Aspect-Oriented Modelling in UPPAAL for Compositional Verification and Testing

Dragos Truscan\textsuperscript{1}, Jüri Vain\textsuperscript{2}, Junaid Iqbal\textsuperscript{1}, Leonidas Tsiopoulos\textsuperscript{2}, and Ivan Porres\textsuperscript{1}

\textsuperscript{1} Åbo Akademi University, Turku, Finland \{firstname.lastname\}@abo.fi, \textsuperscript{2} Tallinn University of Technology, Tallinn, Estonia, \{firstname.lastname\}@ttu.ee

Abstract. We present a technique of aspect-oriented modelling, verification and model-based testing using UPPAAL timed automata. The approach enables compositional testing and verification, and consequently it reduces the search space needed for these activities. In order to apply compositionality we verify the non-interference of aspects via assume-guarantee assertions. We demonstrate the usability of our approach with a crisis management system example, in which we model aspects for resource authentication and mission execution. Via the example we provide conditions of aspect non-interference that are necessary and sufficient for compositional verification and testing. Finally, we demonstrate by complexity analysis and experimental data how our approach is more efficient compared to the non-aspect-oriented and non-compositional model-based verification and testing.

1 Introduction

Model-based testing (MBT) \cite{28} has become one of the black-box testing solutions for reducing software testing effort \cite{10}. MBT suggests the use of abstract behavioural models for specifying the expected behaviour of the system under test (SUT) and for automatically deriving tests from it. According to Utting et al. \cite{29} there are three distinct phases in MBT: test specification, test generation, and test execution.

Compared to traditional testing methods, in MBT, the testing effort is shifted from mere test purpose specifications to modelling the requirements which the SUT should conform to. However, a recent survey by Binder \cite[p.26]{10} showed that two of the main challenges of MBT are in updating the models and in handling their complexity. In addition, the models used in MBT are not always intuitive and usually only part of the system behaviour is modelled to reduce the test generation effort.

Aspect-Oriented Software Development (AOSD) is a paradigm, based on aspect-oriented programming (AOP) \cite{14}, that addresses the effects of crosscutting concerns on software artefacts: scattering – specifications related to one concern are distributed over several units, and tangling – a given unit contains specifications related to several concerns. The main principle of AOSD is to develop multiple concerns of a software system in isolation (via aspects) and later on to combine (weave) them into a complete working system. The perceived benefits of AOSD are: separation of concerns (which improves the developers’ comprehension of large systems), ease of maintenance, evolution and customization, and thus, greater flexibility in development \cite{26}. A survey of industrial projects \cite{22} reports that AOSD’s main benefits are substantial reduction in model size and improved design stability.
Aspect-Oriented Modelling (AOM) [12] combines the ideas behind AOSD with those of model-based software development, where the main focus is placed on how different concerns of the system can be modelled independently and combined later on via composition mechanisms. Experiments confirm that using AOM techniques provides models of better quality [1] and improved readability [4].

Model-checking is one of the methods used for test generation [29]. Given a formal model of the SUT and a set of properties the model should satisfy, a model checker will generate traces (counter examples) via the verification of these properties. The resulting traces can be used later for creating tests. UPPAAL timed automata (UTA) are one of the most popular frameworks for model-checking and testing, due to their capability to specify timing properties of the SUT and to the availability of mature verification tools such as UPPAAL model-checker [8]. However, model checking tools may suffer from scalability problems due to the so called state space explosion problem. Therefore, in this paper we aim at the reduction of the search space by applying model-checking to partial specifications in the form of aspect models.

In order to address the aforementioned MBT challenges, in this paper we present an aspect-oriented modelling method for model-based testing and verification, in the semantic framework of UTA. Through this work, we attempt to answer two research questions (RQ): RQ1 - how can aspect-orientation help in reducing the updating effort of UTA specifications? and RQ2 - how can aspect-orientation help in reducing the verification and testing effort in UPPAAL, and consequently, increase their scalability?

The contributions of this paper are built on our previous work on combining aspect-orientation and UTA-based modelling, presented in [27], where we defined weaving mechanisms for UTA models and associated tool support for aspect weaving. The scalability of aspect-oriented verification and testing depends on whether these activities can be done compositionally, that is, if it is possible to infer the properties, e.g. conformance relation of the composition from the proven/tested properties of the components/aspects in separation. As a first contribution, starting in Section 4, we extend our earlier results by defining and verifying aspect non-interference in UTA via assume-guarantee assertions. Our work builds on two theories: a) UTA, model checking, and model-based testing [9], and b) aspect non-interference and weakly-invasive aspect theory by Katz and Katz [18]. However, we extend the latter to the timing semantics of UTA.

As a second contribution, we show how compositionality of verification and testing can be enabled and exploited for weakly-invasive aspects in the context of UTA. In this work, we assume that the weaving operators are applied to the class of weakly-invasive aspects and that the weaving is conservative with respect to this class. Weakly-invasive aspects may change the control flow and the values of non-local variables, as long as the state after returning the execution to the base model was reachable in the base model without the aspect woven. In order to illustrate our approach, we apply it on a selected part of the Crisis Management System (CMS) case study suggested as reference case study in [16].

2 Related Work

From the modelling perspective, the Unified Modelling Language (UML) [20] has been the de facto modelling language in AOM and several profiles have been proposed for modelling aspects (e.g., [3,13]). In addition, studies [7,19] provide surveys and assessments of aspect-oriented modelling techniques. In contrast to UML generality, we target semantic unambiguity and mature tool support provided by UTA, while introducing
the AOM constructs conservatively. Although UTA is less expressive than UML, it is better suited for timed model checking and test generation.

An earlier attempt of implementing AOM concepts in UTA has been proposed in [25]. This work was handling the aspect models as refinements of locations and edges in the base model, while in the present work we use aspects for both extending the functionality of the system with new features and for refining the specification. In addition, we verify the non-interference of aspects via assume-guarantee assertions as a prerequisite for compositional verification and test case generation.

Aspect interference is a well-known issue in AOSD, resulting from weaving several conflicting aspects on the same base model. This issue has been discussed by several researchers in [21, 34], while a detailed analysis has been presented in [24]. In order to address this issue we build on the work by Katz and Katz [18], which suggests non-interference criteria for weakly-invasive aspects, that is, aspects that may change the control flow and the values of non-local variables, as long as the state after returning the execution in the base model is reachable in the original base model.

Our approach of verifying non-interference of aspects differs in two respects: (i) instead of modelling of AspectJ in extended state machines, we use the more expressive formalism UTA that allows explicit modelling of timing related aspects in addition to functional ones; (ii) while Katz and Katz use a combination of propositional and linear temporal logic (LTL) for expressing non-interference conditions, we presume that the aspect specifications are expressed in Timed Computation Tree Logic (TCTL) [5], which allows one to express also the non-interference of explicit timing properties. We use the UPPAAL model checker to verify whether the aspects are interference-free, and to decide on the suitability of verifying/testing them compositionally.

From test generation perspective there is a fair amount of work on using abstract models for test generation targeted at aspect-oriented programs. In some cases, state models are extracted from aspect-oriented code, e.g. [6]. In other cases, they are built from the requirements of the system using AOM techniques. For instance, D. Xu generated tests from protocol state machines [32] and use case diagrams [30]. In [30], aspectual use cases help generating test requirements, by first transforming use case diagrams into aspect-oriented Petri nets [31], then extracting the corresponding use case sequences using transition, state, and use case coverage. The latter approach is similar to ours but is limited to use case models. It does not consider time and does not benefit from tool support, neither for the weaving nor for the test generation.

W. Xu and D. Xu suggested in [33] an aspect-oriented extension for UML models in order to generate tests for aspect-oriented programs. Similarly, Ali et al. have suggested a UML profile for modelling robustness behaviour via aspect state machines [2]. In their work, aspects are woven via model transformations, and the resulting model is used for test generation via dedicated tools. Our approach is similar, in the sense that independently developed aspect models are woven via composition, and the resulting model is used for test generation. However, we are targeting timed specifications using the UTA formalism and TCTL-based verification instead of UML.

3 Preliminaries

3.1 UPPAAL Timed Automata

UPPAAL Timed Automata (UTA) [9] used for the specification are defined as a closed network of extended timed automata that are called processes. The processes are combined into a single system by synchronous parallel composition like that in process algebra CCS. The nodes of the automata graph are called locations and directed vertices
between locations are called edges. The state of an automaton consists of its current location and assignments to all variables, including clocks. Synchronous communication between processes is done by synchronisation links called channels. A channel relates a pair of transitions in parallel processes where synchronised edges are labelled with symbols for input and output actions (denoted ch? and, respectively, ch!).

Asynchronous communication between processes is modelled using global variables accessible in all processes. Let $\Sigma$ denote a finite alphabet of actions $a, b, \ldots$ and $C$ a finite set of real-valued variables $x, y, z$, denoting clocks. A guard is a conjunctive formula of atomic constraints of the form $x \sim n$ for $c \in C$, $\sim \in \{\geq, \leq, =, >, <\}$ and $n \in \mathbb{N}^+$. We use $G(C)$ to denote the set of clock guards. A timed automaton $A$ is a tuple $\langle N, l_0, E, Inv \rangle$ where $N$ is a finite set of locations, $l_0 \in N$ is the initial location, $E \in N \times G(C) \times \Sigma \times 2^C \times N$ is the set of edges and $Inv : N \rightarrow I(C)$ assigns invariants to locations (here we restrict to constraints in the form: $x \leq n$ or $x < n, n \in \mathbb{N}^+$).

Without the loss of generality we assume that guard conditions are in conjunctive form with conjuncts including besides clock constraints also constraints on integer variables. Similarly to clock conditions, the propositions on integer variables $k$ are of the form $k \sim n$ for $n \in \mathbb{Z}$, and $\sim \in \{\leq, \geq, =, >, <\}$. For formal definition of UTA complete semantics we refer to [9].

3.2 Aspect-Oriented Modelling

AOM has its roots in the AOP, where an aspect is the pivotal concept, a code implementation of a crosscutting concern. Such an aspect can be inserted in the main (base) functionality of a program at different locations called join points where specific conditions specified by pointcuts are satisfied. An aspect is composed of pointcuts specified in the base model and of an advice specifying the crosscutting concern. The exact mapping between UTA and aspect-oriented concepts will be discussed in the rest of this section.

Aspect-Oriented Modelling in UTA The approach discussed in this paper relies on and extends our previous work on introducing aspect-oriented modelling concepts in UTA [27]. Following aspect-oriented principles, different crosscutting concerns are modelled independently starting from requirements, either as a base or advice models. They are then woven into a woven model.

Before we elaborate on AOT, we interpret the AO terminology in the context of UTA, as follows: A Base Model is a set of UTA processes modelling the base (primary) functionality of the system; an Advice Model is an UTA process or a set of parallel processes implementing a crosscutting concern, specifying what should be executed at a given join point; Join points are model fragments in the base model to where an advice model can be woven (for this paper, we limit our join points to model fragments composed of an edge with synchronization, however the approach can be easily generalized to larger model fragments); a Join point model is the set of join points of a base model where advices can be introduced; a Pointcut is the set of join points and conditions under which an advice can be woven, Pointcut expression - is a logic condition which uniquely defines the model fragments (join points) where the weaving is applied; Woven Model, sometimes referred in the literature as augmented model, is an UTA in which the base model is woven with all intended aspect models; Weaving is the process of composing a base model with the aspects specified by the join point model; Weaving adapters are model fragments that allow the execution of
an advice model at the designated join points. A weaving adapter encodes the pointcut expression, the advice type and the join point.

Our approach is based on several assumptions: • We allow unique instances of the same advice model (defined by a UTA template) to be shared by several join points of a base model; • The execution of an advice is atomic w.r.t. its join point. This means that once a join point is reached, the control flow of the base model process containing the join point will be passed to the aspect model and the base model process will wait for the aspect to complete and return to the same join point. However, this does not restrict several join points located in different processes of the base model to be enabled at the same time and thus their corresponding aspects to be executing simultaneously; • An advice model has one entry point and one or several exit points which return to the same join point, and • The base model and advice model can be woven using UTA-specific communication and synchronisation constructs, e.g. synchronising the entry and exit of the advice model with wait in the base model, sharing or refining data between base and advice model, etc.

Weaving Adapters In [27], we defined four types of weaving adapters. They provide support for weaving an advice before, after, and around a join point, similarly to the homonym advice types in AspectJ. The forth adapter type, conditional, has been suggested based on practical considerations, as it will be discussed later on. Due to space reasons, in this paper we will only discuss the adapters used in our example, while we defer the others to [27].

In the current approach, we restrict a join point to a model fragment composed of an UTA edge labeled with a guard expression, channel, and update as depicted in Fig. 1. The channel represents a synchronisation of send and receive actions denoted respectively as (channel!) and (channel?). Optionally, the edge may carry a guard expression and an update expression. However, our approach can be easily extended to more complex model fragments, as long as our assumptions are fulfilled.

The main purpose of the weaving adapters is to allow a systematic and mechanised weaving of advice models at designated join points of the base model. A weaving adapter has a base model side (BMS) and an advice model side (AMS), specifying the model fragment to be included in the base model and, respectively, to the advice model, during weaving.

The generic conditional adapter (Fig. 2-left) allows the weaving of an advice to a join point before or after a model fragment. It allows one to pass the control to the advice and return to a previously reachable location in the base model via one or several exit points. For instance, the conditional adapter will enable the base model to consume the channel? synchronisation, but the advice will decide if the base model should be executed again via exitAdviceRepeat or if the base model should proceed to the next location via exitAdviceContinue. The corresponding generic advice model is shown in Fig. 2-right. As one may notice, this model can return to the join point via two different channels, corresponding to those in Fig 2. If needed, the adapter may be extended with more complex behaviour, for instance with multiple exit points, which we defer for future publications.

Weaving process In our approach, aspects are woven incrementally, namely, for a given base model and a set of advices, we weave one advice at a time to all of its designated join points. In the following step, the next advice is then woven at its own designated join points. We regard the weaving process as a model transformation,
that takes as input a base model, an advice model, and a selected weaving adapter. The pointcut expression is used as model pattern which identifies join points. The transformation inserts the adapter at the join point and it instantiates the UPPAAL template of the advice for each join point.

4 AO Compositional Verification and Testing in UTA

The efficiency of aspect-oriented verification and testing, depends on whether these activities can be done compositionally, i.e., if it is possible to infer from verified properties or passed tests of components in separation the properties and test verdicts of the composition as a whole. In order to enable a compositional approach, we need first to be able to construct UTA specification in a modular way, via AOM. And secondly, we need to ensure the non-interference between components, i.e., in case of AOM it corresponds to the non-interference verification between the aspects. In the following, we detail how aspect non-interference can be introduced and verified in UTA via assume-guarantee assertions.

Assume-guarantee specifications of aspects The first contribution of our paper stands in defining rigorous semantics for enabling compositional verification and testing by adopting aspect non-interference in the context of UTA and TCTL.

The correctness criteria of aspect models are specified in the form of assume-guarantee assertions of aspects. Let that a system $S$ is composed of a base model $B$ and a set of aspects $A_1, \ldots, A_m$. The specification of an aspect $A_i$ is then a pair $(P_{A_i}, R_{A_i})$, where $P_{A_i}$ represents the assumption on the underlying system model and $R_{A_i}$ expresses the guarantee of the augmented system after the aspect $A_i$ is woven. The base model with its environment are assumed to satisfy the assumptions of the aspect models and the augmented system must satisfy their guarantees. Thus, for an aspect $A_i$, $R_{A_i}$ is the conjunction of formulas of the form: $A \square (pointcut_{A_i} \Rightarrow \varphi)$ stating that whenever in some state of the base model $B$ of $A_i$ the pointcut expression is satisfied, it implies satisfaction of $\varphi$, where $\varphi$ is a temporal logic formula expressing what properties are guaranteed when $A_i$ is executed. In particular, guarantees of the form: $\varphi \equiv A \diamond \psi_{ret}$, are expressing what is expected eventually of each execution of aspect $A_i$ by its exit (the TCTL modalities $\diamond$ and $\square$ express the universal and existential quantifications, respectively, over the set of states of a path). Since the TCTL model checker of UPPAAL does not allow nesting of temporal operators, we transform the temporal subformuli of the shape $R_{A_i} \equiv A \square (pointcut_{A_i} \Rightarrow A \diamond \psi_{ret})$ to time bounded leads to formul\i by using the operator ", where $d$ is the upper time bound of reaching the state where $\psi_{ret}$ holds. $\psi_{ret}$ denotes propositional state property defined on location constants $l \in L(M^{base})$ and state variables $V$ (including clocks $c \in C$).
Verification of Non-Interference

We adopt the work on non-interference of weakly-invasive aspects [18] and we rephrase it for the context of UTA and TCTL:

**Definition 1.** A weakly-invasive aspect is an aspect that may change the control flow and the values of shared variables of the base model at the given join point as long as the execution of the system continues from that join point to any state reachable in the base model without the woven aspect.

We also adopt the definition of the non-interference of sequential-weaving given in [18] in a restricted form by considering timing interference of clocks in different aspects.

**Definition 2.** Given a set $A = \{A_1, \ldots, A_n\}$ of aspects and the assume-guarantee pairs $(P_{A_i}, R_{A_i})$ being the specifications of respectively aspect $A_i$, $A$ is said to be interference-free (denoted $IF$) if and only if whenever the assumptions $P_{A_1}, \ldots, P_{A_n}$ hold in a model of a base system $B$, the augmented system resulting after weaving the aspects satisfies the guarantees $R_{A_1}, \ldots, R_{A_n}$.

Let us consider two aspects $A_i$ and $A_j$, that should be woven sequentially to a base system $B$, in the given order ($A_i$ is woven before $A_j$). And let $\oplus$ denote the sequential-weaving operator of two aspects. Then, the verification conditions that aspects weaving must satisfy in order to guarantee non-interference can be summarised as follows:

1. Each of the aspect $A_i$ and $A_j$ is correct by itself, for instance in the case of $A_i$:
   $$IF^0(A_i) : \forall B \models P_{A_i} \Rightarrow B \oplus A_i \models R_{A_i}$$  
   (1)

Condition (1) expresses that given the assumption holds for base model, the system obtained from weaving the aspect must satisfy the guarantee of that aspect.

2. The rules to verify the non-interference of two aspects $A_i$ and $A_j$, when $A_j$ is woven after aspect $A_i$ are:
   $$IF^P(A_i, A_j) : \forall B \models P_{A_i} \land P_{A_j} \Rightarrow B \oplus A_i \models P_{A_j}$$  
   (2)

The above rule expresses that when weaving $A_i$ to a system where the assumption of another aspect $A_j$ holds, the assumption of $A_j$ should be preserved.

$$IF^R(A_i, A_j) : \forall B \models R_{A_i} \land P_{A_j} \Rightarrow B \oplus A_j \models R_{A_i}$$  
   (3)

This rule expresses that when an aspect $A_i$ has already been woven, weaving another aspect $A_j$ preserves the guarantee of $A_i$. Symmetrically, corresponding IF rules for $(A_j, A_i)$ need to be satisfied. When constructing a model of $n$ aspects, for compositional testing of that model one has to run $n$ verification tasks for the $IF^0$ rules and $n^2$ tasks for each $IF^P$ and $IF^R$ rule.

**Compositionality modulo non-interference** In practice, it is not always possible to achieve interference freedom between all aspects involved in testing. In this case, the compositional approach can be applied in the limited form by grouping aspects by the results of non-interference analysis.

Assuming an augmented system $S^A$ comprises a base model $B$ and a set of aspects $A_1, \ldots, A_m$ which have been IF-verified and $\sim IF(A)$ denotes the set of pairs of
aspects which do not satisfy conditions (1)-(3) (i.e., that have interference), then the union $\bigcup Tr(\sim IF(A))$ over transitive closures of interfering aspect pairs constitutes the finest interference free partition $A|\sim IF$ on $A$. Note that the $\sim IF$-relation itself is not necessarily transitive; the transitive closure here is defined just on the set of aspect pairs that constitute the interpretation set of $\sim IF$-predicate.

The practical implication of $\sim IF$ partitioning is that compositional verification and testing are now applicable not only when $IF(A)$, but also when the subsets of $A|\sim IF$ satisfy pairwise the $IF$-condition, i.e. when $IF(A|\sim IF)$. We call this restricted form of compositionality, compositionality modulo non-interference. Thus, except for the extreme case when all aspects are interfering (i.e. all aspects together must be presented in the test or verification model), due to compositionality modulo non-interference, the aspect-oriented models share the advantages of compositional verification and testing that is relevant when verifying and testing large system builds.

Complexity of Model Checking and MBT The worst-case time complexity of model checking TCTL formula $\varphi$ over timed automaton $TA$, with the clock constraints of $\varphi$ and of $TA$ in $\Psi$ is, according to [17]:

$$O(|\varphi| \times (n! \times 2^n \times \Pi_{x \in \varphi} c_x \times |L|^2)), \quad (4)$$

where $n$ is the number of clock regions, $\Psi$- set of clock constraints, $c_x$ maximum clock constant, $x$ - number of formula clocks. $L : Loc \rightarrow 2^{AP}$ is a labelling function for the symbolic states. In UTA, $L$ denotes the product of data constraints over all locations and edges defined in the model and $AP$ is the set of atomic propositions used in guard conditions and invariants.

Model checking TCTL is (i) linear in the length of the formula $\varphi$, (ii) exponential in the number of clocks in the model and $\varphi$, (iii) exponential in the maximal constants with which clocks are compared in the model and $\varphi$. However, using state space reduction techniques the worst case time complexity can be reduced to being quadratic in the number of symbolic states on data variables in the model [17]. With respect to space complexity, the lower bound for the complexity of model checking TCTL for a Timed Automata model is known to be PSPACE-hard [5].

In practice, time and space complexity of model-checking TCTL on UTA, boils down to the size of the symbolic state space and more specifically, to the number of symbolic states (including clock zones) to be explored and, respectively, stored during the verification. Since by definition the number of locations and edges, as well the number of variables of an aspect model is not greater than that of a behaviourally equivalent non-aspect model, we can conclude from the complexity formula above that every small reduction of model elements provides exponential decrease of number of steps of model exploration, including that of checking the aspect related properties that need to be satisfied on the test models. Moreover, the weaving operators are not introducing additional interleavings of the base model and advice states, due to synchronization built into them.

With our proposed approach we aim at reducing the verification and MBT effort by splitting these activities into smaller ones, requiring less number of symbolic states to store or process. Let us consider the non-aspect-oriented ($S^{NA}$) and, respectively, aspect-oriented ($S^A$) specification of a system. Recall that $S^A$ is modelled as a base model $B$ with which a set of models of non-interfering aspect groups $A|\sim IF = \{A_1, \ldots, A_m\}$ is woven. Then we yield the following claims:

Claim – Compositional verification effort: For any property $\varphi_i$ of any $A_j \in (A|\sim IF)$, its model checking effort $E$ (in terms of time or space) is equal or less than the effort of model checking the property on the non-aspect oriented model $S^{NA}$:
\[ E(B \oplus A_j \models \varphi_i) \leq E(S^{NA} \models \varphi_i), \forall i \in 1, \ldots, m. \] (5)

where any of \( A_1, \ldots, A_m \) groups of aspects is woven with the base model.

Claim – Compositional testing effort: the (time or space) effort \( E \) of generating a test case \( T_{i,j}^{A} \) from an aspect-oriented model \( A_j \in (A|_{\sim I_F}) \) will be less than the effort of generating the same test from the non-aspect oriented model \( S^{NA} \):

\[ \forall i, j : E(T_{i,j}^{A}, A_j) \leq E(T_{i,j}^{NA}, S^{NA}) \] (6)

The validity of formula 5 and 6 stems from the fact that aspect-oriented models represent subsets of behaviour of the non-aspect system model \( S^{NA} \). The performance gained by enabling compositionality via aspect-oriented modelling is demonstrated experimentally in the following section.

5 Case Study - Crisis Management System

We take as example a specification inspired from the Crisis Management System (CMS) proposed as reference case study in [16]. The idea behind CMS is to deal with the different kinds of crisis situations, ranging from major to catastrophic accidents, by allocating resources to handle the crisis. The major actors in the CMS are the Coordinator and the Resources. Coordinator receives calls from witnesses reporting an incident and initiates new missions. Each mission has a type which tells the number of resources that have to be allocated. Three procedures of the CMS will be modelled:

**Resource Allocation.** A mission needs to execute within a specified time Mission Time (MT). Coordinator initiates the mission by sending a message to CMS (cSystem). The latter is responsible for allocating resources for the mission and is waiting for them to complete. When Resources complete their tasks within MT, cSystem informs Coordinator about the successful completion of the mission. If a Resource cannot be allocated, cSystem will try to allocate the next available Resource. If cSystem cannot allocate the required Resources within MT, then the mission will timeout. If not all allocated Resources complete their task within MT, the mission is considered failed and Coordinator is notified.

**Authentication.** Every Resource needs to authenticate itself before it can be allocated to a mission. The authentication process follows a simplified version of Needham-Schroeder protocol adopted from [23]. There are two participants in the protocol: A - the initiator, and B - the responder. In the first step, the initiator A sends a message containing its identity to the responder B and after a sequence of encryption/decryption procedures of the information using public and private keys, the initiator is eventually authenticated. In case of an authentication failure a resource cannot be used for the current mission. The authentication process for every resource requires a fixed amount of time, based on the time needed to generate the secret key of B, and to encrypt and decrypt the messages sent between A and B.

**Execute Critical Mission (ECM).** Each resource has a Time to Arrive (TTA) to the mission site and a time to handle the mission once arrived (timeAtLocation). Whenever an allocation request is received by a resource, it will calculate its feasibility. If it cannot reach the site on time, the resource will decline the mission and notify cSystem.
Fig. 3. Flat model with one resource including the Authentication and ECM features

Modeling the CMS For evaluation purposes, we developed two versions of the CMS specification (in the following referred as flat and, respectively, aspect-oriented model), one using traditional, non-aspect-oriented methods and one using the aspect-oriented methods proposed in this paper. To ensure that the two versions are specifying the same behaviour, we use bisimulation as detailed later on.

Fig. 3 depicts the flat version of CMS specified using the traditional modelling approach, containing a Coordinator, a cSystem and a resource (flatResource1) process (only Coordinator and one Resource are shown in order to save space, while cSystem is shown in Fig. 4). cSystem initially allocates the number of required resources. After getting an alloc synchronization from cSystem, resources are required to authenticate in order to receive mission critical information. After authentication, resource assesses its feasibility for the mission and proceeds to Travel. In case of infeasibility or failed authentication, the resource moves to a failed state, after which it can not be used for the mission. If a resource is not feasible for the mission, cSystem will try to allocate another resource. In case of not having enough resources, the mission will fail and cSystem informs Coordinator about failure. Moreover, the successful required resources (i.e that reached the location and performed their task within MT) contribute to overall success and mission is completed when all required resources have successfully reached the Done location in the model. For the aspect-oriented version of the CMS, the starting point is modelling the resource allocation procedure which will be considered as the base model (see Fig. 4). Briefly, CMS receives requests for new missions and starts allocating resources according to the type of mission. When all the resources have completed the mission (allDone()), it will notify the Coordinator that the mission is over (missionDone!). Resource1 awaits to be allocated to a mission and then it notifies when its task is done. A system can and should have more than one resource, as minimum number of resources required for the selected mission type.

The additional procedures of CMS are modelled as two advices, which are incrementally woven into the base model (see Fig. 5, where the cSystem automaton is omitted since it is identical to the one in Fig. 4). The ECM advice refines the time behaviour of how a resource executes a mission. Beside the number of required resources, a critical mission requires that the resource arrives at the location within MT. Thus, we characterize each resource availability by its distance to the incident site, specified as a TTA (time to arrive) clock constant. A resource can be allocated to a mission only if its TTA
is less or equal to MT. This exemplifies one possible option of weaving advice models, as a refinement of an abstract model. The ECM advice is woven in the base model at the alloc[i]? join point using the conditional adapter as described in Section 3.2.

The second advice to be modelled is the Authentication (Auth) of resources. We model the authentication aspect as a separate template. Whenever an internal resource is initialized, it attempts to authenticate itself via eAuth[i] by providing its credentials. The authentication advice is modelled as two parallel automata, one for the initiator and one for the responder. The initiator automaton will receive the credentials of a given resource from the resource process in the base model, while the Responder process will provide the verdict of authentication back to the base model via the two channels of the adapter exAuthT! for successful and exAuthF! for unsuccessful authentication.
Verifying equivalence between flat and aspect models In order to ensure that the flat and the aspect model (composed of $Base + Auth + ECM$) have exactly the same observable behaviour, we employ bisimulation. Intuitively, two UTA are bisimilar if they accept the same timed language, i.e., they perform exactly the same observable action transitions and if they reach bisimilar states. In other words, each of the systems cannot be distinguished from the other by an observer. Bisimilarity is symmetrical. Bisimulation for timed automata has been originally introduced by [9] and as shown in [11] it is decidable for parallel timed processes.

In order to verify bisimilarity, we follow these steps: we compose a new UTA model containing both a flat and an aspect model; then we decide the observable interface between the two models (i.e., the action transitions to be observed); we add additional, side-effect free, synchronization channels between the models for the observable actions; and verify that the complete model never deadlocks on all possible paths via using TCTL formula: $A \parallel \neg \text{deadlock}$. Due to the large size of the models we defer bisimulation details to [15] (available online).

Verification of Aspect Non-Interference In order to verify aspect non-interference we instantiate the IF rules specified earlier. We then formulate them as verification queries in TCTL. For each aspect we specify its assumptions and guarantees as follows:

$P_{Auth}$ - correct public keys for the initiator and responder are provided
$R_{Auth}$ - authentication will take place in max $4 \times nh\_timeout$
$P_{EMC}$ - a resource is in reachable distance (timing-wise) from accident site
$R_{EMC}$ - resource will complete its task in $MT$

Assuming that $IF^0$ rules have been verified for each aspect, let us exemplify this process by instantiating the $IF^p$ for the case when weaving $ECM$ advice after $Auth$ aspect has already been woven: $IF^p(Auth, ECM) : \forall B(B |= P_{Auth} \wedge P_{EMC} \Rightarrow B + Auth \models P_{EMC})$; where in order to verify $P_{EMC}$ we use the following query: $aspectResource1\_allocate \Rightarrow c <= MT - (TTA\_0 + timeAtLocation\_0)$; where $aspectResource1\_allocate$ means that the resource has been authenticated in $4 \times nh\_timeout$. All the other necessary IF rules for this case study are verified in a similar manner.

The verification of non-interference, showed that $Auth$ and $ECM$ advices are non-interfering, thus allowing to weave them incrementally in any order into the base model.

6 Evaluation In order to validate our approach and provide an answer to the two research questions formulated in the introduction, we work with the following hypotheses: (H1) aspect-oriented UTA models are easier to update compared to models built using the traditional method and (H2) the compositional verification and test generation effort of UTA aspect models it is less than the verification and testing effort of the augmented model.

Update effort evaluation During our case study development we have also conducted an experiment on the benefits of using aspect models versus traditional modelling approach. For that purpose we developed two versions of the authentication feature. The first version used a simpler version, in which the resource provided its credentials (authentication token) and the CMS checked if they correspond to a list
of known credentials. The CMS specification was developed both as flat and aspect models and the two models where checked for equivalence via bisimulation. In the second version, we updated the authentication feature to use the Needham-Schroeder protocol as described in Section 5 which resulted in the models described in this paper. During the update we measured the number of changes (statement additions, updates, removals) that we had to perform on the flat model and we obtained that modifying flat models required changing approximately 40% of the models, versus 20% in the case of aspect models.

We noticed that the effort of updating the models was much smaller in the case of aspect models, since it was easier to identify which elements had to be changed, and these elements had in general a local scope, without affecting the specification of the other features. This was an expected result, according to different studies, which confirm that AOM reduces the scattering and tangling of requirements. This also confirms our Hypothesis 1.

Evaluation of Compositional Verification and Testing As stated in the introduction, the verification and testing in UPPAAL are of the same complexity class since verification traces are later used as test sequences. In order to compare the verification and testing effort, and exemplify the difference between the traditional and aspect-oriented approach, we use the verification query below and benchmark it against the flat model, Base + Auth model, and Base + ECM model. Since the flat and Base + Auth + ECM have been proven bisimilar, a property that will pass the verification on the former, is expected to also pass the verification on the latter. For instance, we use the following property “The system should not be in a deadlock state except when the mission is completed, there was a timeout or the mission has failed” which corresponds to: A[] deadlock imply(cSystem.Done || cSystem.TO || cSystem.MissionFailure).

For each version of the models we record the number of stored symbolic states and number of explored symbolic states. These will give an indication on the effort (in terms of space and time) for verifying the query.

The results show that the verification effort for the above query is 871 stored and an 1280 processed symbolic states, respectively. The benefit of compositionality becomes evident when verifying the aspect models independently. The verification search space resulted in 336 stored symbolic states for the Base + Auth model, and in 528 stored symbolic states for the Base + ECM models. Similarly, the number of steps required to verify the query was 635 and, respectively, 535 explored symbolic states.

The individual effort of verifying the query on the aspect models is, as expected, much smaller than verifying the query on the flat model. In this particular case, the sum of the effort of the two verification tasks is also smaller than the verifying the same query on the corresponding flat model, both time and space wise. The results are in line with the verification effort given by formula (4) and thus it confirms our Hypothesis 2.

7 Conclusions and Future Work

In this work, we searched the answers to the two research questions presented in the introduction. When addressing these questions, we have introduced aspect-oriented concepts in the context of UTA specifications with the purpose of creating more modular specifications which facilitate the compositional testing and verification. A set of
generic adapters are used for systematic and tool-supported weaving of UTA-based aspect models without modifying the underlying UTA formalism. In addition, verification rules can be specified in TCTL for ensuring the validity of the weaving process.

Besides improving human comprehension and traceability of requirements the modularity of aspect oriented models provides means to benefit from compositionality of verification and testing. For this purpose, we used assume-guarantees assertions for verifying the non-interference of aspects via model checking. The non-interference analysis facilitates conformance testing by allowing one to construct aspect-oriented models compositionally. That is one prerequisite of scaling up the testing process to the industrial-strength methodology. Another benefit that comes from doing the verification and testing compositionally is that the independent verification task can be run in parallel (using for instance cloud computing technology) and thus drastically reducing the amount of time and computational resources needed compared to the traditional verification approaches.

Our approach has also some limitations stemming from the way we weave aspects via weaving adapters. Even though the weaving adapters are not introducing additional interleavings in the base model, they introduce additional structural elements which increase the complexity of the aspect models and which directly affect the symbolic states space. Thus, the larger the structural complexity of the advice model, the more efficient our approach is when doing compositional testing and verification, since the complexity of the advice model will compensate for the structural complexity introduced by the adapters.

The main potential reason of low efficiency of our approach in certain situations stems from the fact that when the aspects are strongly interfering we loose the advantage gained from the compositional verification of aspect-oriented models. However, our approach is strongly supported by the theoretical background of the complexity of model checking and thus this threat is minimized.

In future work, we plan do define aspect-oriented coverage criteria for UTA, which will allow focusing of the test generation on specific parts of a models. We also plan to conduct a set of larger case studies in which to evaluate the scalability of the approach as well as its advantages from the point of view of incremental test suite updates.

References
