Formal Modelling and Verification in Service-Oriented Computing

by Maurice ter Beek, Stefania Gnesi, Fabio Martinelli, Franco Mazzanti and Marinella Petrocchi

Formal methods and tools are a popular means of analysing the correctness properties of computer network protocols, such as safety, liveness and security. First the protocol under scrutiny is described in a formal language, which often results in a more precise definition of its function. Subsequently, the properties to be analysed are specified in a suitable logic. Finally, to decide whether or not the protocol fulfils certain properties, automatic tools are used to analyse it. The outcome either proves the protocol to be correct with respect to the relevant properties or shows there to be a problem.

Recent trends in telecommunication for the delivery of services such as Web applications have highlighted the need to improve and extend the current network protocols. In this context, we have recently been involved, in collaboration with Telecom Italia Labs, in the formal modelling and verification of some protocols for service-oriented computing. This work is part of the EU-funded research project SENSORIA (FP6-2004-IST-FETPI). Its aim is to develop a novel comprehensive approach to the engineering of software systems for service-oriented overlay computers.

Here we briefly describe our experience in evaluating the suitability of the service-oriented application approach for the delivery of telecommunication services, and in investigating possible improvements and extensions. In particular, we present the formal analysis of a protocol for identity management (end-user identity in a federated context; application versus end-user identity), and a protocol definition supported by formal analysis. Readers are encouraged to refer to the website of the SENSORIA project for more details.

Identity Federation Protocols for Web Services

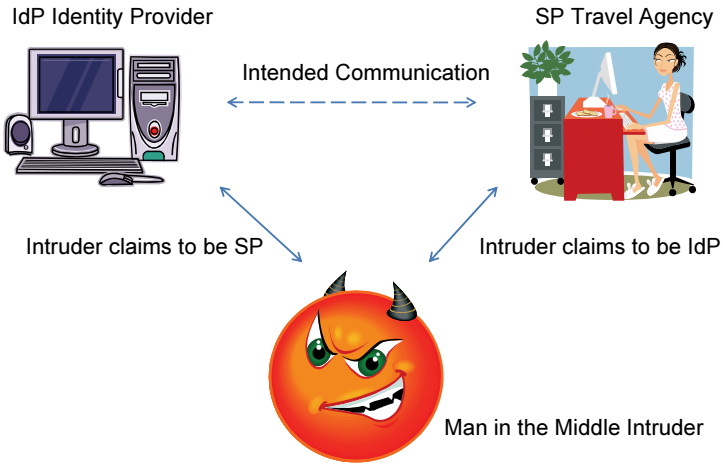
Telecom Italia recently proposed a network protocol that permits end users to access services through different access networks (both mobile and fixed) without explicitly providing any credentials, while the service application level can nevertheless trust the identity and authentication information provided by the access networks. As a result, service providers (SPs) identify a user with the authentication procedure performed by the network provider. After identity federation, a single sign-on suffices for a user to access all services belonging to the same circle of trust of SPs, while keeping personal data private. This is an improvement over having to repeatedly introduce (and remember) one's credentials. This protocol is related to recent solutions specified by Liberty Alliance, who delegate the task of authenticating a user to an identity provider.

We have provided formal analyses of a protocol that transfers identity and authentication information handled by the systems in an operator's network infrastructure to the applications that implement services provided by third-party service providers. These formal analyses verify the correctness of the protocol and investigate its security properties. The analysis results will be used to identify possible flaws or weaknesses in the protocol and to suggest improvements.

As initial steps towards a full formal security analysis of this protocol, we have modelled a number of user scenarios in the process algebra Crypto-CCS. The model checker PaMoChSA (developed by the Security group at IIT-CNR) has then be used to verify their vulnerability with respect to man-in-the-middle attacks by adding a so-called intruder component (see figure).

Asynchronous Web Service Invocation Protocols

Mobile communication networks are typically unstable, since terminal devices may dynamically change their reachability status during their lifetime. Within service-oriented architectures, asynchronous service invocation is often the more suitable paradigm for the choreography and orchestration of mobile components. There is therefore a need for communication protocols that are able to manage both synchronous and asynchronous communication in the presence of unstable network connections. The formal modelling and verification of these protocols is the first step



Verifying vulnerability by adding an intruder component.

towards the successful implementation and evaluation of reliable service-oriented computing applications.

Together with Telecom Italia, we are involved in developing a variant of SOAP coined aSOAP that supports asynchronous communications. This process is driven stepwise by the results of a formal analysis. Hence this activity is different to that described above, where the formal analysis is performed on a protocol already specified in order to verify its correctness. Our aim is to eventually arrive at a proposal of aSOAP of which we can guarantee that it satisfies certain desirable properties. The formal analysis of a preliminary definition of the protocol highlighted several issues and flaws that suggested a complete review was necessary.

We have modelled aSOAP as a set of communicating UML state machines and verified a number of behavioural properties expressed in the action- and state-based temporal logic UCTL. This was done with the model checker UMC, developed in the Formal Methods and Tools group at ISTI-CNR, which creates and traverses the state space on the fly. The advantage of on-the-fly model checking is that often only a fragment of the full state space needs to be generated and analysed to obtain a satisfactory result. The development of UMC is still in progress and a prototypical version is being used internally at ISTI-CNR for academic and experimental purposes. There has not yet been an official public release of the tool, but the current prototype can be accessed via a Web interface.

Links:
SENSORIA:
http://www.sensoria-ist.eu/

PaMoChSA:http://www.iit.cnr.it/staff/fabio.martinelli/pamochsa.htm

UMC: http://fmt.isti.cnr.it/umc/

Please contact:
Maurice ter Beek
ISTI-CNR, Italy
Tel: +39 050 315 3471
E-mail: maurice.terbeek@isti.cnr.it

Challenges in a Service-Oriented World

by Wolfgang Reisig, Karsten Wolf, Jan Bretschneider, Kathrin Kaschner, Niels Lohmann, Peter Massuthe, and Christian Stahl

Interacting services raise a number of new software engineering challenges. To meet these challenges, the behaviour of the involved services must be considered. We present results regarding the behaviour of services in isolation, the interaction of services in choreographies, the exchangeability of a service, and the synthesis of desired partner services.

In a service-oriented world, enterprises use (Web) services to encapsulate parts of their process logic. The Web Service Business Process Execution Language (BPEL) is an established standard to describe Web services. Usually designed in isolation, a service must nevertheless properly interact with other services at run time. A number of techniques have been suggested to check the compatibility of service interfaces and to discover pairs of semantically matching services. The problem of checking behavioural compatibility, however, is rarely addressed. Figure 1 shows typical examples of behavioural incompatibility.

In our joint research groups, and particularly in the Tools4BPEL project, we address behavioural compatibility and provide a series of techniques for detecting and repairing incompatibility, and for synthesizing behaviourally compatible partner services. Here we address the basics of this approach.

Behaviour of Services in Isolation

The designer of a service must guarantee that his/her service is controllable, ie there exists at least one properly interacting partner service. Controllability is thus a criterion that is as fundamental as the notion of soundness in the realm of business processes and workflows. Furthermore, a well-designed service should meet a number of additional properties such as executability of all activities. We suggest techniques for checking this kind of property based on the static analysis of BPEL code and model checking the state space of services.

Interaction of Services in Choreographies

Although a service might be well designed in isolation, it may cause a deadlock in a choreography of services. To prevent such deadlocks, designers must know either how other services in

Figure 1: Services with incompatible behavior (the buyer and seller services wait for each other) and an ill-designed service (a partner of the order service would have to guess the service's capacity).