough to represent any kind of transaction. On the other hand, these entities may not be the best structure to represent some kinds of transaction.

For example, let us consider the operation to get the successors/predecessors of a task, a common operation among those necessary for holistic schedulability tests. A linear transaction [16], where tasks have at most one successor/predecessor, is best represented with a table. A table reduces considerably the complexity of the operation to get the successor/predecessor of a task in a linear transaction. Indeed, with a table, the operation is $O(1)$ while in our implementation, we have to loop through all entries in the set of `Precedence_Dependency`, so the operation to get a successor/predecessor is $O(n)$.

The general solution to the complexity problem is to implement each kind of transaction with the best adapted data structure. However, later in this paper, we will see that experimental results show that the current implementation stays scalable to a real TDMA software radio protocol.

VI. Experiment

To evaluate our implementation in an industrial context, we apply the test in [10] (called WCDOPS+_NIM and based on [18]) to a real TDMA SRP developed by Thales. In the following sections the TDMA SRP system is first presented before we show how the test is applied.

A. TDMA Software Radio Protocol

A TDMA SRP is a communication protocol embedded in a radio station in a mobile ad-hoc wireless network.

1) System View: From a system point of view, a SRP is divided into several layers according to the OSI model for communication systems. Fig. 5 shows an example of such layers.

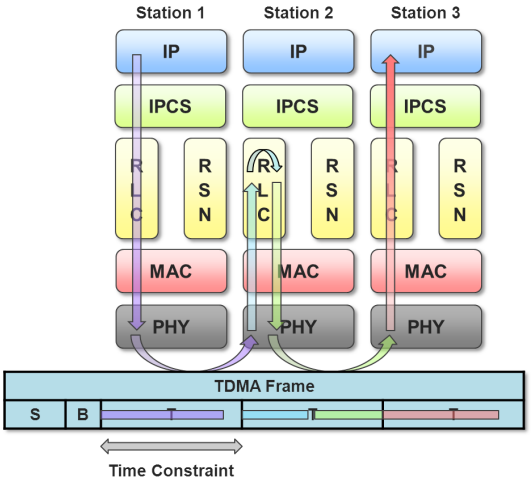

Fig. 5. TDMA SRP System View

In Fig. 5, the **IPCS** layer interfaces with the IP stack of the user system above. The **RLC** layer handles translation between IP packets and radio protocol packets. It also re-routes incoming packets if necessary (e.g. a received packet's destination is a neighbor). The **RSN** layer handles network topology and address updates (e.g. address of neighbor stations in the network appearing/disappearing). When a SRP uses TDMA, the **MAC** layer handles the TDMA protocol by preparing/receiving protocol packets for/from the **PHY** layer that sends/receives them over the air.

In Fig. 5, control and data flows pass through the different layers. The flows are constrained by the TDMA frame. A

TDMA frame is divided into several time slots of different types, durations, and modes. For example in Fig. 5, the TDMA frame has three kinds of slot: **Service** (**S**) for synchronization between stations; **Broadcast** (**B**) for observation/signaling on the network; **Traffic** (**T**) for effective data transmission/reception. Slots of different types do not have the same duration (e.g. a **B** slot is shorter than a **S** and **T** slot). Slots can either be in **Tx** (transmission), **Rx** (reception), or **Idle** mode. A TDMA configuration defines the combination of slots (type and mode) in a TDMA frame. A TDMA frame is repeated after it finishes, with possibly a different configuration. We assume that in a TDMA configuration, only the slot modes change from one TDMA frame to the next.

2) Software and Execution Platform: Fig. 6 shows an example of a the software and the execution platform architecture of a SRP.

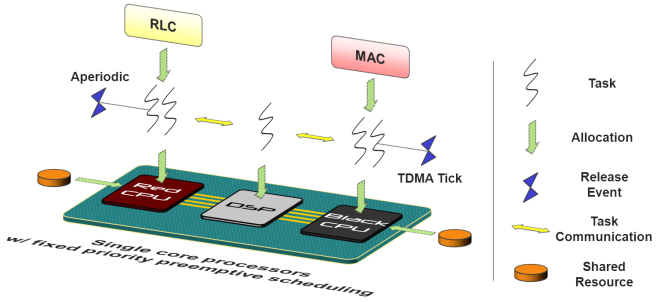

Fig. 6. Software and Execution Platform Architecture

From Fig. 6, we see that the layers are implemented by tasks allocated on processors. In our case-study, tasks that implement layers are POSIX threads so from now on we will call them ”pthread”. Pthreads are scheduled by a fixed priority preemptive policy. Pthreads may communicate and use shared resources. Pthreads handle the flows and they are also constrained by the TDMA frame. For example a pthread may be released by a TDMA tick indicating the start of a slot. Furthermore, each pthread has an execution time that depends on a specific slot, and phreads must finish before some next slot.

Pthreads in the **MAC** layer have hard deadlines to meet, since they handle the TDMA protocol. For this reason the **MAC** layer is the one that interests us for schedulability analysis. The **MAC** layer is implemented on the *Black CPU* and the *DSP* in Fig. 6.

B. MAC Layer Schedulability

In this section we show how to apply the WCDOPS+_NIM test to a **MAC** layer. First the system to analyze is exposed and modeled with a transaction. Then, after applying the test, the schedulability analysis results are discussed. Finally we discuss the modeling of our system with the transactions.

1) System to Analyze: The **MAC** layer has several pthreads constrained by a TDMA frame. The time parameters of pthreads instances are illustrated in Fig. 7. For readability issues, sizes in the figure are not proportional to time values.

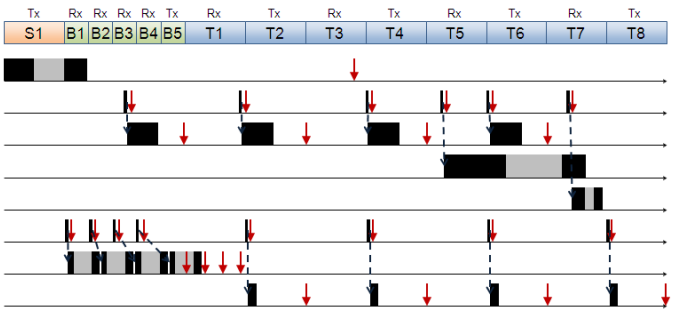

Fig. 7. TDMA Frame and Pthreads: Line = Instances of a pthread; Down arrow = Deadline; Dashed arrow = Precedence; Black = Exec on *Black CPU*; Gray = Exec on *DSP*; 9 pthreads (36 instances) in total

We see that pthreads execute on the *Black CPU* but they may call a function on *DSP* and wait for the answer (i.e. blocking call). We also see that pthreads communicate (i.e. have precedence dependency).

Pthreads are dedicated to either transmission, reception or utility. The basic pthread release pattern is that a reception pthread is released at the start of a **Rx** slot. A transmission pthread is released before a **Tx** slot so data is ready before transmission over the air in the **Tx** slot. There is only one utility pthread that is released at the beginning of the TDMA frame, to prepare the configuration of the next TDMA frame.

Fig. 8 shows the transaction model of pthreads in the **MAC** layer, constrained by the TDMA frame.

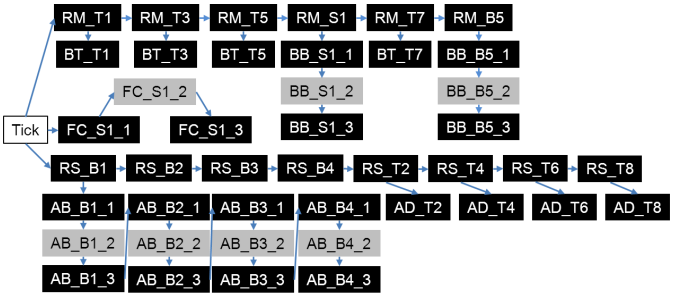

Fig. 8. Tree-Shaped Transaction of **MAC** Layer: Black tasks on *Black CPU*; Gray tasks on *DSP*; Tick task is *ghost root task* [18] on unique processor with 0 WCET

2) Analysis Evaluation: To assess the advantage of applying a holistic schedulability test to our system modeled with a transaction, we compare WCRTs given by WCDOPS+_NIM with those computed by the test in [6]. We choose to compare to the test in [6] because it is a classic one for periodic tasks and because the same approach is applied at Thales.

For each task, let us call the WCRT given by [6] R_{RM}, and $R_{WCDOPS+\_NIM}$ the WCRT given by WCDOPS+_NIM. The ratio $R_{RM}/R_{WCDOPS+\_NIM}$ is computed for each task. In average this ratio is 5.1 so, in average, the test in [6] gives a WCRT more than 5 times higher than WCDOPS+_NIM. Thus to limit pessimism of analysis results, the particular task releases (in time and by other tasks) of our system needs to be considered. It is then beneficial to use transactions to model

a TDMA SRP and apply a holistic schedulability test, since it increases such system's schedulability compared to the test for classic periodic tasks in [6].

3) Modeling Evaluation: Notice that there are 9 pthreads in Fig. 7 but 43 tasks in the transaction in Fig. 8. This is because several instances of a pthread in Fig. 7 are modeled as several tasks in Fig. 8. A pthread on *Black CPU* that makes a blocking call to a function on *DSP*, is also modeled as several tasks in Fig. 8. The transaction used to analyze the system is thus more complex than the original model of the system. In general, the more instances of a pthread there are, the more tasks there are in the transaction. Similarly, in a pthread's execution, the more blocking calls of functions (implemented by other pthreads) there are, the more tasks there are in the transaction.

The difference between model complexities raises several issues. When WCDOPS+_NIM is applied on our model, the time to compute WCRTs takes 7 seconds on an Intel Core i5 @ 2.40GHz. The actual time taken by the analysis is much higher than the time taken to extract information from Fig. 7 and model the system with the transaction in Fig. 8, if done manually. With the XML in Fig. 4, we also see that the modeling process in Cheddar can be tedious. Finally, when the system is modeled manually with transactions, the modeling asks for scheduling analysis theory expertise from designers and the risk of mistakes is not nonexistent. For this reason model transformation tools should be developed.

VII. Conclusion

In this paper we showed how we implemented holistic schedulability tests in the Cheddar real-time scheduling analysis tool, to apply them to an industrial TDMA SRP developed by Thales. We extended the Cheddar-ADL with transactions and generated code of the Cheddar-ADL classes. New tests [1], [22], [16], [18], [10] were then integrated into Cheddar. Experimental results show that a holistic schedulability test, applied to our system, gives WCRT bounds more than 5 times less than classic tests used at Thales. Holistic schedulability tests thus reduce considerably the pessimism of computed WCRTs for our TDMA SRP.

The implementation of the tests in a tool like Cheddar, increases their usability by system designers in the industry. On the other hand, the transaction model of our system is more complex than the original model. Thus model transformations should be developed. In the future we will develop model transformations to integrate the Cheddar tool in a development process at Thales, by transforming system design models to Cheddar-ADL models automatically.

Acknowledgment

The authors would like thank Vuong Nguyen Hong for his contribution to this work.

References

[1] N. Audsley, K. Tindell, and A. Burns. The end of the line for static cyclic scheduling? In *Proceedings: 5th Euromicro Workshop on Real-Time Systems*, 1993.

[2] J. DeAntoni and F. Mallet. TimeSquare: treat your models with logical time. In *Objects, Models, Components, Patterns*, volume 7304 of *Lecture Notes in Computer Science*, pages 34–41. Springer Berlin Heidelberg, Berlin, Germany, 2012.

[3] P. Dissaux, O. Marc, S. Rubini, C. Fotsing, V. Gaudel, F. Singhoff, A. Plantec, V. Nguyen Hong, and H. N. Nam Tran. The SMART project: Multi-agent scheduling simulation of real-time architectures. In *Proc. 7th European Congr. Embedded Real Time Software and Syst.*, Toulouse, France, 2014.

[4] M. G. Harbour, J. G. Garcia, J. Palencia, and J. Drake Moyano. MAST: modeling and analysis suite for real time applications. In *Proc. 13th Euromicro Conf. on Real-Time Syst.*, pages 125–134, Delft, Netherlands, 2001.

[5] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System level performance analysis the SymTA/S approach. *IEE Proc. - Comput. and Digital Techniques*, 152(2):148, Mar, 2005.

[6] M. Joseph and P. Pandya. Finding response times in a real-time system. *Comput. J.*, 29(5):390–395, 1986.

[7] J. Kany and S. Madsen. Design optimisation of fault-tolerant event-triggered embedded systems. Master's thesis, Tech. Univ. of Denmark, Lyngby, Denmark, 2007.

[8] H. Kopetz. Event-triggered versus time-triggered real-time systems. In *Operating Systems of the 90s and Beyond*, volume 563 of *Lecture Notes in Computer Science*, pages 86–101. Springer Berlin Heidelberg, 1991.

[9] S. Li, F. Singhoff, S. Rubini, and M. Bourdellès. Applicability of real-time schedulability analysis on a software radio protocol. *ACM SIGAda Ada Lett.*, 32(3):81–94, Dec, 2012.

[10] S. Li, F. Singohff, R. Stéphane, and M. Bourdellès. Extending schedulability tests of tree-shaped transactions for TDMA radio protocols. In *Proc. 19th IEEE Intl. Conf. on Emerging Technology & Factory Automation*, Barcelona, Spain, 2014.

[11] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, Jan, 1973.

[12] N. Malcolm and W. Zhao. The timed-token protocol for real-time communications. *Comput.*, 27(1):35–41, Jan, 1994.

[13] N. Medvidovic and R. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, 2000.

[14] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Implementation of holistic response-time analysis in rubus-ICE: preliminary findings, issues and experiences. In *Proc. 32nd IEEE Real-Time Syst. Symp., WIP Session*, Vienna, Austria, 2011.

[15] N. Navet, S. Louvart, J. Villanueva, S. Campoy-Martinez, and J. Migge. Timing verification of automotive communication architectures using quantile estimation. In *Proc. 7th European Congr. Embedded Real Time Software and Syst.*, Toulouse, France, 2014.

[16] J. Palencia and M. Harbour. Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In *Proc. 20th IEEE Real-Time Syst. Symp.*, pages 328–339, Phoenix, AZ, 1999.

[17] A. Rahni, E. Grolleau, M. Richard, and P. Richard. Feasibility analysis of real-time transactions. *Real-Time Syst.*, 48(3):320–358, May, 2012.

[18] O. Redell. Analysis of tree-shaped transactions in distributed real time systems. In *Proc. 16th Euromicro Conf. Real-Time Syst.*, pages 239–248, Catania, Italy, 2004.

[19] S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *IEEE Software*, 20(5):42–45, Sep, 2003.

[20] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Syst.*, 28(2-3):101–155, Nov-Dec, 2004.

[21] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185, Sep, 1990.

[22] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2-3):117–134, Apr, 1994.

[23] Tri-Pacific Software Inc. Tri-pacific software inc. : RAPID RMA.

[24] R. Urunuela, A. Déplanche, and Y. Trinquet. STORM a simulation tool for real-time multiprocessor scheduling evaluation. In *Proc. 2010 IEEE Conf. Emerging Technologies and Factory Automation*, pages 1–8, Bilbao, Spain, 2010.