Validation of Embedded Systems Behavioral Models on a Component-Based Ericsson Nikola Tesla Demonstrator [1]

Aneta Vulgarakis[†], Cristina Seceleanu[†], Paul Pettersson[†], Ivan Skuliber[‡], and Darko Huljenić[‡]

[†] Mälardalen Real-Time Research Centre, Västerås, Sweden

[‡] Ericsson Nikola Tesla, Zagreb, Croatia.

aneta.vulgarakis@mdh.se, cristina.seceleanu@mdh.se, paul.pettersson@mdh.se,
ivan.skuliber@ericsson.com, darko.huljenic@ericsson.com

Abstract—**Embedded systems are challenging to design, due to the implementation platform constraints that have to be considered, preferably from early stages of design, next-by system functionality. Hence, embedded system models need to be timing and resource-aware, to make formal verification of extra-functional properties applicable at high levels of abstraction. In most cases, a frequent obstacle to the successful application of such rigorous techniques is the lack of the proposed models' validation against real-world application measurements. In this paper, we show how to model extra-functional behavior, and verify the resulted behavioral models of a component-based Ericsson Nikola Tesla prototype telecommunications system. The models are described in our recently introduced REMES language, with Priced Timed Automata semantics that allows us to apply UPPAAL -based tools for model-checking the system's response time and compute optimal resource usage traces. The validation of our models is ensured by using actual values of timing, CPU, and memory usage in our models, measured by Ericsson researchers on the prototype's source code.**

I. INTRODUCTION

The usefulness, industrial applicability, and scalability of embedded systems (ES) modeling languages and analysis methods can be exercised by performing their validation against measured, quantified behavioral properties. Validation loosely refers to the process of determining if a design is correct with respect to implementation requirements [13]. The two most usual model validation procedures are simulation, which traverses a subset of the system's behaviors, and *formal verification*, which is exhaustive but limited to finite-state systems. Although usually restrictive, in the ES case formal verification proves more applicable due to the constrained way in which ES need to be designed. The latter have to meet extra-functional, platform-driven constraints, such as timing, energy, CPU, or/and memory constraints.

Motivated by the above, in this paper we describe the modeling and formal analysis of a prototype industrial telecommunications system, a demonstrator developed by Ericsson Nikola Tesla, in Croatia [29]. The new system is a proof-of-concept solution, developed by applying the

component-based design paradigm, by adding a newly developed authentication, authorization, and accounting service to a complex basic service telecom system consisting of several existing and reused components, such as a DIAMETER standard protocol, an open-source *Pen* load balancer, and a number of servers. More precisely, the ENT demonstrator uses the basic service to issue call requests, to which the *extension service* delivers the authentication functionality on a request basis. The resulting system is a telecom system that must adhere to the characteristics of existing telecom legacy systems. Thus, two requirements are imposed on the demonstrator: capacity and optimal resource usage. The demonstrator is described in section III.

Our analysis effort of the above properties is driven by both an academic, as well as an industrial interest. The former targets exercising the industrial applicability and validation of our favorite embedded systems behavioral modeling and analysis framework, which is tailored to the recently introduced ProCom component model [24] for real-time ES, briefly recalled in section IV-A. The framework consists of the resource-aware timed behavioral language REMES [23], reviewed in section IV-B, and its underlying formal model given in terms of Priced Timed Automata networks [1], [6] (see section IV-C). The industrial interest is in being able to use a virtual experimental "lab", in which various types of extra-functional analysis of the demonstrator could be carried out, which could provide valuable feedback on the demonstrator's performance, and resource-usage, assuming various settings, prior to an actual implementation of the respective setting.

In section V, the demonstrator is modeled in a component based fashion using the ProCom component modeling language, whereas the functional, timing, and resource-wise behavior of the key components of the system are modeled in REMES, as shown in section VI-A. The salient point of our model, which enables its validation, is the fact that we build it by using the timing and resource values extracted from the actual prototype implementation of the demonstrator. We also show how the combined model is semantically translated into a network of (priced) timed automata to enable model-checking in the tools UPPAAL and
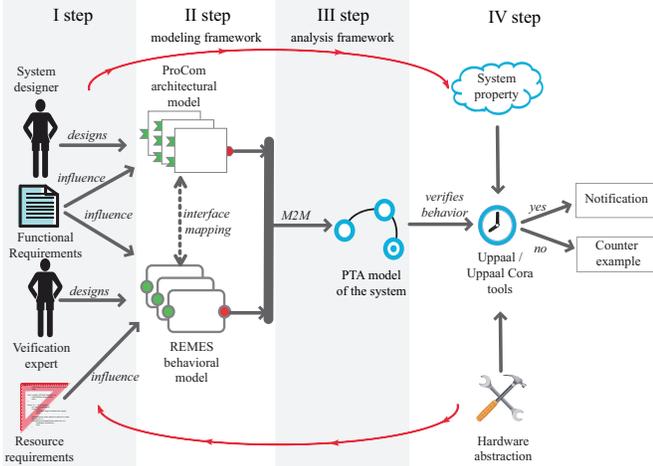
Figure 1. The system validation process.

Figure 2. The deployment architecture of the demonstrator.

UPPAAL CORA (see section VI-C).

Regarding the system analysis, we consider, next to function and timing, a weighted sum of the resources CPU and memory, in which CPU is considered a more critical resource than memory (twice the relative weight of memory). Under this assumption, we derive an optimal system trace, the minimum time, and the minimal total accumulated weighted resource cost for processing a given number of system requests. The results are described in section VI-C.

Similar industrial-driven endeavors include research work within the framework of timed automata [7], [12], [15], GIOTTO [14], or BIP [22]. However, such works either do not consider formal verification of extra-functional system behavior, or do not base it on real timing and resource values measured on actual implementations.

II. OVERVIEW OF THE VALIDATION PROCESS

The validation process that we use in our case study is iterative, allowing feedbacks between steps. It consists of four steps (see Figure 1) as follows.

- **I step.** Based on the system functional requirements the designer builds the ProCom architectural model of the system. Similarly, the verification experts uses both the functional- and resource requirements (such as timing, memory, etc.) to develop the REMES behavioral model of the system.
- **II step.** During this step an interface mapping between the ProCom architectural- and the REMES behavioral model is performed. For more information, we refer the reader to [28].
- **III step.** The ProCom architectural- and the REMES behavioral model are together transformed to priced timed automata (PTA) model for formal analysis. The architectural model gives information about the order of execution of the REMES modes modeling the behavior of the components.
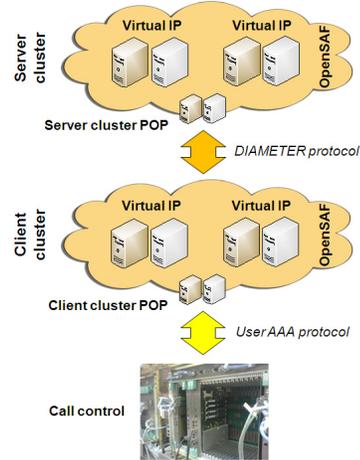
- **IV step.** We assume that we have hardware abstraction that provides us with global available resources (e.g., memory budget, cpu load, bandwidth of the communication network, etc.). To perform model-checking, a PTA model of the system is fed into UPPAAL, together with a hardware abstraction and a desired property (requirement) expressed in a temporal logic. UPPAAL then automatically traverses the system's state space in an exhaustive manner. If an invariant property is satisfied, the tool notifies that the verification finished successfully, or if the invariant property is violated, it reports one of the traces that violates the property as a counter-example to the model. For reachability properties the opposite is true i.e., a trace is reported when the property is satisfied.

III. DESCRIPTION OF THE DEMONSTRATOR

Ericsson Nikola Tesla's (ENT) demonstrator is a prototype of a telecommunications system. It is designed according to current telecommunications industry's trends of adapting horizontal development (systems built from reusable components) methodologies instead of traditionally used vertical ones (systems built from ground-up in-house, now called legacy systems). The organization of the demonstrator is shown in Figure 2 from the perspective of deployment architecture.

In the demonstrator, a new telecommunications service is created with horizontal development. This new service is added to existing, so called *basic service* that was created over the years with vertical development. More precisely, the basic service performs typical call control functionality: decoding of addressing information and routing calls from one end-point to another. When a special kind of processing is needed, it generates events that result with requests (messages) that are being redirected into the *extension service*. The extension service processes messages generated by the

basic service by performing an AAA (authentication, authorization and accounting) functionality that conforms to the widely accepted Internet standard called DIAMETER [20]. The result of the processing are also messages that are sent back to the basic service.

The extension service is realized as *clients*- and *servers cluster*, which communicate via DIAMETER protocol. They should assure high levels of performance through round-robin load balancing, and availability through redundancy. Implementation of high availability and reliability is facilitated with the use of *OpenSAF*. Previous experiments performed by Ericsson researchers show negligible impact of OpenSAF to the overall performance of the demonstrator [29]. Thus, we omit OpenSAF from experiments shown in this paper.

Pen is a third-party open-source load balancer that was customized for the purpose of load balancing stateful AAA protocol between the basic service and the extension service. Pen maintains the information (e.g., IP addresses and ports) about which call control node is communicating with which DIAMETER *client* and uses round-robin method for choosing which client will serve a given request. DIAMETER client receives AAA requests through the AAA protocol between the basic service and the extension service, transforms them into DIAMETER-based AAA requests and sends these DIAMETER-based AAA requests to the DIAMETER servers cluster.

DIAMETER *relay* is a DIAMETER protocol functionality that is used for balancing the load among DIAMETER servers. Similar to Pen, it uses round-robin method for choosing which server will serve a given request. Since each DIAMETER message contains full address information about communicating peers, it just transmits the response received from DIAMETER server to corresponding DIAMETER client that originated the initial request. DIAMETER *server* receives DIAMETER-based AAA requests originated on DIAMETER clients. It processes these requests and returns the results to the relay. Since the original request contained the information about which client created it, the relay knows to which client the response must be sent to.

IV. BACKGROUND

A. The ProCom component model

ProCom [24] is component model that aims at addressing key design characteristics and development concerns of distributed embedded systems. ProCom comprises two-layered component model, and differentiates a component model used for modeling distributed, complex, active and concurrent subsystems (called ProSys) and a component model used for modeling non-distributed, passive and smaller units of control functionality (called ProSave). The two layers relate to each other in the sense that a ProSys component can be modeled out of ProSave components. In this paper,
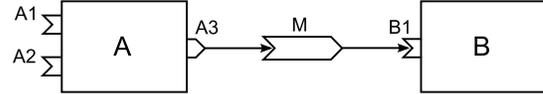
Figure 3. Example of the ProSys notation: Subsystem A has two input ports (A1 and A2) and one output port (A3), and communicates with subsystem B by sending messages over the message channel M.

we only use the more large scale ProSys. The complete specification of ProCom is available in [9].

In ProSys, the communication between subsystems is based on asynchronous message passing. A subsystem receives and sends messages through input- and output message ports, respectively. Message ports are connected though message channels. A message channel captures a piece of shared data that is of interest to one or more subsystems before any producer or receiver of this data has been defined. Figure 3 exemplifies the graphical notation of ProSys.

B. The REMES *language for behavioral modeling*

The REsource Model for Embedded Systems REMES [23] is intended as a meaningful basis for modeling and analysis of resource-constrained behavior of embedded systems.

To enable formal analysis, REMES models can be transformed into timed automata (TA) [3], or priced timed automata (PTA) [1] (see subsection IV-C), depending on the analysis type.

The internal behavior of a component is depicted by a REMES *mode* that can be *atomic* (see Atomic mode 1, Atomic mode 2 in Figure 4) or *composite* (made of atomic modes). The discrete control of a mode is captured by a *control interface* that consists of *entry*- and *exit* points, whereas the data transfer between modes is carried out through a well-defined *data interface* that consists of typed global variables. A composite mode may also have a special *init* entry point where the global variables are initialized.

A composite mode executes by performing a sequence of *discrete steps*, via actions that, once executed, pass the control from the current submode to a different submode. An action, $A = (g, S)$ (e.g., $(y == b, d := u)$ in the figure), is a statement S (in our case $d := u$), preceded by a Boolean condition, the *guard* ($y == b$), which must hold in order for the action to be executed and the corresponding outgoing edge taken. A REMES composite mode may contain *conditional connectors* (decorated with letter C) that allow a possibly nondeterministic selection of one discrete outgoing action to execute, out of many possible ones. Below, via C, one of the empty statement actions, $x \leq a \wedge d == v$ or $d \geq v$ can be chosen for execution.

In REMES one may model timed behavior and resource consumption. Timed behavior is modeled by global continuous variables of specialized type *clock*, that is, x, y in our figure, evolving at rate 1. A Boolean condition called *invariant* (e.g., $y \leq b$) may be used to specify for how long an atomic mode can be executed. Once the invariant stops
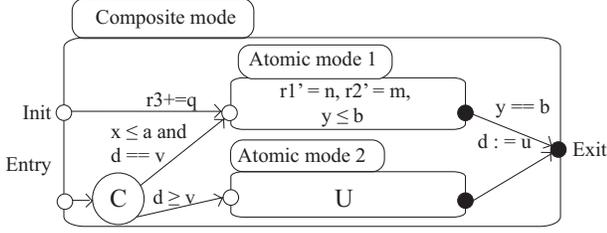
Figure 4. A REMES composite mode.

to hold, the current mode is exited. In case a mode is exited instantaneously after its activation, the mode is called *urgent* (decorated with letter U). In case a mode does not contain any invariant to specify how long it is allowed to delay in that mode, the mode is *non-lazy*; time is allowed to pass in a non-lazy mode until at least one of the guards of the outgoing discrete actions is true.

The composite mode in Figure 4 has two continuous resources (r1 and r2) and one discrete resource r3. Consumption of the continuous resources is expressed by their first derivatives (r1' and r2'), see Atomic mode 1, which give the rates at which the mode consumes the resources, respectively (r1 is consumed at rate n, whereas r2 at rate m, where m and n are integers). Discrete resources are allocated through usual updates, e.g., r3 += q. For a more thorough description of REMES, we refer the reader to our recent work [23].

The REMES language benefits from a set of tools[2] for modeling, simulation and transformation into PTA, which could assist the designer during system development [18].

C. Priced Timed Automata

In the following, we recall the model of priced (or weighted) timed automata [1], [6], an extension of timed automata [3] with prices/costs on both locations and transitions, which acts as the semantics basis of REMES.

Let X be a finite set of clocks and $\mathcal{B}(X)$ the set of formulas obtained as conjunctions of atomic constraints of the form $x \bowtie n$, where $x \in X$, $n \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. The elements of $\mathcal{B}(X)$ are called *clock constraints* over X.

Definition 1: A linearly Priced Timed Automaton (PTA) over clocks X and actions Act is a tuple (L, l_0, E, I, P), where L is a finite set of locations, l_0 is the initial location, $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(X) \times L$ is the set of edges, $I : L \to \mathcal{B}(X)$ assigns invariants to locations, and $P : (L \cup E) \to \mathbb{N}$ assigns prices (or costs) to both locations and edges. In the case of $(l, g, a, r, l') \in E$, we write $l \xrightarrow{g,a,r} l'$. ∎

The semantics of a PTA is defined in terms of a timed transition system over states of the form (l, u), where l is a location, $u \in \mathbf{R}^X$, and the initial state is (l_0, u_0), where u_0 assigned all clocks in X to 0. Intuitively, there are two kinds

of transitions: delay transitions and discrete transitions. In delay transitions,

$$(l, u) \xrightarrow{d,p} (l, u \oplus d)$$

the assignment $u \oplus d$ is the result obtained by incrementing all clocks of the automata with the delay amount d, and $p = P(l) * d$ is the cost of performing the delay. Discrete transitions

$$(l, u) \xrightarrow{a,p} (l', u')$$

correspond to taking an edge $l \xrightarrow{g,a,r} l'$ for which the guard g is satisfied by u. The clock valuation u' of the target state is obtained by modifying u according to updates r. The cost $p = P((l, g, a, r, l'))$ is the price associated with the edge.

A timed trace σ of a PTA is a sequence of alternating delays and action transitions

$$\sigma = (l_0, u_0) \xrightarrow{a_1, p_1} (l_1, u_1) \xrightarrow{a_2, p_2} \dots \xrightarrow{a_n, p_n} (l_n, u_n)$$

and the cost of performing σ is $\sum_{i=1}^n p_i$. For a given state (l, u), the minimum cost of reaching (l, u) is the infimum of the costs of the finite traces ending in (l, u).

A network of PTA $A_1 || \dots || A_n$ over X and Act is defined as the parallel composition of n PTA over X and Act. Semantically, a network again describes a timed transition system obtained from those components, by requiring synchrony on delay transitions and requiring discrete transitions to synchronize on complementary actions (i.e. $a?$ is complementary to $a!$).

In order to specify properties of PTA, the logic Weighted CTL (WCTL) has been introduced [8]. WCTL extends Timed CTL with resets and testing of cost variables. We refer the reader to [8] for a thorough introduction of WCTL.

V. THE PROCOM ARCHITECTURE OF THE DEMONSTRATOR

In this section, we describe a software architecture of the ENT demonstrator that adheres to the ProCom component model. The internal design of the ENT demonstrator is modeled by two larger subsystems: Basic Service and Extension Service, as depicted in Figure 5. The interfaces of the subsystems are expressed in terms of message ports.
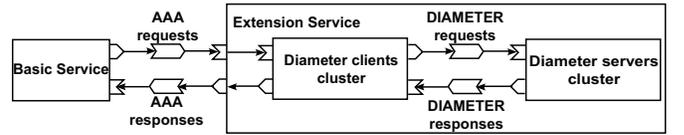
Figure 5. The ProSys model of the ENT demonstrator.

Basic Service is an existing legacy ProSys component. Extension Service is a subsystem composed of two smaller ProSys components: Diameter clients cluster and Diameter servers cluster. We consider that there are four clients (resp. servers) in Diameter clients cluster (resp. Diameter servers cluster). Each of these ProSys components may be further decomposed into either smaller ProSys components, or into

ProSave components, depending on the level of complexity of the functionality, and the possibility for distribution. Accordingly, the Diameter clients cluster component is built from Pen ProSave component and four Client ProSave components. Similarly, the Diameter servers cluster component is made of Relay ProSave component and four Server ProSave components.

The component Basic Service sends AAA requests to Extension Service. These requests are forwarded to Diameter clients cluster component. Inside this cluster, the Pen component is responsible with forwarding these messages in a round-robin fashion to each of the four clients. The Diameter clients cluster is the client side of the DIAMETER protocol. Thus, inside Diameter clients cluster component AAA requests are transformed into DIAMETER requests and are forwarded to the Diameter servers cluster component. Relay, similarly to Pen, forwards the DIAMETER requests messages in a round-robin manner to each of the four servers. The servers process these requests and return DIAMETER responses to Relay that forwards them to Diameter clients cluster. In the end, Diameter clients cluster component transforms DIAMETER responses into AAA responses and sends them back to Basic Service.

VI. REMES MODELING AND FORMAL ANALYSIS OF THE DEMONSTRATOR

A. The REMES model of the ENT demonstrator

We model the functional, timing and resource usage behavior of the ENT components as models in REMES. Due to space limitation, here we only present the REMES models of the Pen component, one of the clients from Diameter clients cluster and one of the servers from Diameter servers cluster, depicted in Figure 6(a), 6(b), and 6(c), respectively. Relay has similar behavior as the Pen component.

In the ENT demonstator, we make use of two resources: memory and CPU. We assume CPU as a continuous resource, and we treat memory as a discrete resource. Note that in the current version of the demonstrator the clients and the servers are homogenous. From the measurements performed on the source code, we have concluded that Relay is the the slowest component in the ENT demonstrator and the one that consumes the most resources. Moreover, since the servers are homogenous, we have noticed that Relay's round-robin load balancing protocol always sends messages coming from the first-, second-, third- and fourth- client to the first-, second-, third- and fourth server, respectively.

The Pen mode is made up of two submodes: Pen_Input and Pen_Output. Pen starts executing by entering Pen_Input mode and its submode Receive_IP, where it reads instantaneously the addresses of the clients. Pen may be reentered in case the Basic Service component sends a new request (depicted with the Boolean variable req) or in case one of the clients is ready to send a response (i.e., at least one of the Boolean variables client1prio, client2prio, client3prio

or client4prio is evaluated to true). Basic Service may send requests to Pen only when Pen is free (captured with the Boolean variable penfree).

The Pen_Input mode is responsible for sending requests to the clients in a round-robin fashion. We use the variable counter to ensure the round-robin principle. For example, Pen is ready to send a message to Client1 when the guard client1prio and (counter mod 4==0) is evaluated to true. The Pen_Output mode receives responses from the clients and forwards them to Basic Service.

Client1 receives requests from Pen, processes them, and forwards the processed requests to Relay. Later, Relay sends responses to the requests back to Client1. Client1 sends on the responses to Pen. Note that Client1 can process only one request at a time. The fact that Client1 has to send back response to Pen before receiving a new request is depicted with the Boolean variable client1prio.

The Client1 component remains in the non-lazy mode Waiting_for_Pen until receiving a send_c1 message from Pen. When this happens, the component goes via sequence of submodes: Client1_to_Relay, Waiting_for_Relay and Client1_to_Pen. Client1 stays in modes Client1_to_Relay and Client1_to_Pen as long as their invariants hold (i.e., until $t \leq 25$ and $t \leq 103$, respectively). The submode Waiting_for_Relay is exited when a relay_to_c1 message arrives.

The Server1 component receives requests from Relay, processes them, and sends them back to Relay. Server1 is entered when send_c1 becomes true. The Server1 mode is exited after 68 time units.

B. Formal analysis goals

As previously mentioned, the demonstrator used as the case-study in this paper addresses the telecommunications industry's trend of adopting the component-based system design paradigm, that is, construct the system out of reusable components, as a combination of in-house but also third party components. The most important requirement imposed on the demonstrator is to ensure an acceptable performance of the extension service, that is, handling 100 calls per second. For this to happen, and assuming a linear timing behavior per bursts of requests, the end-to-end response time of, say, 500 AAA requests should be less or at most 5 seconds.

To verify this, and, at the same time, validate the abstract descriptions, we have considered in our behavioral models (REMES and the corresponding PTA) the actual source code measured values of the authorization request, and authorization answer response times, respectively, as well as their respective CPU load, and memory usage, for each component of the demonstrator: Pen, DIAMETER client, DIAMETER relay, and DIAMETER server.

Before embarking upon formal validation, we need to check the absence of deadlocks, property specified in
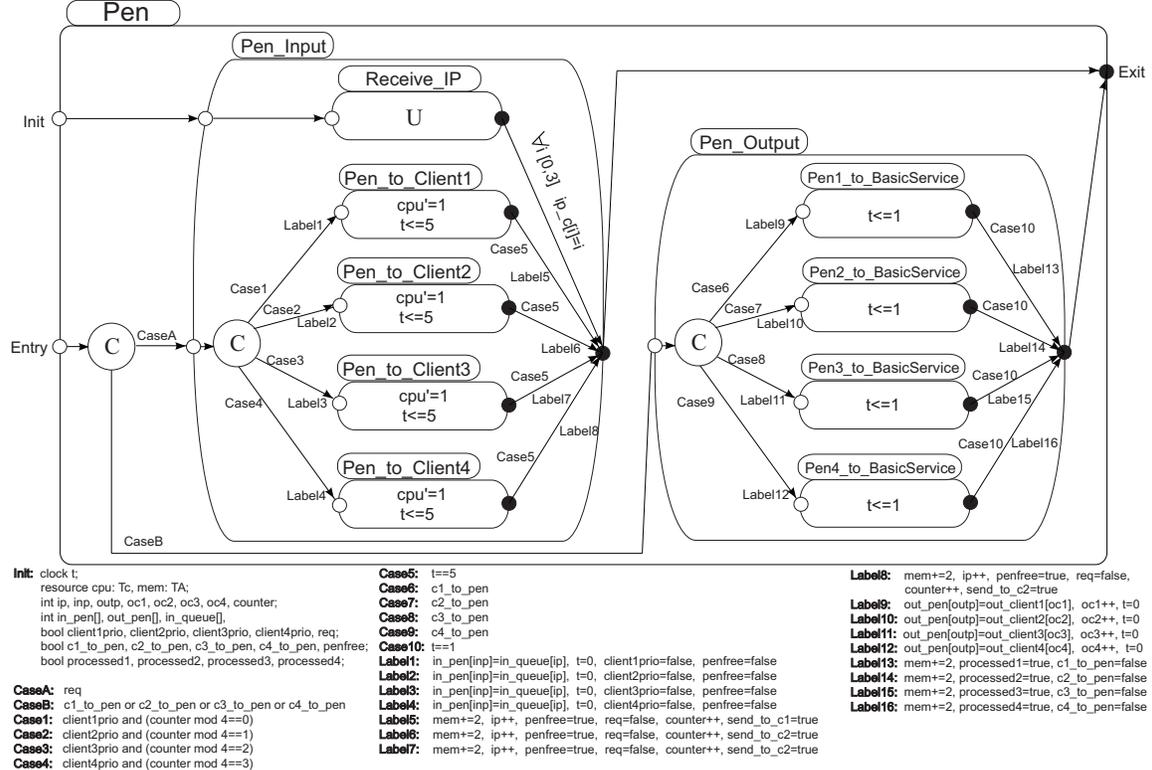
(a) The Pen component modeled in REMES.
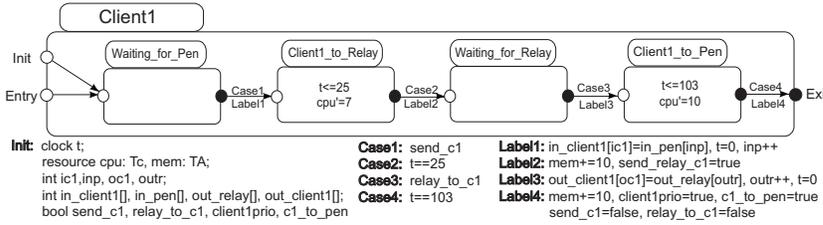
(b) The Client1 component modeled in REMES.
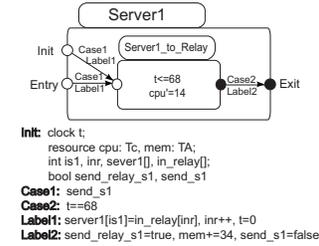
(c) The Server1 component modeled in REMES.

Figure 6. Pen, Client1 and Server1 modeled in REMES.

UPPAAL as follows:

$$A \square \; \text{not deadlock}$$

By using the measured values of the demonstrator's extra-functional attributes, we aim at:

- model-checking the REMES system model's capacity (number of handled requests per second), as well as
- computing an optimal execution trace for the overall consumption of resources (CPU and memory).

Verifying the demonstrator's capacity is a crucial performance requirement of the system, and we will next show that we actually deliver a performance guarantee, since model-checking is an exhaustive verification technique. To accomplish the response time verification, we define a global clock variable that stores the elapsed time from the start time of sending the authorization request to the Pen, until the request is served and returns to the call controller in the basic

service. Computing optimal resource-aware traces relies on a weighted sum representation of the resource function, which accounts for both types of resources simultaneously, allowing the designer to set the level of criticality for both resources (identical weights meaning equal importance).

C. PTA model of the ENT demonstrator and analysis results

We have analyzed the REMES-based ENT demonstrator, by semantically translating it into a network of PTA models, in UPPAAL CORA[3]. Here, we present only the PTA models of the Pen_Input submode and the Client1 mode, shown in Figure 7 and 8, respectively.

The PTA of Pen_Input has six locations: Start, Receive_IP, Waiting_for_BasicService, Pen_to_Client1,

[3]See the web page www.uppaal.org for more information about the UPPAAL CORA tool. UPPAAL CORA is a branch of the UPPAAL tool for cost optimal reachability analysis.

Pen_to_Client2, Pen_to_Client3 and Pen_to_Client4. The synchronization between Basic_Service and Pen_Input is modeled with channel req. Similarly, the synchronization between Pen_Input and Client1 is modeled with channel send_c1. The selection of the clients is controlled by the variables client1prio, ..., client4prio and counter.

The PTA of Client1 consists of five locations: Start, Waiting_for_Pen, Client1_to_Relay, Waiting_for_Relay and Pen_to_Client1. The synchronization between Client1 and Relay is modeled by using two channels send_relay_c1 (models requests sent from Client1 to Relay), and c1_to_relay (models responses sent from Relay to Client1). The synchronization between Client1 and Pen_Output is modeled by using channel c1_to_pen.

In our analysis model, we consider CPU to be a more critical resource than memory. The cost model that we use is derived from the measurements carried out on the actual source code. The resource-usage cost is influenced by the weights of CPU and memory, and the consumed resources of all transitions and locations. In the ENT demonstrator, we consider the following total cost function

$$c_{tot} = \mathsf{wcpu} \times c_{cpu} + \mathsf{wmem} \times c_{mem}$$

where $\mathsf{wcpu} = 2$ and $\mathsf{wmem} = 1$, and c_{cpu} and c_{mem} are the accumulated consumed amounts of CPU and memory, respectively.

After providing UPPAAL CORA with the PTA model of the ENT demonstrator, we were able to study the minimum cost reachability problem i.e., to find an execution trace of the system that results in the minimum possible total resource cost. For illustration, let's check for an optimal trace satisfying the reachability property: $E<>(\mathsf{processed}[3]==4)$, that is, a trace in which four requests are eventually processed by the ENT demonstrator. The execution trace that was found by UPPAAL CORA is presented in Figure 9 and the cost of this best trace is 173308. We use the value 781 ms (from code measurements) as time needed for the basic service to process 500 requests.

From our analysis model we were also able to determine the time needed for processing a certain number of requests. For response time only, we have performed the analysis in UPPAAL (in order to get a TA model we have removed the costs from the PTA model). UPPAAL has calculated that the time needed for handling 1 request is 3,42 time units (ms). If we consider that the processing time grows linearly, then for processing 500 requests our UPPAAL model of the ENT demonstrator needs 1710 time units. This number is just slightly higher than the source code measured value, that is, 1690 ms. The verification result shows that the capacity of the demonstrator (extension service alone) is actually greater than the required 100 requests per second. Also, this result concludes our behavioral model formal validation, regarding the end-to-end response time, which has been a central design issue of the demonstrator.

VII. RELATED WORK

Today there exist several languages and environments for modeling, verification and validation of embedded systems, such as CHARON [2], REMES and timed automata. Some of these languages and environments have been applied to real-sized industrial case studies.

CHARON is a modeling language for modular design of interacting hybrid systems. CHARON has been applied for modeling and analysis of a wide range of systems, such as automotive power trains and vehicle-to-vehicle control systems [17]. CHARON does not provide a mechanism for adding information about systems resource consumption. Accordingly, resource-wise properties can not be verified with CHARON.

Timed automata is a general purpose modeling framework that can be analyzed by using the UPPAAL tool-kit or the KRONOS [12] model checker. UPPAAL has been used for formal analysis of a number of industrial case-studies, which can roughly be divided in two classes: realtime controllers [19] and real-time communication protocols [7], [11], [15], [25]. The modeled systems have been validated and verified using the tool UPPAAL to satisfy the functionality and safety requirements on the system. However, resource-wise analysis have not been performed.

REMES has recently been used for behavioral modeling and analysis of an abstracted versions of a temperature control system [27], a car wash PLC controller [27] and an intelligent shuttle system [10]. For analysis purposes REMES models have been translated to (priced) timed automata and they have been verified in the UPPAAL tool-kit. The listed case studies have been checked for correctness against functional, timing and resource-wise requirements. In contrast to our case study, the resource-wise cost model used in the listed case studies has not come from the actual source code.

GIOTTO [14] is a time-triggered language and a tool-supported methodology for the development of distributed real-time embedded control systems. GIOTTO has been used in [21] for developing an implementation model of an elevator- and a hovercraft system. In order to perform analysis, GIOTTO models are translated to timed automata and verified in UPPAAL. The translation from GIOTTO to UPPAAL is an effort to bridge the gap between control law design and real-time implementation, by offering formal analysis at an intermediate layer (the GIOTTO model) that can check both platform independent (such as functionality and timing) and platform dependent properties (e.g., synchronization and scheduling). In contrast to REMES, GIOTTO does not consider resource-wise system behavior.

HYTECH [16] is a symbolic model checker for formal analysis of linear hybrid automata. It has been used for verification of an audio control protocol [16] and abstracted
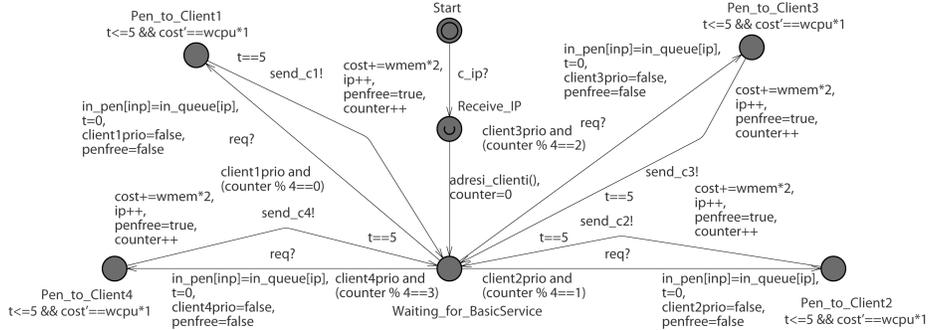
Figure 7. PTA model of the Pen_Input component.
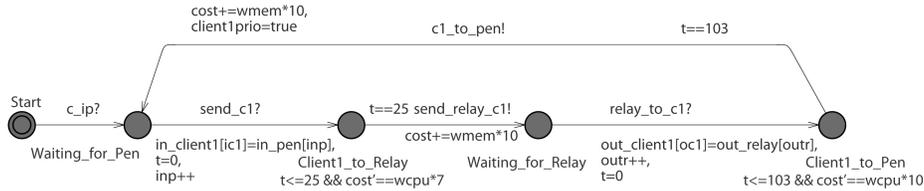
Figure 8. PTA model of the Client1 component.

automotive control system [26]. With HYTECH one can perform reachability verification, and parametric analysis. One drawback of HYTECH is that it is a text-only tool and as such is not particularly "user-friendly".

BIP [22], consists of a research-oriented component model and associated execution/simulation and verification tools. It focuses on verifying safety-, timing- (e.g., worst-case execution time or end-to-end delay) and synchronization properties. Several case studies have been carried out with BIP, such as MPEG4 encoder [22], TinyOS [5], and a robotic system called DALA [4].

VIII. CONCLUSIONS

In this paper, we have presented a case study where our recently introduced framework REMES is applied to model and analyze a new telecommunication system by Ericsson Nikola Tesla. The new system is horizontally developed by adding a newly developed authentication, authorization, and accounting service to a complex basic service telecom system consisting of several existing and reused components such as a DIAMETER standard protocol, an open-source *Pen* load balancer, and a number of servers.

As modeling result, we have shown how the system has been modeled in a component based fashion using the ProCom component modeling language, and how the functional, timing, and resource-wise behavior of the key components of the system have been modeled in REMES. We have also shown how the combined model is semantically translated into a network of (priced) timed automata to enable model-checking in the tools UPPAAL and UPPAAL CORA.

In the system analysis, we have, in addition to function and timing, considered a weighted sum of the resources CPU and memory, in which designers have chosen to consider CPU the most critical resource (twice the relative weight of memory). In this setting, we have derived an optimal system trace, the minimum time, and the minimal total accumulated weighted resource cost for processing a given number of system requests (in our case 1 and 4). Allowing the system designers to gain deeper understanding in the system's resource behavior might prove valuable to further optimize the system design, and adjust the resources provided by the underlying implementation platform, accordingly.

As future work on the ENT demonstrator formal analysis, we plan to model and verify other protocols for serving requests than round-robin. By checking possible performance in such other cases (like the first-in-first-out protocol), we could feed Ericsson researchers with important insights on the system's behavior, which might save unnecessary implementation time. We also intend to consider the case of heterogeneous servers, that is, processing the same request takes different time on each server, respectively.

REFERENCES

[1] R. Alur. Optimal paths in weighted timed automata. In *HSCC01: Hybrid Systems: Computation and Control*, pages 49–62. Springer, 2001.

[2] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierarchical Modeling and Analysis of Embedded Systems. *Proc. of the IEEE*, 8(3):231–274, 1987.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
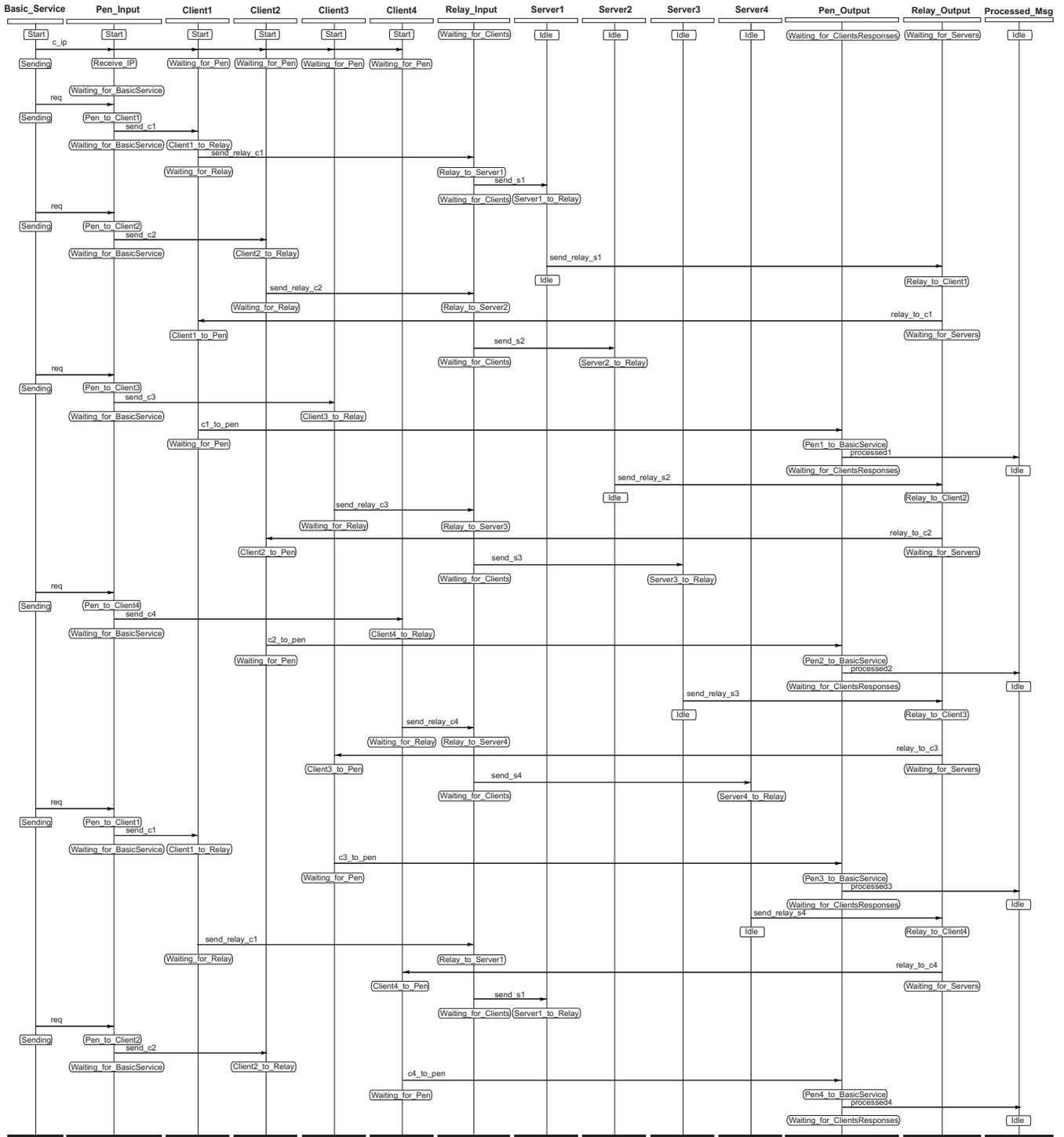
Figure 9.   Optimal trace for processing four requests.

[4] A. Basu, M. Gallien, C.Lesire, T-H. Nguyen, S. Bensalem, F. Ingrand, and J. Sifakis. Incremental Component-Based Construction and Verification of a Robotic System. In *Proc. of the 18th European Conference on Artificial Intelligence (ECAI 2008)*. IOS Press, 2008.

[5] A. Basu, L. Mounier, M. Poulhiès, J. Pulou, and J. Sifakis. Using BIP for Modeling and Verification of Networked Systems – A Case Study on TinyOS-based Networks. *Proc. of the 6th IEEE International Symposium on Network Computing and Applications*, 0:257–260, 2007.

[6] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-Cost Reachability for Priced Timed Automata. In *Proc. of the 4th International Workshop on Hybris Systems: Computation and Control, LNCS 2034*, pages 147–161. Springer–Verlag, 2001.

[7] O. Bengtsson, W. O. D. Griffioen, K. J. Kristoffersen, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Automated Analysis of an Audio Control Protocol Using UPPAAL. In *Journal of Logic and Algebraic Programming*, 2002.

[8] T. Brihaye, V. Bruyère, and J-F. Raskin. Model-checking for weighted timed automata. In *Proc. of FORMATS-FTRTFT04*, number 3253 in LNCS, pages 277–292. Springer–Verlag, 2004.

[9] Tomáš Bureš, Jan Carlson, Ivica Crnković, Séverine Sentilles, and Aneta Vulgarakis. ProCom – the Progress Component Model Reference Manual, version 1.0. Technical Report MDH-MRTC-230/2008-1-SE, Mälardalen University, June 2008.

[10] A. Čaušević, C. Seceleanu, and P. Pettersson. Modeling and Reasoning about Service Behaviors and their Compositions. In *Proc. of 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA 2010)*. Springer LNCS, October 2010.

[11] A. David and W. Yi. Modelling and Analysis of a Commercial Field Bus Protocol. In *Proc. of 12th Euromicro Conference on Real-Time Systems*, pages 165–172. IEEE CS Press, 2000.

[12] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with Kronos. In *Proc. 1995 IEEE Real-Time Systems Symposium, RTSS'95*, pages 66–75. IEEE CS Press, 1995.

[13] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli. Design of embedded systems: formal models, validation, and synthesis. *Proc. of the IEEE*, 85(3):366–390, 1997.

[14] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: a Time-triggered Language for Embedded Programming. Technical report, Berkeley, CA, USA, 2001.

[15] A. Hessel and P. Pettersson. Model-based Testing of a WAP Gateway: an Industrial Case-Study. In *International Workshop on Formal Methods for Industrial Critical Systems (FMICS'06)*, pages 116–131. Springer-Verlag, February 2006.

[16] P-H. Ho and H. Wong-Toi. Automated Analysis of an Audio Control Protocol. In *Proc. of Computer Aided Verification (CAV'95)*, pages 381–394. Springer-Verlag, 1995.

[17] F. Ivančić. Report on Verification of the MoBIES Vehicle-Vehicle Automotive OEP Problem, 2002.

[18] D. Ivanov, M. Orlić, C. Seceleanu, and A. Vulgarakis. REMES Tool-chain: A Set of Integrated Tools for Behavioral Modeling and Analysis of Embedded Systems. In *Proc. of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 361–362, 2010.

[19] M. Lindahl, P. Pettersson, and W. Yi. Formal Design and Analysis of a Gear-Box Controller. In *Proc. of the 4th Workshop on Tools and Algorithms for the Construction and Analysis of Systems, LNCS 1384*, pages 281–297. Springer–Verlag, 1998.

[20] D. Matijašević, I. Gizdić, and D. Huljenić. Mechanisms for Diameter service performance enhancement. In *Proc. of the 17th International Conference on Software, Telecommunications and Computer Networks - SoftCOM*, 2009.

[21] R. K. Poddar and P. Bhaduri. Verification of Giotto based embedded control systems. *Nordic Journal of Computing*, 13:266–293, December 2006.

[22] M. Poulhiès, J. Pulou, C. Rippert, and J. Sifakis. A Methodology and Supporting Tools for the Development of Component-Based Embedded Systems. In *Proc. of the 13th Monterey conference on Composition of embedded systems: scientific and industrial issues*, pages 75–96, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] C. Seceleanu, A. Vulgarakis, and P. Pettersson. REMES: A Resource Model for Embedded Systems. In *Proc. of the 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009)*. IEEE CS, June 2009.

[24] S. Sentilles, A. Vulgarakis, T. Bureš, J. Carlson, and I. Crnković. A component model for control-intensive distributed embedded systems. In *Proc. of the 11th International Symposium on Component Based Software Engineering (CBSE2008)*, pages 310–317. Springer, Oct 2008.

[25] D. P. L. Simons and M. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. *STTT*, 3(4):469–485, 2001.

[26] T. Stauner, O. Müller, and M. Fuchs. Using HyTech to Verify an Automotive Control System. In *Proc. Hybrid and Real-Time Systems (HART'97), LNCS 1201*, pages 139–153. Springer-Verlag, 1997.

[27] A. Vulgarakis and A. Čaušević. Applying REMES behavioral modeling to PLC systems. In *Proc. of 22nd International Symposium on Information, Communication and Automation Technologies - ICAT 2009*. IEEE, October 2009.

[28] A. Vulgarakis, S. Sentilles, J. Carlson, and C. Seceleanu. Integrating Behavioral Descriptions into a Component Model for Embedded Systems. In *Proc. of the 36th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, September 2010.

[29] M. Zemljić, I. Skuliber, and S. Dešić. Utilization of Open-Source High Availability Middleware in Next Generation Telecom Services. In *Proc. of the 17th International Conference on Software, Telecommunications and Computer Networks - SoftCOM*, 2009.