Automated Composition of Timed Services by Planning as Model Checking

Daniel Stöhr, Sabine Glesner

Technische Universität Berlin, Chair Software Engineering for Embedded Systems,
daniel.stoehr@tu-berlin.de, sabine.glesner@tu-berlin.de,
www.pes.tu-berlin.de

Abstract. Techniques of automated service composition can shorten development time by generating a concrete service composition out of a set of abstract composition requirements. However, no existing fully automated approach is able to deal with timed services and timed composition requirements. In this work, we propose an approach for the automated composition of timed services, represented as timed i/o automata, by adapting the AI planning method Planning as Model Checking. Thus, the concept of automated service composition can be used in domains with real-time requirements. As case study, we model a system where medical devices need synchronization during surgery.

Keywords: automated service composition, timed services, real-time, timed i/o automata, planning as model checking

1 Introduction

Designing controller programs coordinating distributed components in a safety- and time-critical environment, e.g., for synchronizing medical devices, is a very complex and time-consuming task. While keeping development time short, the software engineers have to assure that the overall system fulfills functional and safety-critical requirements. These opposites yield the need for automated and scalable tools supporting the development process. In our domain, such tools have to produce correct results and have to deal with nondeterminism and time as part of the service behavior. Speaking in terms of Service-oriented Architectures (SOAs), which are an uprising paradigm in those domains, the problem of designing a controller corresponds to the problem of finding a suitable orchestrator to compose a given set of services.

Therefore, we propose an approach for the automated composition of timed services including real-time properties as composition requirements. As to the authors' knowledge, no existing fully automated approach for service composition is able to deal with those requirements. To realize our approach, we describe the behavior of the services as *timed i/o automata* [VL92] and the orchestrator as an automaton handling the input and output actions of the original automata. Thus, we adapt the AI planning method *planning as model checking* [GT00] to realize the automated composition process, by bringing real-time into the existing theory.

Moreover, as a part of future work, we want to implement a tool realizing the resulting composition algorithms. Such a tool can be used to shorten the development process for systems in our domain, because an initially correct controller model is generated where a hand-made model had to be created before. To discuss the requirements for our approach, we present a case study where medical devices have to be synchronized during surgery.

The rest of this paper is structured as follows. In Section 2 we shortly outline the concepts of automated service composition, timed i/o automata, and planning as model checking. They form the basis for our approach. Afterwards, in Section 3 we present our case study, our proposed approach and what extensions to the existing theory are required. In Section 4 we discuss related work. Finally, in Section 5, we conclude this paper and give an outlook on future work.

2 Background

In this section we introduce works upon which our approach is based. In Section 2.1, we shortly explain the concept of automated service composition and the decision for the underlying theory of our approach. Afterwards, in Section 2.2, we introduce timed i/o automata, used to formally represent the services to be composed. Finally, in Section 2.3 we present *planning as model checking*, which forms the basis for the composition process of our approach.

2.1 Automated Service Composition

In the context of our work, the term automated service composition denotes the process of generating an orchestrator for a set of services out of a set of composition requirements. An orchestrator is a central service within a service composition that communicates with the other services and directs messages between them in order to create the system behavior described through the requirements.

In [BP10] an exhaustive overview is given on automated composition approaches for web services. They present 27 approaches realizing different forms of automated service composition and compare them to each other with respect to a certain set of properties. In the following we outline the properties, which are relevant for our work.

Automation Describes the degree of automation, offered by an approach. We need a high degree of automation. Besides of a formal description of the composition requirements on a set of services, the user intervention shall be reduced to a minimum.

Nondeterminism An action may produce different nondeterministic outcomes. In our domain nondeterminism occures, e.g., as device alarms.

Scalability Our approach shall deal with large and complex sets of services. Hence, we have to design our composition algorithms for high efficiency.

Correctness Compositions are guaranteed to be correct w.r.t. the compostion requirements. Because our approach shall generate compositions for safety-critical domains, we have to assure the correctness of resulting compositions.

Based on the above-mentioned survey, we compared these properties to the different composition approaches. We decided that model checking based methods of automated service composition are most suitable to form the basis of our approach. These approaches are the only ones, fulfilling all these properties at once. However, none of the presented approaches includes real-time composition requirements.

2.2 Timed I/O Automata (TIOA)

To realize our approach, we need a suiting formalization to describe the behavior of the orchestrator and of the services to be composed. For that, we chose the theory of *timed i/o automata* (*TIOA* [VL92]). These are *finite automata*, whose actions are divided into input and output actions, so that we can describe the interface of our services. Moreover, we can express timed behavior over a set of clocks. Therefore, guards exist for transitions, and invariants for states.

In our case study (the synchronization of medical devices), we can model each device as a distinct automaton. Messages between connected devices are represented as input- and output-actions. With guards and invariants, we can express timed conditions on the interaction of our devices, e.g. when a device has to react within a certain timeframe.

An example for the graphical represention of TIOA is given in Section 3.1 where we present our case study.

2.3 Planning as Model Checking

In the following, we outline the AI planning method *planning as model checking* [GT00]. The underlying idea of this technique is to generate *plans* for a given *planning domain* by determining whether formulas are true in a model.

The planning domain is described through a model similar to *finite automata*. The planning problem is described in temporal logics, as a CTL formula, containing desired final states and constraints on the paths allowed in the planning domain. Solving the planning problem for a planning domain means finding paths leading from the initial state to the final states. Here, the problem is solved by lifting it to a model checking problem. The planning problem is expressed as a corresponding *kripke structure* and the plan is generated by checking whether suitable temporal logic formulas are true within it. For this purpose, an iterative algorithm checks paths in the structure against corresponding parts of the formulas.

The algorithm is based on *Binary Decision Diagrams* a data structure that can represent kripke structures as graphs representing boolean formulas (a common technique for solving model checking problems). The plan is iteratively built up as a BDD by comparing it to other BDDs (representing the domain and requirements) and by performing transformations on it. It has been implemented within the *Model Based Planner* [MBP].

For our approach, we will translate the set of TIOA into a planning domain by building up the crossproduct. Thus, we transform the problem of automated composition into a planning problem.

3 Automated Service Composition with Real-Time Requirements

In the first part of this section we present a case study, demonstrating how our approach can be applied. In Section 3.2, we describe the workflow of our approach. In Section 3.3 we outline problems we have to tackle when extending the existing theory, by reffering to our case study.

3.1 Case Study

We have investigated the requirements for our approach by creating a TIOA model of a use case where an x-ray device and an anesthesia machine ventilator need synchronization during surgery. The use case is described in [AGWL09], a work where the technical interoperability between medical devices is investigated. In this scenario, an x-ray image of a patient's chest had to be taken under general anesthesia. When the x-ray is performed, it must be ensured that the patient's lungs are empty in order to receive a clear image. Therefore, parameters of the anesthesia machine's ventilator are accessed by the controller, to trigger the x-ray exactly between two breaths. In Figure 1 we show a simplified version of the system model we created.

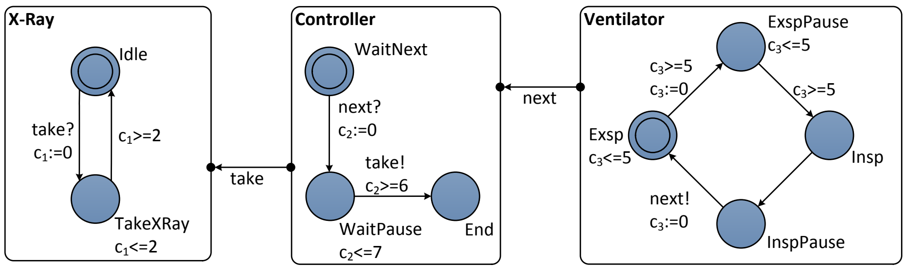

Fig. 1. synchronizing an x-ray and an anesthesia machine ventilator

The left automaton, `X-Ray`, describes the behavior of the x-ray device which, if triggered, needs 2 time units for taking an image. In it's initial state, `Idle`, it awaits the reception of the input signal `take`. If the signal is triggered the automaton changes to the state `TakeXRay` and resets the clock `c1`. The guard of the transition back to `Idle` ensures that the x-ray stays in `TakeXRay` for at least 2 time units and the invariant ensures that the state is left after at most 2 time units.

The right automaton describes the ventilator machine which controls the breathing, i.e., the respiration cycle of the patient. In its' initial state `Exsp` the ventilator lets the patient breath out. The invariant and transition guard ensure, that the exspiration phase lasts exactly 5 time units. When the state `ExspPause` is activated, the respiration pauses for 5 time units. Afterwards, the inspiration phase and pause take place, analoguous to the exspiration phase (here, we ommited the time constraints since they are not relevant for our use case). When `Exsp` is entered again, the signal `next` is emmited. That signal is the only possibility for other devices to synchronize with the ventilator.

Before describing the controller, we outline its behavior via CTL formulas (these will be the composition requirements for our approach). Firstly, we have a functional property (1)**AF** *TakeXRay* which says that the x-ray has to be performed somewhere during the controllers execution. Secondly, theres a safety property (2)**AG** *TakeXRay*→*ExspPause* which describes that whenever the x-ray is exposed the ventilator has to be in the exspiration pause mode.

Based on these requirements we can model the `Controller`. In its initial state `WaitNext` the controller waits for the signal `next` stating that a new respiration cycle begins. In `WaitPause` the controller is ready to trigger the x-ray (requirement 1) and waits for the right point in time to do so (requirement 2). The corresponding transition takes place after at least 6 time units. This is the point where the ventilator has entered `ExspPause` for sure. The invariant ensures that the exposure time of the x-ray does not overlap with the ventilator's inspiration phase.

3.2 The Approach

In this section we present our approach for realizing the automated composition of timed services. Therefore, we adapt the AI Planning method *planning as model checking* (Section 2.3). As language for our service models we have chosen TIOA (Section 2.2). We realize our approach by bringing real-time into the theory and by building a framework to make the theory compatible with Timed i/o Automata (inspired by the work discussed in Section 4). The workflow of our proposed approach is visualized in Figure 2.

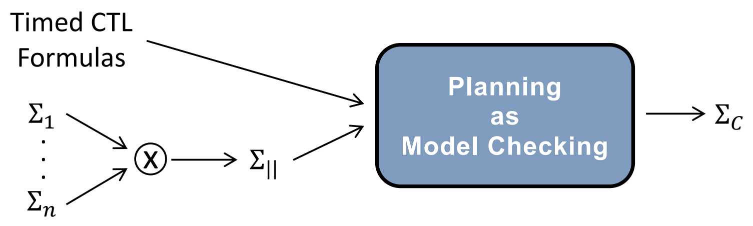

Fig. 2. Workflow of our approach for the automated composition of timed services

Initially, a set of TIOA $\Sigma_1, ..., \Sigma_n$ describes the communicational behavior of our services, and a set of *Timed CTL formulas* [ACD93] describes functional and real-time composition requirements. In a first step, the parallel product $\Sigma_{||} = \Sigma_1|...|\Sigma_n$ is built. In the sense of planning as model checking, $\Sigma_{||}$ leads to the *planning domain* and the formulas to the *planning problem*. In our example the crossproduct of the x-ray and ventilator automata are the planning domain and the CTL formulas (1) and (2) are the planning problem.

Afterwards, the algorithms of planning as model checking are applied to solve the planning problem by identifying all paths fulfilling our requirements. This gives us the *Control Automaton* Σ_C handling the inputs and outputs of the original automata, so that the required overall behavior of the composition is assured. By the means of service composition, Σ_C is an *orchestrator*.

Since the existing theory and implementation of planning as model checking can only deal with untimed domains and 'simple' CTL formulas, we have to make it capable of dealing with timed domains and Timed CTL formulas. We describe the problems, we await for the extensions, in the next section.

3.3 Extending the Existing Theory

In this section we sketch the problems we have to tackle for extending the existing planning theory, described in section 2.3. We identified three situations where timed behavior has to be considered. In the following, we enlist these properties and how they will affect the extensions.

Interaction of existing services Guards and Invariants can produce situations in which deadlocks may occur or where safety properties may be violated. In our example that would happen in the controller state `WaitPause` if the x-ray would be triggered too early or too late, so that condition (2) is violated. In our hand-implemented controller we solved this through additional guards and invariants.
To automate this step, the extended planning theory has to analyze the behavior of the existing services for those situations. We can achieve this by using an extended version of BDDs capable of representing timed automata (as used in the Rabbit Model Checker [Rab]). Here, several BDDs are used to represent the functional and timed behavior of an automaton seperately. Therefore, we need adapt the BDD based operations of the planning algorithm.

Something has to happen in a specific moment This takes place, for instance, if we want the x-ray to be triggered exactly 6 time units after `next` has been received.
Here, we have to modify how the planning algorithms resolve single planning goals because those now depend on time constraints. Moreover, we have to find a way to express those requirements in Timed CTL.

Something has to happen iteratively within a specific interval This situation takes place, e.g., if we want an x-ray image to be taken every 100 time units. Here, too, CTL is not sufficient to express those requirements. For this situation we have to solve problems similiar to the point above.

In the first part of this section we have presented a case study showing that time constraints occur, when medical devices have to be synchronized over a central controller. Since automated service composition can accelerate the development of those controllers and no composition approach is able to deal with time, we have proposed an approach for the automated composition of timed services. In the last part we outlined the problems we have to solve to realize our approach.

4 Related Work

In this section, we present works related to our approach. Firstly, we describe an already existing approach for automated service composition of web services that uses planning as model checking for the composition process (but does not include real-time requirements). Afterwards, we outline works bringing together (automated) service composition and real-time.

In [PTB05] a framework is described that uses planning as model checking to automatically generate a BPEL composition out of a given set of web services and composition requirements. The way how the planning theory was utilized to solve the composition problem served as an inspiration for our proposed approach. In contrast to our work, this tool cannot handle composition requirements expressing real-time properties of the services to be composed. However, real-time capabilites are one of the main characteristics of our proposed approach. Furthermore, this approach was designed for the domain of bussiness processes and does not apply to our domain of controlling distributed devices in a safety-critical environment.

Most (if not all) works that try to bring together *non-automated* service composition and real-time consider time as measurement for communication latency between world-wide distributed services [MGY+10] or telephone servers [LL07]. In these cases, time is a part of the *Quality of Service* and helps choosing a proper service instance during the composition process. These works do not solve our problem because we need time as a part of the service's behavior itself.

As to the authors' knowledge, [KDM+09] is the only work where an apporach for *automated* service composition with real-time requirements is realized. In contrast to our approach, this work offers a very low degree of automation, because the overall workflow of a BPEL composition has to exist before time requirements can be specified. Our approach, on the other hand, offers a very high degree of automation by generating the orchestrator from scratch.

5 Conclusion & Future Work

In this work we have proposed an approach for the automated composition of services with real-time capabilities. The domain for our approach is the generation of controller programs coordinating distributed services in a safety-critical environment out of a set of functional and safety-critical composition requirements. We presented a case study, where medical devices have to be synchronized, and have used *timed i/o automata* as a formal description of the services' communicational behavior. To realize the automated composition process we currently adapt the AI planning method *planning as model checking* and have outlined the extensions we will have to bring in to make it capable of dealing with time.

In future work, we perform a larger case study than the one presented here, where we model devices used during a specific diagnostic method (a PET/CT scanner and an injection pump). We work on that case study in close cooperation with the Charité Berlin.

By using our proposed approach, the development time for the above-mentioned controller programs can be shortened because certain development steps can be performed automatically. Furthermore, the generated controller model is correct with respect to the composition requirements due to the use of model checking in the generation process. Thus, the iterative step of initially designing and refining a controller model by hand can be skipped and development time can be saved.

References

ACD93. Alur, R.; Courcoubetis, C.; Dill, D.: Model-Checking in Dense Real-time. Information and Computation, vol.104, pp. 2-34, 1993.

AGWL09. Arney, D.; Goldman, J. M.; Whitehead, S. F.; Lee, I.: Synchronizing an X-ray and Anesthesia Machine Ventilator: A Medical Device Interoperability Case Study. International Conference on Biomedical Electronics and Devices, pp. 52-60, 2009.

BP10. Baryannis , G.; Plexousakis , D.: Automated Web Service Composition: State of the Art and Research Challenges. Technical Report - Foundation for Research & Technology - Hellas Institute of Computer Science, 2010.

GT00. Giunchiglia, F.; Traverso, P.: Planning as Model Checking. Recent Advances in AI Planning, LNCS vol. 1809/2000, pp.1-20, Springer Berlin/Heidelberg, 2000.

KDM$^+$09. Kallel, S.; Charfi, A.; Dinkelaker, T.; Mezini, M.; Jmaiel, M.: Specifying and Monitoring Temporal Properties in Web Services Compositions. Seventh IEEE European Conference on Web Services, pp.148-157, IEEE press, 2009.

LL07. Lin, L.; Lin, P.; Orchestration in Web Services and Real-Time Communications. Communications Magazine vol.45, no.7, pp.44-50, IEEE press, 2007.

MBP. MBP: a Model Based Planner. http://mbp.fbk.eu/. Last visited: February 2012.

MGY$^+$10. Moussa, H.; Gao, T.; Yen, I.; Bastani, F.; Jeng, J.: Toward effective service composition for real-time SOA-based systems. Service Oriented Computing and Applications Vol.4, pp.17-31, Springer London, 2010.

PTB05. Pistore , M.; Traverso , P.; Bertoli , P.: Automated Composition of Web Services by Planning in Asynchronous Domains. Artificial Intelligence vol. 174, pp.316-361, Elsevier Science Publishers, 2010.

Rab. Rabbit and Cottbus Timed Automata. http://www.sosy-lab.org/ dbeyer/Rabbit/. Last visited: February 2012.

VL92. Vaandrager, F.; Lynch, N.: Action Transducers and Timed Automata. Proceedings CONCUR'92, LNCS vol.630, pp.436-455. Springer-Verlag, 1992.