Design and Analysis of the Composed Telecom Services

Piergiorgio Bertoli[1], Laura Ferrari[2], Raman Kazhamiakin[1],
Corrado Moiso[2], Marco Pistore[2], and Ermes Thuegaz[2]

[1] FBK-Irst, via Sommarive 18, 38050, Trento, Italy
{bertoli,raman,pistore}@itc.it
[2] Telecom Italia, via G. Reiss Romoli 274, 10148 Torino, Italy
{corrado.moiso,laura1.ferrari,ermes.thuegaz}@telecomitalia.it

Abstract. Telecommunication (TelCo) is a key applicative area where adopting the web service paradigm has an enormous potential to facilitate the development process of powerful, complex functionalities on top of existing ones. At the same time, the specific features of TelCo applications pose difficult challenges to the adoption of standard languages and tools for web services. For instance, they need to cope with *asynchronous* communications, driven by *heterogeneous events*, and to handle *concurrent, long-running* transactions, involving the interaction of *dynamically evolving sets* of partners.

The contribution of this paper is twofold. First, based on our analysis of current TelCo standards, and referring to a real-life case study, we identify and discuss a restricted set of *orchestration patterns*, and show how these can be modeled using the de-facto standard language for web services, WS-BPEL. Then, we confront with the crucial issue of guaranteeing the correctness of such models, providing an automated support for the formal verification of their behavior, based on specific and advanced model-checking techniques. Our tests on the reference scenario witness the effectiveness of the approach, and identify its limits.

1 Introduction

Telecommunication is one of the most important applicative areas in terms of economic and social impact; recent technological advances have triggered an enormous growth in this area, both in terms of the number and complexity of available services. Service-oriented architectures based on the Web service paradigm open up a possibility to establish new, more powerful services by suitably combining the ones already available in a convenient and efficient way, and it is therefore natural that TelCo applications are rapidly evolving in this direction.

Recent trend in TelCo domain is to adopt Web service technology to expose TelCo network capabilities (e.g. call control, sending/receiving messages, access information on end users) to the application deployed in 3rd party domains. Apart from the complexities inherent to the design of generic service compositions, such adoption has to face the problems that are specific for the TelCo capabilities and applications. In particular, they strongly rely on *asynchronous communications* to handle *heterogeneous events* generated by network resources; moreover, they usually handle large, and possibly dynamically evolving, sets of *concurrent processes* which realize *long-running transactions* composed of logically correlated communications. Therefore, the language used

E. Di Nitto and M. Ripeanu (Eds.): ICSOC 2007 Workshops, LNCS 4907, pp. 282–294, 2009.

to define composed TelCo services must provide adequate means to dynamically create, destroy and synchronize execution threads, and to build and maintain logical associations of (asynchronous) communications to a long-running transaction. The complexity of such mechanisms, the need to consider all possible interactions and critical runs between independently evolving processes, makes standard testing techniques insufficient for composition correctness, and call for the adoption of automated verification techniques. In turn, also due to the features above, supporting automated verification is an extremely difficult task; so far, even large companies such as Telecom Italia lack an integrated verification process in the context of a service creation environment, and rely on the very limited support provided by proprietary platforms [1].

This paper confronts both with the issue of designing composed TelCo services using standard languages and tools, and with that of enforcing their correctness by providing by means of automated verification tools.

More specifically, based on our analysis of TelCo applications and standards, and making reference to a real-life case study, we identify specific *orchestration patterns* which capture the relevant features of TelCo composed services. Based on these, we evaluate to which extent the de-facto standard language for web services, WS-BPEL, is apt to represent and execute composed TelCo services, identifying strengths and weaknesses of the language, and directions for improving it.

Then, we tackle the fundamental issue of automatically verifying whether a composed TelCo service, interacting with its partners, satisfies its specifications - e.g. absence of deadlocks or livelocks induced by critical runs, constraints on message orderings, data- and time-related behavioral properties. We exploit advanced techniques presented in [2,3,4], experimentally demonstrating their effectiveness over our reference case study, and discussing their limitations.

The paper is organized as follows. Section 2 provides the framework for TelCo composed services, introduces a reference scenario, and discusses the problems related to the design and analysis of such applications. Section 3 discusses the issues in representing TelCo composed services using the WS-BPEL language, identifying the key patterns for these applications and describing the way they can be implemented in WS-BPEL. Section 4 discusses the formal verification, describing its underlying framework and an experimental evaluation. Section 5 wraps up discussing related and future work.

2 TelCo Composed Services

TelCo applications encompass a huge variety of scenarios, from distributed social networks based on functionalities deployed on mobile phones to multimedia broadcasting, B2B and B2C linking and so on. Among these, a particularly interesting and relevant scenario is the one where a *composed* service orchestrates a number of *component* services that interface existing TelCo capabilities. This situation models the more and more frequent scenario, where the availability of a set of services is the starting point to provide a more powerful combined service either to a end user or to a business customer. Figure 1 depicts the reference framework for such a composition: a composite service interacts with one or more component services that provide uniform interfaces to TelCo capabilities based on heterogeneous, specific protocols. In turn, each component service

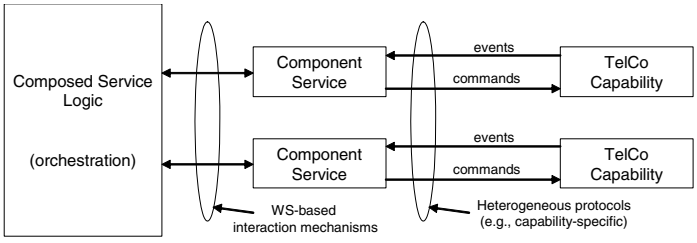

Fig. 1. Composed TelCo service model

may interact with several composed services, thus being involved in several business transactions at the same time. This requires an association of communications to their logical thread, avoiding unwanted interferences among different transactions.

The design and development of such kind of applications pose relevant challenges to the existing languages, tools and methodologies. To ground our discussion, we first introduce a reference TelCo application that, while being simple enough to follow, presents most of the challenging features that can be encountered in such a framework.

2.1 The Case Study: Multimedia Conference

Our reference application is a multimedia conference (MMC) service, offering the possibility to organize a multimedia conference given the availability of multimedia clients. More specifically, the service must enable a user (the "owner" of the MMC) to set up a conference by inviting some partners. (We assume that the partners have previously registered to some sort of registry, e.g. a DNS service; such registration phase may take place using standard TelCo infrastructures). Each of the partners must either acknowledge or refuse the conference invitation, within a timeout. Once the conference is set up, participants may change their status w.r.t. the conference: for instance they may be momentarily busy and unable to respond; they may signal these changes to the owner, as well as, of course, hang off. Also, the MMC owner has the right to delete single participants from the conference, as well as to shut down the conference. In both cases, the participants are notified of the owners decision before being disconnected. To realize this scheme, the logic of the composed MMC service has to interact with a variety of actors, as shown by the architecture in Fig. 2. More specifically:

- *Owner*: identifies the user who manages the multimedia conference; the Owner can create and terminate a conference, and he can also invite or delete new participants.
- *Clients*: identify new participants to a multimedia conference created by the Owner. A new participant is invited to a conference and is notified of this invitation. In order to receive such notifications, a Client must previously register to a DNS service providing its notification reference address (e.g. a URL of a notification Web Service). The Client must accept the invitation to connect to the multimedia conference.
- *MMC process*: the composed service that is used to manage the multimedia conference. It receives the Owner commands and orchestrate the clients and the multimedia conference Web service.

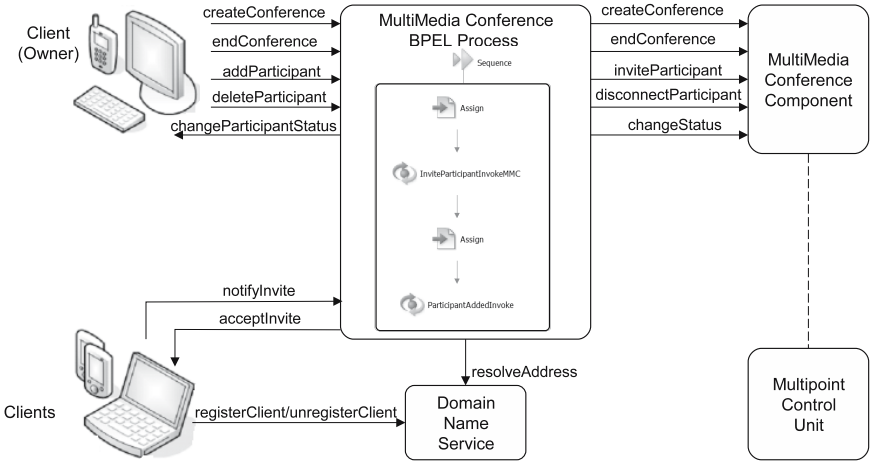

Fig. 2. Reference architecture of the multimedia conference scenario

- *Domain Name Server* (DNS): it receives the client registrations and provides information about clients to the MMC process when the process needs to send an invitation to a new client.
- *Multimedia Conference Component* (MCC) Web Service: it is the interface module between the MMC process and the resource adapters. The MCC service provides the same interface to the MMC process independently from the resource adapters used in the implemented solution. The architecture used is the same adopted and defined in standard TelCo approaches such as Parlay-X [5].
- *Multi Conference Unit* (MCU): it is the module implementing the network functionalities. These interfaces are depending on the network solution adopted in the implementation and are out of the scope of this document.

All the actors are assumed to provide WS interfaces for their interactions. While we aim at expressing the MMC process in WS-BPEL, this is not required for the remaining actors.

2.2 Design and Verification Issues

While conceptually simple, the MMC scenario presents features that pose relevant development challenges, both concerning the implementation using a standard Web service language such as WS-BPEL, and the ability to check their correctness via an automated verification procedure – an essential functionality given the complexity of the implementation.

In particular, the actors taking part to the scenario are independent; they perform asynchronous message exchanges. Moreover, the MMC must react on the basis of data-related as well as time-related conditions (e.g., message correlations identifying long-running transactions with the clients, and timeouts). Finally, since the number of

participating actors is not known a priori, it is necessary to dynamically create, synchronize and destroy execution threads, appropriately associating them with transaction contexts.

These features have an obvious impact on the complexity of the design, making the testing techniques insufficient for guaranteeing the implementation correctness even for simple properties, such as absence of deadlocks and livelocks. On the other side, modelling and formalizing the above phenomena drastically increases the complexity of the analysis, and restricts the applicability of the existing verification approaches. In the following sections, we tackle, in turn, the issues in designing and verifying composed TelCo applications, such as MMC.

3 Design of TelCo Composed Services

Our analysis of TelCo service standards, such as Parlay-X [5] and OSA-Parlay [6], reveals that the above features of the MMC are indeed general in the setting of composed TelCo services. In particular, these features appear together within a restricted set of *orchestration patterns* that we identified as characteristic of composed TelCo services. Such patterns can be understood as the basic bricks that can be used to drive the design and implementation of composed TelCo Services. We describe them in turn.

Event-Based Notification. This pattern, schematically represented in Fig. 3 (left), considers the typical situation where a set of possible (heterogeneous) events is defined, and the orchestrator is interested in being notified of some of them. In our example, the MMC needs to be notified of changes in in status of the participants; in particular, with the *inviteParticipant* message, the MMC subscribes for receiving notification messages, delivered as *changeStatus* messages. We remark that this situation is typical for many of composed TelCo services, e.g. when tracking the location of mobile terminals in the context of fleet management. To realize this scheme, two different mechanisms are combined: an event subscription/unsubscription, by which the component service dynamically determines which events he is interested in; and a notification, by which the component delivers the event to the composed service.

Event-Based Solicit-Response. Analogously to the event-based notification pattern, this pattern accounts for the existence of a set of events; but in this case, the orchestrator service is interested in directly handling them, rather than being notified of them. That is, the notification mechanism is replaced by a specular mechanism where the composed service takes the event in charge, and delivers some result back to the component. We omit a schematic representation for lack of space.

Solicit-Started Transaction. This pattern, schematically shown in Fig. 3 (right), represents the typical TelCo situation, where a set of services is handled concurrently, and each of them may originate a long-running transaction. In the our scenario, MMC must handle a set of conference participants not known at design time: every invited client is processed independently, and may go through a sequence of status changes

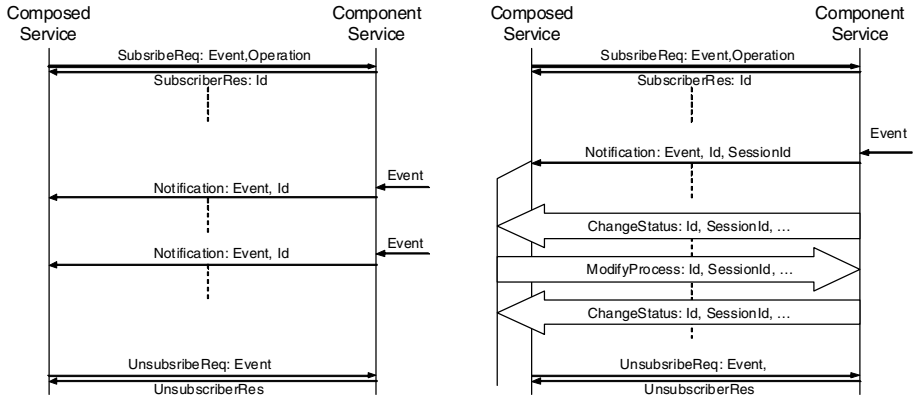

Fig. 3. Application Patterns

before terminating. This requires that each client transaction is uniquely identified by a transaction context, and handled by a separate thread. Similar situations take place e.g. within the context of the routing of incoming calls according to some customized rules. To realize this, not only the orchestrator must be able to dynamically determine the set of events he is interested in, but also to correlate different communications to the corresponding long-running transactions, avoiding interferences between them. That is, this pattern is characterized by an ability to generate multiple threads on demand, and to associate their interactions to a unique 'transaction context' that must act as the selection criteria to dispatch communications to the proper thread.

Request-Started Transaction. This pattern is analogous to the one above, but the handling of events is taken in charge by the composed service (within a separate transaction), rather than notified. This means that, in a way similar to the event-based solicit-response, the central notification mechanism is replaced by a solicit-response mechanism to enable the event handling on the side of the component.

We remark that these patterns are derived from the analysis of applications, where the business logics is interleaved with the TelCo capabilities, controlled through standard Parlay-X or OSA-Parlay interfaces. In this way, the patterns concentrate on the management of TelCo capabilities within composed 3rd party application, and encompass diverse aspects of the compositions, such as interactions or transaction management. This makes them rather different from the patterns identified in [7], which focus on communication alone, and, originating from a general analysis, discard some relevant TelCo-specific features. Naturally, certain patterns presented in [7] can be used to capture portions of TelCo applications, and the elemental communication schemata (corresponding to WSDL operations) appear both in interaction and orchestration patterns. The above orchestration patterns, instead, stem from an analysis upon business protocols rather than at a communication level, and this explains their diversity w.r.t. purely interaction patterns.

3.1 WS-BPEL Modeling

We now discuss the possibility to represent the above patterns by means of the WS-BPEL language. To do so, we associate each pattern with a set of language requirements, and consider the way these requirements can be addressed in WS-BPEL:

1. the ability to handle asynchronous communications, as well as to suspend activities on messages, and to resume them upon timeouts;
2. the ability to handle long-running transactions, that is, to generate unique identifiers, associating them to logical flows of interactions, and using them to identify the messages belonging to them, dispatching them to the correct thread;
3. the ability to handle subscription-unsubscription to event sets.

Moreover, the solicit-started and request-started transaction patterns require the ability to generate, synchronize and destroy parallel threads of computation.

Several of the above features are directly supported by WS-BPEL, and therefore can be readily implemented. In particular, WS-BPEL provides primitives for both the asynchronous and synchronous interactions; it allows for realization of suspending over fixed sets of events and timeouts using "pick" construct and event handlers.

However, having the component service able to specify exactly which events he is interested in handling is not trivial, since at this stage of evolution, there is no immediate way to specify, within a message, that a single operation has to be handled. One possible, still partially satisfactory, workaround to this is to use WSDL interfaces as operation containers, and specify the WSDL reference for the kind of operations (events) that need be considered. For instance, in the MMC scenario, the "changeParticipantStatus" operation is specified into a WSDL file which is referred to when subscribing to the notification of such messages.

Finally, managing a dynamic set of components and the associated long-running transactions, is possible using WS-BPEL, albeit not trivial. In particular, this requires structuring the process that spawns threads into two separate WS-BPEL entities, and making use of two kinds of WS-BPEL constructs: creation and synchronization of process instances, and message correlation. The two WS-BPEL entities play the role of a "dispatcher" and an "event handler" respectively: the dispatcher generates instances of the handler, and the handler manages the events and transactions associated to a particular thread by making use of unique message correlation sets. In our scenario, the MMC "dispatcher" intercepts the invitation requests generates new instances of the "participant handler" that handle one of the participants. The latter are uniquely recognized by their URL, which is used as a dynamic reference point and establishes the message correlation.

4 Formal Analysis of Composed TelCo Services

Service compositions in TelCo domain make the highest demands of the correctness analysis methods. Indeed, the ultimate use of concurrent asynchronous interactions, data- and time-based reaction rules, dynamic creation and synchronization of execution threads not only make the testing techniques insufficient, but also require substantial extensions of standard automated formal verification techniques. In particular,

specific techniques are required for the adequate representation of asynchronous message exchanges and queues, for modelling and reasoning on complex XML data, process instantiation, timed properties. Oversimplification of these problems by existing approaches may often lead to incomplete or incorrect analysis results.

Below we introduce a formal framework apt to represent composed TelCo services at an appropriate level of abstraction, enabling its automated verification.

4.1 Analysis Framework

Our framework enables the verification of TelCo composed services against behavioral properties (e.g., absence of deadlocks): the composition specification, given as a set of interacting WS-BPEL processes, is automatically translated into a corresponding formal model, which is then processed, analyzed, and finally verified by the back-end verification engines.

In order to address the features peculiar to the TelCo domain, the framework integrates a variety of special models and techniques, such as analysis of asynchronous interactions, data- and time-related properties [2,3,4]. The underlying formal model allows for representing a composition specification given as a set of interacting WS-BPEL processes. It focuses on modeling the execution of the composition participants, and on the representation of asynchronism of the message exchanges.

The behavior of the composition participants is captured by the *State Transition System* that reflects the *control flow* (i.e., the evolution of the composition and its participants); *data flow* (the modification and exchange of the business data during interactions); and *time flow* (the timed properties of the model, such as long running activities and timeouts) of a participant.

Asynchronous interactions are captured by the *Communication Model* that represents message exchange with a set of (ordered/unordered, bounded/unbounded) queues.

Managing Asynchronism. The key feature of the communication model in our formalism is that it is parametric with respect to the queue structures and properties.

In [2] we discussed how the difference in these parameters affects the behavior of a composition. We also presented the *adequacy analysis*, a way to identify a communication model that allows us to consider the most complete set of the composition executions, while being as simple as possible for the implementation. The obtained results allows one to see how the concurrent message exchanges affect the behavior, whether the messages can be left in the queues without being ever consumed, or the queue content may grow unboundedly. Given these features, a more careful analysis of a wide range of asynchronous scenarios and patterns, such as those appearing in MMC, is possible.

Managing Data and Time Properties. Our formalism allows also for representing the data and time aspects of the service specification. It provides a way to express higher level properties, such as XML data operations and functions, activity durations or quantitative behavioral properties. These modelling capabilities are supported with the corresponding analysis techniques. In particular, we propose data abstraction and

refinement techniques for XML data handling [3]; qualitative and quantitative algorithms are proposed for the timed analysis [4].

We remark that, while these properties play an important role for the representation and analysis of service compositions, in many existing approaches these aspects are omitted, or at best considered in a very restricted way, thus often leading to incomplete and incorrect analysis results.

Managing Multiple Process Instances. In TelCo applications it is often the case that the number of composition participants is not fixed: new instances are created and destroyed dynamically. This restricts the use of finite state verification techniques, such as model checking. Indeed, such techniques require a priori knowledge about the system state space, which is violated in these settings.

In order to address this problem we have extended the previous approach as follows. The composition specification is parametric with respect to the (maximum) number of instances of each participant class. Every instance has a unique ID that represents the correlation identity. The identification and routing of messages to a proper process instance is then determined by the message contents. A concrete, finite state composition model, and the corresponding message routing schema are generated automatically. In the future, we plan to relax these constraints, and develop techniques that deal with arbitrary number of process instances, e.g., similarly to the approach presented in [8].

The presented verification framework is implemented as a WS-VERIFY toolkit, distributed as a part of the Astro project (`http://astroproject.org`). It incorporates the translation and verification capabilities presented above, and exploits the two state of the art model checkers, namely NuSMV [9] or SPIN [10], as the verification engines. Besides predefined behavioral properties (i.e., freedom of deadlocks or livelocks) the toolkit allows for specifying complex requirements, such as reachability properties, data and time flow requirements, orderings rules between process activities, etc. A Linear-time Temporal Logics (LTL, [11]) is exploited for these purposes.

4.2 Verification of the MMC Application

We used this framework for the verification of the MMC application. The composition specification has been incrementally refined; different implementations scenarios have been automatically analyzed. All the actors of the composition have been represented with the corresponding WS-BPEL process that defines its behavioral protocol.

Nominal Scenario. The nominal scenario of the composition is defined by the following steps:

1. Upon receiving a message from the owner, a new instance of the MMC process, dispatcher part, is created. The dispatcher request the MCC Web service, acknowledges the owner, and creates a new subprocess (participant handler) that manages events associated to the conference owner.
2. The owner may add a participant invoking an appropriate operation of the MMC process. The dispatcher creates a new subprocess for managing the corresponding client. The subprocess notifies a client about the invitation, waits for acceptance,

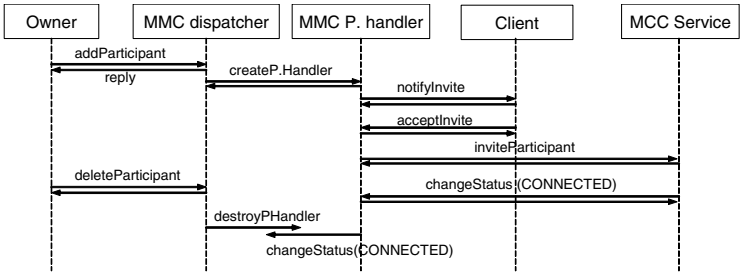

Fig. 4. MMC counterexample

and requests the MCC service to add the client to the multimedia conference. In this way, it also subscribes for notifications on the status changes of the client.

3. In a similar way the participants are removed, and the the conference is terminated. The MMC dispatcher process (and the participant handlers) interacts with the MCC service in order to perform the requested operations.
4. The MCC service notifies the participant handlers about the status changes of the clients (e.g., disconnect from the network). The status is propagated to the MMC dispatcher process, which handles the conference as a whole. When the owner has disconnected, the conference should be terminated, and the MMC dispatcher sends an appropriate request to the MCC service.

Adequacy Analysis. The MMC application exhibits high level of concurrency among interactions. Consequently, the adequacy analysis applied to the scenario reports that the composition is sensitive to various message reorderings: the communication model, where the messages are consumed from the queues in arbitrary order, is needed for capturing all possible composition executions. Indeed, the owner may perform the requests independently of the status notifications from the MCC service that in turn may be independent from the interactions between the MCC process and its subprocesses. Depending on the ordering of these events, different conversations occur.

Requirements Verification. Having defined the communication model, we verify the composition against behavioral properties, for example:

- *Deadlock freedom*: each participant should eventually terminate in its final state.
- *Possibility to successfully complete*: there exists at least one execution that satisfies the nominal scenario (i.e., the conference is terminated and MMC process received a reply from the MCC service).

In case of violation the tool reports a counterexample trace (e.g., in the form of Message Sequence Chart), as the one presented in Fig. 4. Here the tool reports a violation of the deadlock freedom, which can be explained as follows. The conference owner requests a new participant to be added, and the MMC dispatcher process immediately acknowledges the request. Then the latter creates a new instance of the MMC participant handler

Table 1. Experimental results

Specification Size (states/trans/vars)	Adequacy (time)	Global Model Size (states/trans)	NuSMV (time/memory)	SPIN (time/memory)
123/133/49	0.14sec	1407/1487	6sec/16Mb	2.6sec/1.57Mb
181/199/88	0.30sec	3185/3420	29sec/25Mb	4.5sec/1.67Mb
191/216/80	0.59sec	7767/8412	326sec/33Mb	3.5sec/1.68Mb

that initiates the corresponding interaction sequence with the client (*notifyInvite* and *acceptInvite* interactions) and with MCC (*inviteParticipant*). Meanwhile, for some reason the owner decides to remove the participant (*deleteParticipant* interaction). At the same time the participant is connected, which is is notified by the MCC service. At this point, two concurrent request-response operations are initiated: the MMC dispatcher process requests the participant handler to finish (*destroyPHandler*), while the latter tries to propagate the status update. As a result, the two processes are mutually blocked.

The problem here is due to the fact that the owner does not know when the participant is actually connected. The problem may be resolved in several ways. First, instead of immediate response, the MMC dispatcher may reply to the *addParticipant* request only when the corresponding status update is received. Second, it is possible to propagate the status update directly to the owner. Third, the two-way notification may be replaced with the one-way, and therefore the MMC participant handler will not block.

Verification Results. We verified the above behavioral properties of the MMC application using the WS-VERIFY tool and exploiting both the NuSMV and SPIN model checkers for comparison. We experimented with several implementations of the composition that differ both in the implemented features and the patterns used. The results of the experiments are presented in Table 1. For each version the table reports the size of the specification (total number of local states/transitions/variables), the time of the adequacy analysis (AA), the size of the generated global model[1] (GM Size, states/transitions), and the verification time/memory use under different model checkers.

The size of the global model shows that the system is highly asynchronous and concurrent: the interleaving of activities and their mutual dependencies lead to the considerable difference between the specification size and the corresponding global model. This explains also better performance of the SPIN: it is implemented specifically to deal with highly asynchronous systems. These results, however, does not hold for the situation, when the system does not violate a property. In this case, the SPIN model checker has to perform the complete exploration of the state space and the performance drastically decreases, while this does not hold for NuSMV, where symbolic techniques are implemented. Also certain features (e.g., data non-determinism) are difficult to efficiently represent in SPIN, thus restricting the applicability of certain kinds of analysis (e.g., data abstractions) possible with NuSMV.

[1] We remark that here the reduced numbers are provided, achieved using the Partial Order Reduction techniques [12]. The actual number are orders of magnitude bigger.

We remark also that the MMC case study, and the specification we were able to verify with our tool are considerably more complex and expressive than those found in the works related to ours (see, e.g., [13,14]).

5 Related Works and Conclusions

In this paper we presented our approach to the design and correctness analysis of Web service compositions in TelCo domain. We introduced a set of orchestration patterns that form the building blocks for the development of TelCo composed services, and demonstrated how such compositions may be realized using the WS-BPEL language. We stress the fact that these patterns are focused on the ways the TelCo capabilities available through standard interfaces are controlled within composed business processes, and therefore incorporate different aspects of the business logics, such as communications and transactions. In this way, the patterns are different from those discussed in, e.g., [15,7], which focus on a particular aspect, and, originating from a general analysis, do not capture the TelCo-specific application features. We also remark, that certain properties of the presented patterns (e.g., to have multiple transaction manged by the same service instance) require special ways to be modelled and implemented in WS-BPEL.

We also presented a framework for automated verification of TelCo service compositions. In the framework we have integrated and extended our previous results, in order to uniformly address the important features of the TelCo domain, such as asynchronous interactions and concurrent events, dynamic process creation, data and time management. The result of the analysis of the MMC case study demonstrate the complexity of the verification in this domain. There exists a wide range of the research work towards formal representation and verification of service compositions based, e.g., on finite state models [14,8,13,16], process-calculi [17,18], petri nets [19,20]. Certain works address also specific challenges, such as asynchronous interactions [21], timed properties [22,23], or data modelling [24]. These challenges, however, addressed only partially, potentially leading to incomplete analysis results in the TelCo domain. On the contrary, the framework presented here aims at integration the approaches to these challenges and provides a deeper insight to the potential problems.

In the future we plan to improve and extend the capabilities of the analysis framework. First, we plan to search for better ways of managing dynamic sets of processes in the verification. Second, we plan to increase the efficiency of the analysis applying the techniques, specifically adopted to deal with the asynchronous systems, such those combining the symbolic model checking with the partial order reduction.

References

1. Turner, G.: Service Creation. BT Technology Journal 13(2), 80–86 (1995)
2. Kazhamiakin, R., Pistore, M., Santuari, L.: Analysis of Communication Models in Web Service Compositions. In: Proc. International World Wide Web Conference (WWW) (2006)
3. Kazhamiakin, R., Pistore, M.: Static Verification of Control and Data in Web Service Compositions. In: Proc. International Conference on Web Services (ICWS) (2006)

4. Kazhamiakin, R., Pandya, P.K., Pistore, M.: Representation, Verification, and Computation of Timed Properties in Web Service Compositions. In: Proc. International Conference on Web Services (ICWS) (2006)
5. Parlay-X Group: 3GPP, Open Service Access (OSA) - Parlay X Web Services (Release 6)
6. OSA Parlay Group: 3GPP, Open Service Access (OSA) - Application Programming Interface (Release 6)
7. Barros, A., Dumas, M., ter Hofstede, A.H.M.: Service interaction patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
8. Fu, X., Bultan, T., Su, J.: Formal verification of e-services and workflows. In: Bussler, C.J., McIlraith, S.A., Orlowska, M.E., Pernici, B., Yang, J. (eds.) CAiSE 2002 and WES 2002. LNCS, vol. 2512, pp. 188–202. Springer, Heidelberg (2002)
9. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NuSMV: a New Symbolic Model Checker. Int. Journal on Software Tools for Technology Transfer (STTT) 2(4) (2000)
10. Holzmann, G.J.: The Model Checker SPIN. Software Engineering 23(5), 279–295 (1997)
11. Emerson, E.A.: Temporal and Modal Logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics. Elsevier, Amsterdam (1990)
12. Peled, D.: Combining Partal Order Reductions with On-the-fly Model Checking. In: Dill, D.L. (ed.) CAV 1994. LNCS, vol. 818. Springer, Heidelberg (1994)
13. Fu, X., Bultan, T., Su, J.: Analysis of Interacting BPEL Web Services. In: Proc. International World Wide Web Conference (WWW) (2004)
14. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based verification of web service compositions. In: Proc. International Conference on Automated Software Engineering (ASE) (2003)
15. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Analysis of web services composition languages: The case of BPEL4WS. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 200–215. Springer, Heidelberg (2003)
16. Nakajima, S.: Model-Checking Verification for Reliable Web Wervice. In: Proc. OOPSLA Workshop on Object-Oriented Web Services (2002)
17. Ferrara, A.: Web Services: a Process Algebra Approach. In: Proc. of the International Conference on Service Oriented Computing (ICSOC), pp. 242–251 (2004)
18. Boreale, M., Bruni, R., Caires, L., De Nicola, R., Lanese, I., Loreti, M., Martins, F., Montanari, U., Ravara, A., Sangiorgi, D., Vasconcelos, V., Zavattaro, G.: SCC: A Service Centered Calculus. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 38–57. Springer, Heidelberg (2006)
19. Schmidt, K., Stahl, C.: A petri net semantic for BPEL4WS – validation and application. In: Proceedings of the 11th Workshop on Algorithms and Tools for Petri Nets, pp. 1–6 (2004)
20. Zhang, J., Chung, J.Y., Chang, C.K., Kim, S.W.: WS-Net: A Petri-net Based Specification Model for Web Services. In: Proc. of the International Conference on Web Services (2004)
21. Fu, X., Bultan, T., Su, J.: Conversation protocols: A formalism for specification and verification of reactive electronic services. In: H. Ibarra, O., Dang, Z. (eds.) CIAA 2003. LNCS, vol. 2759, pp. 188–200. Springer, Heidelberg (2003)
22. Díaz, G., Pardo, J.J., Cambronero, M., Valero, V., Cuartero, F.: Automatic Translation of WS-CDL Choreographies to Timed Automata. In: Proc. International Workshop on Web Services and Formal Methods (WS-FM) (2005)
23. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On Temporal Abstractions of Web Service Protocols. In: Procs. of CAiSE Forum (2005)
24. Deutsch, A., Sui, L., Vianu, V.: Specification and Verification of Data-driven Web Services. In: PODS 2004: Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 71–82 (2004)