

Modelling and verification of BPEL business processes

Marina Mongiello

Dipartimento di Elettronica ed Elettrotecnica
Politecnico di Bari, Italy
mongiello@poliba.it

Daniela Castelluccia

Dipartimento di Elettronica ed Elettrotecnica
Politecnico di Bari, Italy
d.castelluccia@virgilio.it

Abstract

A business process is a complex web service with functions provided by different web services, which are already existing in web and are dynamically integrated for granting a more complex business task. For this reason, business processes have become more and more diffuse in B2B and B2C domains, so that the importance of their activities asks for a high-level of reliability. Methods and tools for supporting automatic system verification and validation could be useful. Among the techniques of automatic verification, we choose Model Checking method, because we applied it efficiently for verification of a single web service and in this paper we extend the area of application also in business processes. Descriptions of the behavior of a business process are coded using a standard language, BPEL4WS, that has broadly spread because it is able to describe a business process as both an executable process and an abstract process. Therefore, we model a BPEL description of a generic business process with a formal model and we formalize correctness properties about the reliability of the business process design. Also, we build a framework that performs automatic verification of formal models of business processes through NuSMV model checker. If there is a violation of correctness specifications, NuSMV provides counter-examples, so we can locate errors and effect right changes for correcting business process design.

1. Introduction

Nowadays, the functions of web services have become more and more applied in many domains like Business to Business, Business to Consumer and Enterprise applications. For instance, a company has to make a purchase: it can seek from the web advices of different suppliers or it can check for offering of one supplier for days and it buys when the price goes below a specified threshold. Generally speaking, when different companies want to realize transactions among them using web services, the functions pro-

vided by different web services have to be integrated together for granting a more complex function.

We define business process (BP) a complex web service, that originates from the composition of web services, which are already existing and dynamically integrated in one complex process. This integration is realized through a synchronous exchange of messages or through asynchronous not correlated interactions.

It is necessary to define a formal description of composition of web services with a standard language, so that it is commonly understandable both to human designers and to tools for automatic composition. Standard languages for single web services are used (SOAP, Web Services Description Language and Universal Description, Discovery, and Integration), but they are not suitable enough for complex web services. SOAP defines the protocol of receipt and dispatch of messages in XML format, WSDL describes the functions that can be provided by a web service through a connection to a definite port number, UDDI provides a method to publish a web service so that it can be discovered in automatic way in the web and other web services can access to it. However, even if WSDL defines the public interface of a web service, it doesn't provide informations about its behavior with other web services.

Descriptions of behavior of web services and their interactions can be defined using a standard language of high level like BPEL4WS (Business Process Execution Language for Web Services), simply BPEL, which is able to explain business process from a high level point of view. Particularly speaking, BPEL describes all the features and the services provided by every web service that is a partner in the business process (executable process). BPEL also describes the protocol of message exchange between partners for performing the complex task, without revealing operations that partners internally complete (abstract process). So, BPEL has broadly spread because it is able to model a business process as both an executable process and an abstract process.

However, it is possibile that problems happen during the execution of a business process: for instance, problems of

interaction like invocations of remote services that do not answer, problems in software like deadlocks in remote systems, problems of net-communications like temporary congestions. The designer of a business process has to also verify these situations, therefore a framework that can automatically check for weakness in design could be useful. After automatic verification, the BP design can be revised before the real system implementation is submitted to testing.

Among the techniques of automatic verification, the Model Checking method can be efficiently applied for verification of a single web service, as our previous papers show [8],[9],[11]. In this paper, we extend the area of Model Checking applications also in business processes, which are described through BPEL.

The structure of this article is the following one. The section 2 introduces the notion of composition of web services with particular attention to BPEL4WS description language. This section also describes the state of the art of formal techniques in BP verification. The section 3 explains how a business process, which is described in BPEL, can be turned into our formal model. The section 4 explains which specifications can be verified in our model; specifically, temporal properties can be expressed and verified about messages, that are sent during BP execution. The section 5 describes the architecture of our system prototype for automatic verification of formal model of business processes through Model Checking. Finally, the main purposes and the future improvements of our research are synthesized.

2. Composition of Web Services

The main reason of the great diffusion of web services is the possibility to have a big collection of available software programs, that are already implemented and existing in Internet. They are accessible through standard protocols as SOAP, WSDL and UDDI. Through such standards, the services provided by a web application can automatically be discovered and integrated in further web services or they can be coordinated to build more complex services.

At a basic level, a web service is a collection of: "activities", whose execution has to perform tasks of concern, and "messages", that allow the web service to contribute in the activities of a business process. Activities are similar to traditional programs; messages are interactions among different web services, however web services keep own autonomy.

Different languages have been adopted to describe interactions of business processes: BPEL [3], WSCI [12], BPML [1], DAML-S [2]. The development of web services, that is based on these standards, is supported by different implementation platform as .Net and J2EE. Automatic composition of web services has been topic of concern for re-

searchers in the last years. For instance, in [7] a mapping from BPML into OWL-S is realized: BPML is a graph-based description language for composition of web service that is especially suitable to model requirements of enterprise applications; OWL-S is a XML-based ontology that allows identification, composition, execution, monitoring of web services. However, the mapping is partial because of the limited expressiveness of OWL-S (OWL-S is not able to describe information sharing among web services). In [10] a Petri net-based framework is explained for verification of business processes, which are described with another standard language, DAML-S.

However, BPEL is the most employed standard de-facto, because it is able not only to describe the services provided by every web services of a business process, but also it describes the global behavior of the business process, that is based on a synchronous and asynchronous exchange of messages. Specifically, a business process described in BPEL incorporates different functions provided by web service partners, exclusively using the respective public interfaces and hiding their inside implementations. This separation between public and private aspects involves two advantages: companies that provide web services do not reveal their methods of internal data management or decision making; besides, in case of change of internal implementation, the published services are accessible to the business process with the same methodology. In this way, BPEL provides a platform-independent language for describing the message exchange protocol of the business process.

The approaches to the verification of BPEL processes have been very different. One of researchers about this topic is Bultan, that in [4] represents a business process through: 1- a conversation schema, that identifies every entity and every message sent among entities, 2- a set of guarded automata, that identifies the behavior of every entity. The verification of web service interaction is performed after a further translation in Linear Temporal Logic, that can be submitted to Spin. However, this analysis exclusively involves synchronous communication, where messages are immediately consumed by the web service receiver. The limit of Bultan's paper is overcome by Pistore et al. in [5], where BPEL processes have an asynchronous exchange of messages, represented by message queues. Then, this model of BPEL processes is formalized in CTL propositions for SMV model checker.

Adopting the same methodology, we represent business processes, which are described through BPEL, in formal models that include specific correctness properties related to message exchange protocol. Then, we turn these models into SMV models, so that NuSMV checker is able to verify the validity of such properties and we are able to locate and correct possible errors in the business process design through counterexamples.

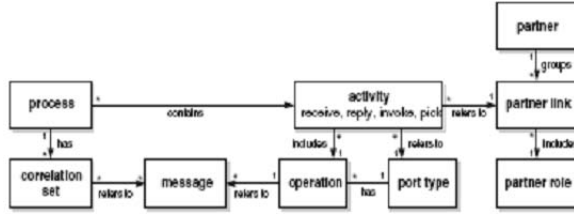


Figure 1. BPEL metamodel.

3. Formal model

The papers, that are described in the previous section, model a business process adopting the point of view of "conversation" among web services or adopting the point of view of the activities, which are performed in the business process and are described through UML activity diagrams. However, we know that activity diagrams do not follow Object-Oriented paradigm [6], therefore we choose to model business process with a Object-Oriented-based scheme and we approve an approach focused on the communication.

In fact, BPEL describes a business process with: 1. a set of partners, having own features defined in partnerLink, partnerRole and Activity 2. a set of messages, sent among partners that invoke operations of web services through connections to portTypes

BPEL metamodel is introduced in the following figure:

We model every single web service partner as a finite state machine, which is represented with a graph having nodes and arcs. Then, we model the communication among partners through a set of activities that synchronize the states of the different partners. For this reason, entities and messages are the fundamental components of our formal model.

An entity identifies a web service partner in the business process and it is marked by specific attributes.

Definition 1. Given a set of attributes A_i that mark the entity i , every attribute $a \in A_i$ can have one of the possible values that belong to the set: $V = \bigcup_{a \in A_i} V_a$

The entity state is identified by the value of the attributes of the entity in the instant of execution.

Definition 2. Given a set of attributes A_i that mark the entity i , the value of all the attributes in the current instant defines the state s of the entity: $\forall a \in A_i, s(a) \in V$ An entity can evolve from a state to another state through a transition, that is determined by the execution of an activity that manipulates in atomic way the value of one or more attributes. So, a transition is the connection among two states of the entity: origin state and destination state.

Definition 3. Given an entity i , a transition T is defined by a binary relation among a pair of states, s_1 and s_2 , that belong to the set of possible states S_i of the entity:

$T : s_1 \in S_i \rightarrow s_2 \in S_i$ A transition can be defined by pre-conditions and post-conditions: generally speaking, a condition is a Boolean expression that returns a true or false value according to the value of a control attribute of the current state. If pre-condition has a positive value, activity is performed and a new state is reached.

A transition can be non-deterministic, so that the same message makes a state evolving in different successor states that are all equally possible. For instance, an activity can complete in attended way or it can produce computational errors or inconsistent data. Fault tolerance is examined in building the formal model: there are successor states that model network problems or non-responding servers as well as other problems that occur at the technical level at runtime; in this way, they are amenable to model checking.

Therefore, it is possible a transition from the current state in one or more successor states. The set of successor states can be defined as $S' = s' S | T : s T s'$.

However, it is also possible that transitions from the current state to successor states do not exist, therefore we define the current state as a final state. The set of final states can be defined as $S_f = s_f \in S | \text{not } T : s_f T s'$

When there is a state transition among pairs of states that do not belong to the same entity, it represents a message that is exchanged among two entities.

Definition 4. Given two different entities i and j , a message M is defined by a binary relation among a pair of states $s_1 \in S_i$ and $s_2 \in S_{j \neq i} : M_{i,j} : s_1 \in S_i \rightarrow s_2 \in S_j \neq i$ A message can be a request of services or a response among two entities that are partners in the business process and it is represented through control conditions, that force sequential operations. For instance, the service of another entity can be invoked only if the current entity has completed its elaborations. A control attribute can be used for marking if elaborations are finished. A possible execution of the business process is a sequence of states and it is determined by state transitions of the same entity and messages sent among different entities.

Definition 5. Given n entities, a path of communication is a sequence of states s_0, s_1, s_2, s_3 . where $s_0 \in S_1 \xrightarrow{T} s_1 \in S_1 \xrightarrow{T} s_2 \in S_1 \xrightarrow{M_{1,7}} s_3 \in S_7 \xrightarrow{T} s_4 \in S_7 \xrightarrow{M_{7,4}} s_5 \in S_4 \xrightarrow{T} s_6 \in S_4$

According to the previous definitions, in our formal model the concept of business process is defined as following:

Definition 6. A Business Process is the set of all possible sequences of execution s_0, s_1, s_2, s_3

So we build a formal model of a BP as a unique graph with all sequences of the set. We will verify in this graph the soundness of specific properties, that are defined in Computational Tree Logic and described in the following section.

4. Correctness Properties

The verification of a business process involves many objectives: it has to validate the system, i.e. to examine if system behavior is consistent with the attended behavior; it has to verify the system, i.e. to prove specifications of correctness; it has to study system performance, i.e. to consider resources, timing and delays, throughput

In this paper, we focus on the verification of properties that include a temporal order. To this purpose, we exploit the expressiveness of temporal logics, that are been applied to the verification of similar properties in other domains as digital circuits, protocols of communication and software development. Particularly speaking, we adopt CTL, because it allows to define specifications not only along a single computational sequence but also some or all the possible executions of the business process. Therefore, it is possible to define:

- "invariant" properties, that have to be validated in all states of all sequences of execution of BP; for instance, we can check that the value of a specific control attribute is true in all reachable states or we can notice activities that must not be performed in any case;

- properties of final states of all sequences of execution; for instance, it is possible to verify the objectives, checking the final value of specific control attributes or it is possible to prove the absence of faults or deadlock in the BP definition;

- temporal properties that verify the coherence of activities performed along the sequences of execution; for instance, we can check if there is an answer message (affirmative or negative) following a message of service request. Such properties are predicates that often impose preconditions or postconditions in the execution of activities of the business process.

The correctness properties must be defined by designers inside BPEL description of the business process through Xpath predicates, then they are automatically translated in CTL by our framework of automatic verification, described in the next section.

5. Framework Architecture

We know that it would be too much difficult for a BP designer to study all the principles of our formal model and to opportunely apply the methodology of model construction and verification for effecting correction and refinement of BP design. Contrarily, a designer is able to provide a description of BP through BPEL, that explains the business logic of the business process in XML standard language.

To use BPEL description in conjunction with Symbolic Model Checking technique for automatic verification, we implement a framework which is able to automatically



Figure 2. Role of BPEL2SMV component.

transform BPEL design in the corresponding formal model; then, the formal model is expressed in NuSMV input language and it is given as source program to the model checker.

The following figure underlines the role of this framework, called BPEL2SMV:

This component has three main packages, according to the three principal functions that it completes:

- BPEL Manager.

It imports BPEL design in a file that is structured in XML language, then it analyzes BPEL file and extracts every information about the finite state machine, representative of every web service partner, and about messages in the business process. The result is a set of graphs (one for every entity) and a set of conditions (one for every message); instead, correctness properties will be assigned in Model Manager package.

- Model Manager.

It is the principal package of the tool, because it calls the main classes of connected packages to start operations as model creation or SMV code creation. It internally memorizes graphs in suitable data structures, that is a set of adjacency matrix; it manages the assignment of labels of correctness properties to the states of the formal model and then adds CTL specifications to the model to be subsequently verified through model checker.

- SMV Manager.

It effects analysis of all states of every graph of the formal model and recovers information for the transformation of every graph in one module of SMV code. It builds the content of a specific module (Monitor), that models messages of the formal model, then it inserts correctness specifications in the main module. Finally, it submits SMV code to automatic verification process through NuSMV model checker.

NuSMV model checker automatically performs verification of CTL specifications on the formal model and provides counter-examples if there is a violation of some correctness property. This gives designers the opportunity to understand wrong BP behavior, to locate errors and to effect right changes for correcting BP design. Then, the modified design is again submitted to model checker for verification of correctness properties. This methodology determines a gradual correction and refinement of BP design, before it is definitely implemented.

6. Conclusion and Future Development

In conclusion, in this paper we propose a formal model of a BPEL description of a generic business process and we build a tool for supporting model checking-based verification of BP design, using our formal model.

The main purpose of our Model Checking-based tool is to unify phase of BP planning with phase of BP verification in a single automatic activity, so that:

- BPEL2SMV performs an "a priori" verification of BP design, with notable benefits in term of saving of development time and testing costs and in term of increasing of quality;

- the verification is automated because BPEL2SMV embeds NuSMV model checker, which replaces logical reasoning in the task to establish system correctness;

- BPEL2SMV is able to prove or to deny system correctness, since it effects an exhaustive evaluation of specifications in all the possible states of the system;

- verification using Model Checking is formal, because correctness specifications are precise mathematical propositions and the formal model is not ambiguous but declared according to a syntactic notation, which is established by model checker; in this way, the model checker is able to perform analysis with rigorous methods.

In the future, we extend this framework with modules, which translate BPML business process in our formal model. Besides, the framework will embed other model checkers like SPIN or UPPAAL.

References

- [1] BPML. Business process modeling language.
- [2] DMAL-S. Daml-s and owl-s.
- [3] T. A. et al. Business process execution language for web services. Version 1.1 Specifications (2003).
- [4] X. Fu, T. Bultan, and J. Su. Analysis of interacting bpel web services. In *Proc. of the 11th Intl. World WideWeb Conf.*, 2004.
- [5] R. Kazhamiakin and M. Pistore. A parametric communication model for the verification of bpel4ws compositions. In *Proc. of the WFSM*, 2005.
- [6] A. Kleppe and J. Warmer. Unification of static and dynamic semantics of uml-a study in redefining the semantics of the uml using the uml oo meta modelling approach. In *Technical report*, 2005.
- [7] L. Guo, Y. Chen-Burger, and D. Robertson. Mapping to business process model to semantic web service model. In *Proc. of the IEEE International Conference on Web Services*, 2004.
- [8] M. Mongiello, D. Castelluccia, R. Totaro, and M. Ruta. A model checking-based tool for verification of web applications. In *SVV 2005*, 2005.
- [9] M. Mongiello, R. Totaro, D. Castelluccia, F.M. Donini, and G. Piscitelli. A model checking based approach for web applications design verification. In *AICA*, 2005.
- [10] S. Narayanan and S. McIlraith. Simulation, verification and automated composition of web services. In *Proc. of the 11th Intl. World WideWeb Conf.*, 2002.
- [11] E. D. Sciascio, F. Donini, M. Mongiello, R. Totaro, and D. Castelluccia. Design verification of web applications using symbolic model checking. In *ICWE*, 2005.
- [12] WSCI. Web service choreography interface (wsci).