ARC Collaborative
(TP)I : (Timed/Probabilistic) Interfaces

Aalborg - Nantes - Rennes (INRIA)

Summary

Several industrial sectors involving complex embedded systems have recently experienced deep changes in their organization, aerospace and automotive being the most prominent examples. In the past, they were organized around vertically integrated companies, supporting in-house design activities from specification to implementation. Nowadays, systems are tremendously big and complex, and it is almost impossible for one single team to have the complete control of the entire chain of design from the specification to the implementation. In fact, complex systems now result from the assembling of several components. These many components are in general designed by teams, working *independently* but with a common agreement on what the interface of each component should be. Such an interface precises the behaviors expected from the component as well as the environment in where it can be used. The main advantage is that it does not impose any constraint on the way the component is implemented, hence allowing for independent implementation. According to state of practice, interfaces are typically described using Word/Excel text documents or modeling languages such as UML/XML. We instead recommend relying most possibly on mathematically sound formalisms, thus best reducing ambiguities. We propose to study mathematical formalisms for interface theories with all the operators needed to reason on them in a proper way, i.e., composition, refinement, quotient, and dissimilar alphabets. We will also focus on implementation (integration in *UPPAAL* and *INTERSMV toolsets*) and efficient algorithms. Our results will be extended to timed and stochastic systems. Finally, we will also consider concurrency and modular verification.

1 Structure of the document and summary of the proposition

This document is a proposal for an ARC Collaborative ; it is divided into four sections. In Section 2, we present the partners involved in the proposition. In Section 3, we present our research proposal on interface theories. The proposal contains a brief state-of-the art, our objectives, a research calendar, and the competences of the partners. Section 4 discusses the meetings and the costs. Finally Section 5 briefly discusses related research projects.

2 The teams

The project will be leaded by Axel Legay, CR2 at INRIA/IRISA Rennes since December 2008. Three teams are involved in the application : S4 (INRIA, Rennes), IRCCyN (Nantes), and Aalborg (Denmark). Competences provided by the members of these teams will be emphasized in the research proposal (see Section 3.5). The rest of this section provides a brief description of the teams.

1. The S4 Team at INRIA/IRISA Rennes. The team is leaded by Benoît Caillaud. Members of the S4 team that will be involved in the project : Axel Legay, Benoît Delahaye (PhD Student), and Benoît Caillaud. Claude Jard, professor at ENS Cachan and member of the DISTRIBCOM team at IRISA/INRIA Rennes will also be involved in the project.
2. IRCCyN Nantes. Members of the IRCCyN team involved in the project : Didier Lime and Olivier H. Roux.

3. Aalborg Team leaded by Kim Larsen in Denmark. Members of the Aalborg team that will be involved in the project : Kim Larsen, Alexandre David. Mikkel Pedersen, PhD student of the team will also be involved. Andrzej Wasowski, an associate professor at IT Unviersity of Copenhagen and collaborator of Kim Larsen within the *MT-LAB project* (`www.mtlab.dk`) will also be involved.

Remark 1 *Other members of the S4 and the VERTECS teams at Rennes such as Albert Benveniste and Nathalie Bertrand and Sophie Pinchinat have shown interest to our project and will punctually contribute.*

3 The project

3.1 Scientific focus

Several industrial sectors involving complex embedded systems have recently experienced deep changes in their organization, aerospace and automotive being the most prominent examples. In the past, they were organized around vertically integrated companies, supporting in-house design activities from specification to implementation.

Nowadays, systems are tremendously big and complex, and it is almost impossible for one single team to have the complete control of the entire chain of design from the specification to the implementation. In fact, complex systems now result from the assembling of several components. These many components are in general designed by teams, working *independently* but with a common agreement on what the interface of each component should be. Such an interface precises the behaviors expected from the component as well as the environment in where it can be used. The main advantage is that it does not impose any constraint on the way the component is implemented :

> *Several components can be implemented by different teams of engineers providing that those teams respect the interfaces on which all of them agree.*

According to state of practice, interfaces are typically described using Word/Excel text documents or modeling languages such as UML/XML. We instead recommend relying most possibly on mathematically sound formalisms, thus best reducing ambiguities. Mathematical foundations that allow to reason at the abstract level of interfaces, in order to infer properties of the global implementation, and to design or to advisedly (re)use components is a very active research area, known as *compositional reasoning* [45]. Aiming at practical applications *in fine*, the software engineering point of view naturally leads to the following requirements for a good theory of interfaces.

Remark 2 *In the rest of the document, one will use the following equivalences (depending on the context) : specification = interface ; implementation = component.*

1. It should be decidable whether an interface admits an implementation (a model). This means that one should be capable to decide whether the requirements stated by the interface can be implemented. One should also be capable to synthesize an implementation for such an interface. In our theory, an implementation shall not be viewed as a programming language but rather as a mathematical object that represents a set of programming languages sharing

common properties. The ability to decide whether a given component implements a given interface is of clear importance, and this must be performed with efficient algorithms.

2. It is important to be able to replace a component by another one without modifying the behaviors of the whole design. At the level of interfaces, this corresponds to the concept of *Refinement*. Refinement allows one to replace, in any context, an interface by a more detailed version of it. Refinement should entail substitutability of interface implementations, meaning that every implementation satisfying a refinement also satisfies the larger interface. For the sake of controlling design complexity, it is desirable to be able to decide whether there exists an interface that refines two different interfaces. This is called *shared refinement*. In many situations, we are looking for the *greatest lower bound*, i.e., the shared refinement that could be refined by any other shared refinement.

3. Large systems are concurrently developed for their different *aspects* or *viewpoints* by different teams using different frameworks and tools. Examples of such aspects include the functional aspect and the safety aspect. Each of these aspects requires specific frameworks and tools for their analysis and design. Yet, they are not totally independent but rather interact. The issue of dealing with multiple aspects or multiple viewpoints is thus essential. This implies that several interfaces are associated with a given component, namely (at least) one per viewpoint. These interfaces are to be interpreted in a conjunctive way. This conjunction operation should satisfy the following property :

   > *Given two view-points represented by two interfaces, any implementation that satisfies the conjunction must satisfy the two view-points.*

4. The interface theory should also provide a combination operation, which reflects the standard interaction/composition between systems. In practice, one should be capable to decide whether there exists at least one environment in where two components can work together, i.e., in where the composition makes sense. Another, but more difficult, objective is to synthesize such an environment. Finally, the composition operation should satisfy the following property :

   > *Given two components satisfying two interfaces, the theory must ensure that the composition of the two components satisfies the composition of their corresponding interfaces.*

5. A quotient operator, dual to composition is of interest to perform incremental

design. Intuitively, the quotient enables to describe a part of a global specification assuming another part is already realized by some component.

6. Complex systems are built by combining components possessing dissimilar alphabets for referencing ports and variables. It is thus important to properly handle those different alphabets when combining interfaces.

All the above operations and properties should be checked and performed with efficient algorithms. In addition, a good interface theory should have the *independent implementability property*. More precisely, the operations of conjunction and composition must be associative, meaning that the systems can be composed in any order. Moreover, those operations should be stable with respect to refinement.

Our main objective is to provide new mathematical foundations for interface theories. The rest of the document is divided into four sections. In Section 5, we will discuss existing results on interface theories. In Section 3.3, we will describe our research proposal. Sections 3.4 and 3.5 discuss the calendar and the competences, respectively.

3.2 State-of-the-art (brief account on)

Building good interface theories has been the subject of intensive studies (see, e.g., [45, 33, 19, 37, 43, 32, 35]) and the concept find applications in a wide range of areas including *web services* [16] and *product lines* [56, 48]. Recently, two models have been emphasized : (1) *interface automata* [33] and (2) *modal specifications* [47]. Interface automata is a game-based variation of the well-known model of input/output automata [54] which deals with open systems, their refinement and composition, and puts the emphasis on interface compatibility. Modal specifications is a language theoretic account of a fragment of the modal mu-calculus logic [42] which admits a richer composition algebra with product, conjunction and quotient operators.

Modal specifications correspond to *deterministic* modal automata [47], i.e., automata whose transitions are typed with *may* and *must* modalities. A modal specification thus represents a set of implementations ; informally, a must transition is available in every component that implements the modal specification, while a may transition needs not be. The components that implement modal specifications are prefix-closed languages, or equivalently deterministic automata/transition systems.

Satisfiability of modal specifications is decidable. Refinement between modal specifications coincides with language inclusion [47]. Conjunction is effectively computed via a product-like construction. It can be shown that the conjunction of two modal specifications correspond to their greatest common refinement. Combination of modal specifications, handling synchronization products *à la* Arnold and Nivat [7], and

the dual quotient operator can be efficiently handled in this setting [58, 59].

In interface automata [33], an interface is represented by an input/output automaton [54], i.e., an automaton whose transitions are labeled with *input* or *output* actions. The semantics of such an automaton is given by a two-player game : an *Input* player represents the environment, and an *Output* player represents the component itself. Interface automata do not encompass any notion of implementation, because one cannot distinguish between interfaces and implementations.

Refinement between interface automata corresponds to the alternating refinement relation between games [5], i.e., an interface refines another if its environment is more permissive whereas its component is more restrictive. Shared refinement is defined in an ad-hoc manner [36] for a particular class of interfaces [27]. Contrary to most interfaces theories, the game-based interpretation offers an *optimistic* treatment of composition : two interfaces can be composed if there exists at least one environment (i.e., one strategy for the Input player) in which they can interact together in a safe way (i.e., whatever the strategy of the Output player is). This is referred to as compatibility of interfaces. A quotient, which is the adjoint of the game-based composition, has been proposed in [18].

It is worth mentioning that, in existing work on interface automata and modal specifications, there is nothing about dissimilar alphabets. This is somehow surprising as it seems to be a quite natural question when performing operations that involve several components, e.g., conjunction, composition, and quotient. In fact, as stated in [60], an explicit mechanism to handle dissimilar alphabets is not needed when considering interface automata, since conjunction is not discussed for this model. For the case of composition/quotient, instead, we shall see that the notion is implicitly encompassed in the definition of compatibility. Conjunction and quotient operators [47, 58, 59] that have been proposed for modal specification do not take dissimilar alphabet into account.

In conclusion, both models have advantages and disadvantages :

- Interface automata is a model that allows designers to make assumptions on the environment, which is mainly useful to derive a rich notion for composition. In addition, the notion of dissimilar alphabets is not needed. Unfortunately, the model is incomplete as conjunction and shared refinement are not defined.
- Modal specification is a rich language algebra model on which most of requirements for a good interface theory can be considered. Unfortunately, *may* and *must* modalities are not sufficient to derive a rich notion for composition including compatibility. Moreover, the notion of dissimilar alphabets is missing.

In a recent work, we have proposed a new model called *modal interfaces* [48, 60], which combines advantages of modal specifications and interface automata. Roughly speaking, modal interfaces are modal specifications combined with the composition operation developed for interface automata. The idea is to typeset the may and must transition with Inputs and Outputs that are used only when performing the composition operation and barely ignored otherwise. The model also encompasses the concept of dissimilar alphabets. We have proposed efficient algorithms for computing composition, conjunction, and quotient as well as to check for refinement and satisfiability. The model of modal interfaces (and several extensions) is implemented in a tool called *INTERSMV* (the first version of the tool will be released in January 2010 for the review of the European Project COMBEST [30]). The tool relies on powerful symbolic representations (combination of Binary Decision Diagrams [24]) that allow for conciseness and efficiency. *INTERSMV* is currently evaluated on several case studies coming from two European projects : COMBEST [30] and SPEEDS [61].

Remark 3 *As we shall see in Section 3.3.2, probabilistic and timed extensions of interface theories also exist. However, the development of these models does not reach the maturity of the one of modal interfaces.*

3.3 Our objectives

Our objectives are (1) to improve the theory of modal interfaces and (2) to study extensions that will allow to model a broader class of systems as well as to express more interactions between them. Our research proposal can be divided in four main parts :

1. In Section 3.3.1, we propose several directions to improve existing results on modal interfaces. This includes heuristics to increase the efficiency of the algorithms as well as generalizing existing operators.
2. In Section 3.3.2, we propose several extensions of interface theories in order to take stochastic and timed behaviors into account.
3. In Section 3.3.3, we suggest to use interface theories in order to verify a system by looking at its various components.
4. In Section 3.3.4 we propose a new operator to handle concurrency in an explicit manner.

The two first parts of the proposal suggest extensions of the work we conducted over the two last years. The last two points suggest new research directions.

3.3.1 Improving the modal interfaces framework

As stated above, the algorithms developed for modal interfaces are already quite efficient. However, we believe that there is still room for improvements. As

an example, most of existing algorithms for computing the composition of two interfaces do not try to reduce the size[1] of the result. It is thus not difficult to find situations where the composition algorithm produces a huge automaton while a very small one also exists [17, 38]. We will investigate heuristic methods such as learning algorithms [6] (already deployed for interface automata [17, 38]) or sat-solvers [17] to improve the composition operator.

Our implementation of modal interfaces can incorporate variables. This is a feature that is not present in most of existing interface theories. Those variables can be used, as an example, to model global and shared resources [32]. Due to the implementation and for decidability reasons, those variables have to be range-bounded. It is however sometimes more convenient to use an infinite-state representation (see [64] for an overview). As an example, consider a system manipulating integer variables ; it may be easier to assume that those variables are not bounded rather than fixing an arbitrary maximal value[2] for them [21, 11, 12]. Another example is the one of communication protocols : it may be better to assume that the number of places in a communication channel is not bounded rather than assuming an arbitrary bound which may not reflect the reality [20]. We believe that all the algorithms developed for modal interfaces extend to (semi-)algorithms for infinite-state interfaces. For doing so, we propose to use principles that are similar to those introduced in [22, 23, 34, 10, 1].

Finally, we would like to extend the theory in order to handle nondeterministic behaviors. Nondeterminism may arise when the system has to take internal decisions that are not visible to the external world. A part of the theory of interface automata and modal specifications already extends to this setting. However, there is nothing for the quotient operator.

Remark 4 *There exists several other works on quotient for nondeterministic systems (e.g., [8, 49, 65]). It is worth mentioning that those works propose techniques that are either capable to synthesize an formula or a system, but not a modal interface*[3]*.*

3.3.2 Probabilities and time

Stochastic and timed aspects have not been studied in details for interface theories. Those aspects are however of crucial interest. Time can be a crucial parameter in practice, for example in embedded systems. Probabilities can be used to ensure fairness and robustness of communication systems as well as to model faults.

[1] In terms of number of states.

[2] The size of the range would depend on the architecture.

[3] Not even a modal specification.

In a very recent work [25], we have proposed what seems to be the first complete interface theory for stochastic systems. In our theory, systems are represented by Markov Chains and interfaces by *Constraint Markov Chains* (CMCs in short). CMCs are Markov Chains whose probability distributions are replaced by constraints that represent a set of probability distributions[4]. The model thus allows to finitely represent a possibly infinite set of Markov Chains.

CMCs's theory currently suffers from a major drawback : it does not allow for non stochastic behavior. This is problematic since it is well-known that many models require to mix stochastic and nonstochastic behaviors.

We propose to enrich CMCs with nonstochastic behaviors by adding may and must transitions to the model. In the new model, one will distinguish between two types of states : (1) the stochastic states in where one moves to some next state with a given probability (example : there is a probability that the system moves with some fault and a probability that it moves correctly), and (2) the nondeterministic states in where one may or must move to some next state, depending of the choice made by the component/environment. This new extension, which we will called Constraint Modal Markov Decision Processes (CMMDPs in short), corresponds to an interface theory for Markov Decision Processes (such a theory does not yet exists). We believe that the algorithms and the theory for CMMDPS could be obtained from a combinations of those defined for CMCs and those defined for modal specifications. The main difficulty will be to make this combination in such a way that the new model satisfies all good requirements for an interface theory as well as the independent design property. We also plan to give a game-based flavor to the composition operation. Like for modal interfaces, this should be done by equipping the nonprobabilistic transitions with Inputs and Outputs modalities.

As a second step, we plan to move from Markov Decision Processes to *Probabilistic timed automata* [46], i.e., timed automata whose discrete transitions may be enriched with stochastic behaviors. For doing so, we will enrich CMMDPS with clocks. This will mainly impact the semantic of the may and must transitions. We already studied *modal event-clock specifications*, a timed extension of modal specifications [15, 14]. We propose to marry the algorithms developed for modal event-clock specifications with those we will propose for CMMDPS. In a second step, we also plan to study a timed extension the modal interface theory, i.e., an extension of timed modal specification with Inputs and Outputs modalities in addition to the may and must modalities. In this context, we propose to define a new timed game to perform the composition operation. Our definitions should ensure that one cannot win the game with a strategy that blocks the time. It is known that considering non-blocking strategies makes it harder to ensure the independent implementability

[4]As an example, each transition can be equipped with an interval and the probability to take this transition is any value in the interval [41].

property even for the simple case of interface automata [35, 31].

The new theories will be implemented in the *UPPAAL* Toolset [63]. *UPPAAL* is a tool for the specification and the verification of timed systems, which has recently been enriched with probabilistic timed automata. *UPPAAL* is under development and maintained since more than ten years. The tool has been downloaded 30 000 times and successfully applied to various case studies submitted by industrial partners [5] (NASA, CNES, Bosch, ABB, ...). Its development is strategic for Aalborg university. *UPPAAL* provide efficient data structures to manipulate probabilistic timed automata. The tool also proposes several game-based algorithms for checking composition of timed automata. We believe that these features will serve as a good basis to develop efficient algorithms and data structures for our new models.

Remark 5 *We will not implement the probabilistic/timed extension of interface theories in* INTERSMV*. Indeed,* INTERSMV *is a tool that uses efficient data structure and symbolic representations that are designed for nondeterministic systems only. We have no hope that such structures can be adapted to time and probabilistic systems.*

Remark 6 *We hope that some of the results we will obtain in Section 3.3.1 will also extend to the new models discussed in this section. However, we will not consider the quotient operator. Indeed, this operator is not defined for CMCs and we believe that it is a very difficult problem that deserves its own research project.*

3.3.3 Modular verification and link with temporal logics

We believe that it is important to provide engineers with a mechanism to check whether the interfaces they specify satisfy some requirements written in a concise language. For doing so, we propose to express such requirements with temporal logics [4, 57, 29, 3, 28]. One should thus be capable to decide whether an interface satisfies a given property expressed in such a logic – which also implies that any implementation of the interface should also satisfy the property. Moreover, one should be capable to decide whether the whole design satisfies a given property only by observing the properties satisfied by its components. This approach is called *modular verification*. It is a very important feature as it allows to reduce the complexity of the verification process [6]. We propose to enrich our models with a (modular) verification procedure. This is a tedious task, especially when taking stochastic and timed behaviors into account (see [40] for discussions regarding the case of stochastic systems).

[5] The tool offers nice user interfaces that allow for a direct use by industrials/engineers. Those interfaces are definitively part of the success of *UPPAAL*.

[6] There are also situations where the verifications techniques are not powerful enough to work on the whole design, but can give an answer when considering the components separately.

Observe that Modular verification already exists at the level of components through the conjunction operation : the conjunction of two components must satisfy the conjunction of their corresponding interfaces. Here we go one step further since we consider the level of interfaces rather than the one of components. Other alternatives exist. As an example, one could imagine to use temporal logics as a formalism for representing interfaces. However, we believe that operations such as composition or refinement are much more easier to understand at the level of automata than at the one of formula.

One of the major difficulties here will be to ensure that if an interface satisfies some property, then the property is also satisfied by any component that is an implementation of the interface. This is a hard problem, especially when considering combination operators such as conjunction, composition, or refinement.

Remark 7 *The tool* Chic *[27] proposes to specify properties of interface automata with the* Alternating Temporal Logic *[4]. However, there is nothing about modular verification for such interfaces.*

Remark 8 *The above proposal does not discuss the choice of the temporal logic that will be used. This choice will mostly depend on the model under consideration.*

Another interesting problem related to temporal logics is the one that consists in synthesizing an interface from a given property. It would also be of interest to be capable to synthesize a composition for two interfaces and to ensure that this composition also satisfies some property. Up to now, the composition operation defined for interface automata (and hence for modal interfaces) only ensures that their exists an environment in where the two interfaces can work together. This environment may either be useless or too permissive, hence the use of temporal logic to make hypothesis on its behaviors.

Remark 9 *There are a lot of work on synthesizing an automaton from a formula but, to the best of our knowledge, nothing has been done on modal interfaces or on its probabilistic/timed variants. What we suggest for composition is also definitively new.*

Finally, model checking temporal properties of interfaces raises the problem of describing counter-examples for those cases where the formula is not satisfied. Finding a nice way to report such counter-examples to the user should also be investigated.

3.3.4 Concurrency made explicit

The composition operator naturally introduces concurrency into the system but the usual analysis techniques will explicitly or implicitly compute the product of all the components, thus destroying this concurrency. We will therefore investigate the possibility of maintaining the concurrency of the interfaces expressed as multiple components through the use of true concurrency semantics and unfoldings [39, 55]. One possible interest of considering such *concurrent interfaces* is to explicit causal dependencies, which could be of great importance in specifications. The question of composition, conjunction, refinement, and even causalities [62, 66] raise a lot of original problems when considering this new operator. Observe also that maintaining concurrency may allow to avoid building a huge automaton for the composition operation (see Section 3.3.1).

Remark 10 UPPAAL *already allows for an explicit representation of concurrence. However, this is only for the composition defined on* timed automata *[3] or the one defined on* timed input/output automata *[44, 13]. Those compositions are much simpler than the one defined on interfaces.*

3.4 Calendar

Some results on improving modal interfaces and timed/probabilistic extensions should be obtained at the end of the first year. We are quite confident since (1) we already have some preliminary results on stochastic and timed systems that could be extended, and (2) *INTERSMV* should serve as a good basis to observe interactions between components and to develop heuristics to improve the composition operation. Nondetermism and infinite-state extensions should be investigated during the second year.

Modular verification and concurrency, which are new directions in this area, will be studied in parallel. We will start with modal interfaces and, if successful, we will move to timed/stochastic extensions.

3.5 Competences of partners and interactions

By reading the research proposal, one can see that competences are needed in (at least) five areas : (1) interface theories, (2) timed systems, (3) probabilistic systems, (4) concurrent systems, and (5) implementation of mathematical theories with a practical evaluation. We now give more insights regarding the competences provided by each partner and the existing collaborations between them. It is worth mentioning that collaborations already exist between all the teams.

1. Both Axel Legay and Benoît Caillaud are experts in interface theories on which they wrote several papers. In addition to the models of modal interfaces and timed modal specifications, Axel Legay has collaborated with NASA and is the author of a tool called TICC [2]. This tools implements the theory of interface automata (with the exception of the quotient operator) and one of its extensions called *sociable interfaces* [32]. He also proposed several new model checking techniques for infinite-state and stochastic systems as well as new algorithms to solve (timed) games. Those algorithms have been implemented in three tools : LASH [50], T(O)RMC [51], and APEX [52]. Benoît Caillaud has been working on topics related to the synthesis and control of concurrent systems. He is the author of the *Synet Petri-net synthesis software* [9], a tool which has found applications in work-flow engineering and communicating distributed control synthesis. Claude Jard is an expert in concurrency, timed systems, and unfolding theory. Recently, in cooperation with Thomas Chatain, he solved the problem of constructing finite complete prefixes of unfoldings of safe time Petri nets. A similar work has also been done for networks of timed automata and is based on symbolic representations [26]. His current interest is the introduction of parameters and the use of unfolding in dynamical verification (supervision) in cooperation with the IRCCyN's group.
2. Both Didier Lime and Olivier H. Roux are experts in (concurrent) timed systems and their implementation. They are the core developers of the Romeo tool [53] for the verification of time Petri nets. They have been doing some recent work on symbolic unfoldings with Claude Jard. Didier Lime is also expert in timed games and participates in the development of *UPPAAL* with Aalborg, especially *UPPAAL-Tiga*, the flavor of Uppaal dedicated to timed games.
3. Kim Larsen and his team are experts in the area of timed systems. Together with various team members, Larsen developed and promoted the *UPPAAL* toolset. Kim Larsen has ongoing collaborations with S4 on interface theories (especially stochastic and timed extensions [25, 31]). He also has a wide knowledge of probabilistic systems and probabilistic timed automata.

The three teams share common competences but are also complementary. As an example, Rennes is the leader for interface theories and the development of *INTER-SMV* but it needs competences from IRCCyN when considering (concurrent) timed systems, especially when switching to implementation in *UPPAAL* and modular verification. This knowledge is easily exploitable due to the geographic proximity of Nantes and Rennes. Aalborg is an indisputable partner. Indeed, they will provide other teams with *UPPAAL*, and they are the one who have the expertise on probabilistic timed automata and their implementation.

4 Meetings and Costs

We plan to have bimensual meetings as well as several visits between the members of the different teams. We will also organize a workshop to present our results at the end of the second year. Regarding the costs, we would like to ask for 20000 euros for the first year and between 20000 and 25000 euros for the second year (the 5000 euros difference would be for the organization of the workshop).

Since the teams involved in the project are not located in the same site/city/country, we believe that a postdoctoral student will be of a great help in order to coordinate a part of the work. The student will work full time on the project and she/he will mainly focus on probabilistic and timed extensions (see Section 3.3.2). One of her/his main duty will be to be responsible for the development of efficient algorithms for handling such models. She/he will also be responsible for the implementation in *UPPAAL*. The postdoc will be employed by the S4 team at Rennes and she/he will have to visit the other teams.

In this proposition, implementation represents a substantial amount of work. In order to help us to make this objective, we will also apply for an engineer (ADT 2010). The engineer will work in close collaboration with the *UPPAAL* team. She/he will also participate to the development of *INTERSMV*.

As a summary, we ask for money to (1) employ a postdoctoral student, and (2) organize the meetings and the visits (between 40000 and 45000 euros). We will apply for the funding of a software engineer (ADT2010), who will contribute to the implementation of the results of the ARC in *UPPAAL* and *INTERSMV*.

5 Related and forthcoming projects

This ARC proposal is the continuation of the work done by the S4 team on interface theories for the European Projects COMBEST [30] and SPEEDS [61]. The tool *INTERSMV* is one of the main release of INRIA for the COMBEST project. Our recent experience shows that compositional design reasoning with interface theories raises a lot of interest from industrial partners (Airbus, EADS, IAI, and SAAB for SPEEDS and COMBEST projects). We are thus convinced that our implementations in the *UPPAAL* and *INTERSMV* toolsets will be used in many forthcoming projects.

As we already said in Section , interface theories find applications in a wide range of areas including web services [16] and product lines [56, 48]. Axel Legay and Kim Larsen and Andrzej Wasowski are currently discussing a submission to a FET open call for a European project on web services, product lines, and reliable systems. The results we will obtain with this ARC, especially on probabilistic and timed extensions, will certainly influence the write up of this submission.

Références

[1] P. A. Abdulla, A. Bouajjani, and J. d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1) :153–169, 2008.

[2] B. T. Adler, L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy. Ticc : A tool for interface compatibility and composition. In *Proc. 18th Int. Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 59–62. Springer, 2006.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science.*, 126(2) :183–235, 1994.

[4] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5) :672–713, 2002.

[5] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *Proc. 9th Int. Conference on Concurrency Theory (CONCUR)*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.

[6] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2) :87–106, 1987.

[7] A. Arnold and M. Nivat. Metric interpretations of infinite trees and semantics of non deterministic recursive programs. *Theoretical Comput. Sci.*, 11, 1980.

[8] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1) :7–34, 2003.

[9] E. Badouel, B. Caillaud, and P. Darondeau. Distributing finite automata through petri net synthesis. *Journal on Formal Aspects of Computing*, 13 :447–470, 2002.

[10] C. Baier, N. Bertrand, and P. Schnoebelen. On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In *Proc. 13th Int. Conference on Logic for Programming, Artificial Intelligence,*

and Reasoning (LPAR), volume 4246 of *Lecture Notes in Computer Science*, pages 347–361. Springer, 2006.

[11] S. Bardin, A. Finkel, and J. Leroux. Faster acceleration of counter automata in practice. In *Proc. 10th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lecture Notes in Computer Science*, pages 576–590. Springer, 2004.

[12] S. Bardin, A. Finkel, J. Leroux, and P. Schnoebelen. Flat acceleration in symbolic model checking. In *Proc. 3th Int. Conference on Automated Technology for Verification and Analysis (ATVA)*, volume 3707 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2005.

[13] J. Berendsen and F. W. Vaandrager. Compositional abstraction in real-time model checking. In *Proc. 6th Int. Convference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 5215 of *lncs*. Springer, 2008.

[14] N. Bertrand, A. Legay, S. Pinchinat, and J.-B. Raclet. A compositional approach on modal specifications for timed systems. In *Proc. 11th Int. Conference on Formal Engineering Methods (ICFEM)*, Lecture Notes in Computer Science. Springer, 2009. to appear.

[15] N. Bertrand, S. Pinchinat, and J.-B. Raclet. Refinement and consistency of timed modal specifications. In *Proc. 3rd Int. Conference on Language and Automata Theory and Applications (LATA)*, volume 5457 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2009.

[16] D. Beyer, A. Chakrabarti, and T. A. Henzinger. Web service interfaces. In *Proc. 14th int. Conference on World Wide Web (WWW)*, pages 148–159. ACM, 2005.

[17] D. Beyer, T. A. Henzinger, and V. Singh. Algorithms for interface synthesis. In *Proc. 19th Int. Conference on Computer Aided Verification (CAV)*, volume 4590 of *Lecture Notes in Computer Science*, pages 4–19. Springer, 2007.

[18] P. Bhaduri. Synthesis of interface automata. In *Proc. 3rd Automated Technology for Verification and Analysis Conference (ATVA)*, volume 3707 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2005.

[19] S. Bliudze and J. Sifakis. A notion of glue expressiveness for component-based systems. In *Proc. 19th Int. Conference on Concurrency Theory (CONCUR)*, volume 5201 of *Lecture Notes in Computer Science*, pages 508–522. Springer, 2008.

[20] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds (extended abstract). In *Proc. 8th Int. Conference on Computer Aided Verification (CAV)*, volume 1102 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1996.

[21] B. Boigelot, S. Jodogne, and P. Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(3) :614–633, 2005.

[22] B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large (extended abstract). In *Proc. 15th Int. Conference on Computer Aided Verification (CAV)*, Lecture Notes in Computer Science, pages 223–235. Springer, 2003.

[23] B. Boigelot, A. Legay, and P. Wolper. Omega-regular model checking. In *Proc. 10th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lecture Notes in Computer Science*, pages 561–575. Springer, 2004.

[24] R. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Survey*, 24(3) :293–318, 1992.

[25] B. Caillaud, B. Delahaye, K. Larsen, A. Legay, M. Peddersen, and A. Wasowski. Compositional design methodology with constraint markov chains. Technical report, INRIA/IRISA Rennes, 2009. submitted for publication.

[26] F. Cassez, T. Chatain, and C. Jard. Symbolic unfoldings for networks of timed automata. In *ATVA*, volume 4218 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 2006.

[27] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang. Synchronous and bidirectional component interfaces. In *Proc. 14th Int. Conference on Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 414–427, 2002.

[28] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 147–188. Springer, 2004.

[29] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[30] Strep combest (component-based embedded systems design techniques). http ://www.combest.eu/home/.

[31] A. David, K. Larsen, A. Legay, U. Nyman, and A. Wasowski. Timed i/o automata :a complete specification theory for real-time systems. Technical report, INRIA/IRISA Rennes, 2009. submitted for publication.

[32] L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea. Sociable interfaces. In *Proc. 5th Int. Workshop on Frontiers of Combining Systems (FroCos)*, volume 3717 of *Lecture Notes in Computer Science*, pages 81–105. Springer, 2005.

[33] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proc. 9th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering (FSE)*, pages 109–120. ACM Press, 2001.

[34] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th Int. Conference on Concurrency Theory (CONCUR)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[35] L. de Alfaro, T. A. Henzinger, and M. Stoelinga. Timed interfaces. In *Proc. 2nd Workshop on Embedded Software (EMSOFT)*, volume 2491 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2002.

[36] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *Proc. 8th Int. Conference on Embedded Software (EMSOFT'08)*, pages 79–88. ACM Press, 2008.

[37] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. *Proc. of the IEEE*, 91(1) :127–144, 2003.

[38] M. Emmi, D. Giannakopoulou, and C. S. Pasareanu. Assume-guarantee verification for interface automata. In *FM*, volume 5014 of *Lecture Notes in Computer Science*, pages 116–131. Springer, 2008.

[39] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23 :151–195, 1994.

[40] K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-objective model checking of markov decision processes. In *Proc. 13th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 4424 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2007.

[41] H. Fecher, M. Leucker, and V. Wolf. Don't Know in probabilistic systems. In *SPIN*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.

[42] G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete-event systems. *Discrete Event Dynamic Systems*, 17(2) :181–205, 2007.

[43] C. Fournet, C. A. R. Hoare, S. K. Rajamani, and J. Rehof. Stuck-free conformance. In *Proc. 16th Int. Conference on Computer Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2004.

[44] B. Gebremichael and F. W. Vaandrager. Specifying urgency in timed i/o automata. In *SEFM*, pages 64–74, 2005.

[45] T. A. Henzinger and J. Sifakis. The embedded systems design challenge. In *Proc. 14th Int. Symposium on Formal Methods (FM)*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[46] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. In *Proc. 2sd Int. Convference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 3253 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2004.

[47] K. G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 1989.

[48] K. G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *Proc. 16th European Symposium on Programming Languages and Systems (ESOP'07)*, volume 4421 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2007.

[49] K. G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proc. 5th Annual IEEE Symposium on Logic in Computer Science, LICS*, pages 108–117. IEEE Computer Society Press, 1990.

[50] The Liège Automata-based Symbolic Handler (LASH). Available at `http ://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[51] A. Legay. T(o)rmc : A tool for (omega-)regular model checking. In *Proc. 20th Int. Conference on Computer Aided Verification (CAV)*, volume 5123 of *Lecture Notes in Computer Science*, pages 548–551. Springer, 2008.

[52] A. Legay, A. Murawski, J. Ouaknine, and J. Worrell. On automated verification of probabilistic programs. In *Proc. 14th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 4963 of *LNCS*, pages 173–187. Springer, 2008.

[53] D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez. Romeo : A parametric model-checker for petri nets with stopwatches. In S. Kowalewski and A. Philippou, editors, *Proc. 15th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 5505 of *Lecture Notes in Computer Science*, pages 54–57, York, United Kingdom, Mar. 2009. Springer.

[54] N. Lynch and M. R. Tuttle. An introduction to Input/Output automata. *CWI-quarterly*, 2(3), 1989.

[55] K. L. McMillan. Using unfolding to avoid the state space explosion problem in the verification of asynchronous circuits. In *Proceedings of CAV*, volume 663 of *Lecture Notes in Comupter Science*, pages 164–177. Springer, 1992.

[56] U. Nyman. *Modal Transition Systems as the Basis for Interface Theories and Product Lines*. PhD thesis, Aalborg University, Department of Computer Science, September 2008.

[57] A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57, 1977.

[58] J.-B. Raclet. *Quotient de spécifications pour la réutilisation de composants.* PhD thesis, Université de Rennes I, december 2007. (In French).

[59] J.-B. Raclet. Residual for component specifications. In *Proc. 4th Int. Workshop on Formal Aspects of Component Software (FACS)*, volume 215 of *Electronic Notes Theoretical Computer Science*, pages 93–110, 2008.

[60] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. Modal interfaces : Unifying interface automata and modal cifications. In *Proc. 9th Int. Conference on Embedded Software (EMSOFT'09)*, pages 87–96. ACM, 2009.

[61] Speeds. http ://www.speeds.eu.com.

[62] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee. On relational interfaces. In *Proc. 9th Int. Conference on Embedded Software (EMSOFT'09)*, pages 67–76. ACM, 2009.

[63] The UPPAAL tool. Available at `http ://www.uppaal.com/`.

[64] P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. Conference on Computer Aided Verification (CAV)*, volume 1427 of *Lecture Notes in Computer Science*, pages 88–97. Springer-Verlag, 1998.

[65] N. Yevtushenko, T. Villa, R. K. Brayton, A. Petrenko, and A. L. Sangiovanni-Vincentelli. Sequential synthesis by language equation solving. Technical Report UCB/ERL M03/9, EECS Department, University of California, Berkeley, 2003.

[66] Y. Zhou. *Interface Theories for Causality Analysis in Actor Networks.* PhD thesis, University of California, Berkley, 2007.