Sensoria

016004

Software Engineering for Service-Oriented Overlay Computers

www.sensoria-ist.eu

D8.6

Relations Among Case Studies and Theme 2 Results

Qualitative and Quantitative Analysis of Aspects of Services

Lead contractor for deliverable: ISTI
Author(s): Maurice H. ter Beek and Stefania Gnesi (ISTI)
with contributions from L. Acciai, M. Boreale, R. Pugliese, F. Tiezzi (DSIUF), H. Gao (DTU), D. Latella (ISTI), S. Corfini, G. Ferrari, C. Montangero, L. Semini (PISA), A. Clark, S. Gilmore, M. Tribastone (UEDIN) and S. Reiff-Marganiec (ULEICES)

Due date of deliverable: August 31, 2008
Actual submission date: September 22, 2008
Revision: 1
Dissemination level: PU

Contract start date: September 1, 2005 Duration: 48 months
Project coordinator: LMU
Partners: LMU, UNITN, ULEICES, UWARSAW, DTU, PISA, DSIUF,
UNIBO, ISTI, FFCUL, UEDIN, ATX, TILab, FAST, BUTE,
S&N, LSS-Imperial, LSS-UCL, MIP, ATXT, CIR

Integrated Project funded by the
European Community under the
"Information Society Technologies"
Programme (2002—2006)

Information Society Technologies

Contents

1 Introduction **3**

2 Formalise the S&N Credit Request **3**

3 Specifying and Analysing SOC Applications with COWS **4**

4 Design and Verification of Long-Running Transactions **5**

5 Relational Analysis for Delivery of Services **6**

6 Relational Analysis of Correlation **6**

7 A Flow-Sensitive Analysis of Privacy Properties **7**

8 Logic-Based Conflict Detection for Distributed Policies **8**

9 Modelling the Credit Request Scenario in CaSPiS **9**

10 A Methodology to Check Services' Correctness and Replaceability **10**

11 A Model Checking Approach for the Analysis of SOC Applications **11**

12 Formal Verification of an Automotive Scenario in SOC **12**

13 Formal Verification of Secure Service Contracts **13**

14 Specification of a Requirement for an Automotive Scenario with MoSL **14**

15 Safety and Response-Time Analyses of an Automotive Scenario **15**

16 Conclusion **16**

17 Relevant SENSORIA Publications and Reports **16**

1 Introduction

This deliverable provides an overview of the relations between the SENSORIA case studies from WP8 and the technical work carried out as part of Theme 2: *Mathematical analysis and verification techniques and tools for system behaviour and quality of service properties*. This theme deals with logic developments, type systems and static analysis techniques to develop tools and methods to guarantee a high level of security and trust for the location transparent delivery of services, while allowing mobility of resources, as well as with the development of stochastic techniques to analyse quantitative aspects of services. These activities are broken down into two work packages:

WP3 Qualitative Aspects of Services;

WP4 Quantitative Aspects of Services.

This deliverable thus shows how the theoretical approaches to qualitative and quantitative aspects of services from WP3 and WP4 can be applied to some case study scenarios from the Service-Oriented Computing (SOC) domain.

This deliverable is structured as follows. After this Introduction, we present a series of contributions that are the result of applying analysis and verification techniques from WP3 and WP4 (in that order) to case study scenarios. Each contribution is meant to describe the experience of applying a particular technique, method or language to a scenario, and *not* the technique itself (which is presented in deliverables related to WP3 and WP4). As such, the contributions attempt to answer the following questions:

Aim: was there an intention to prove or provide something specific, besides validating the technique, method or language?

Experience: which were the problems (if any) faced when applying the technique, method or language to the case study and what are the results (referring to publications)?

Benefits: what is the advantage of applying the technique, method or language to the case study (from the engineering, scientific or business point of view)?

Feedback: did the application of the technique, method or language to the case study lead to an improvement of that technique, method or language?

2 Formalise the S&N Credit Request

Aim: The objective of this work is to give an introduction to how cryptographic systems, in particular the underlying cryptographic protocols, can be formalised. Specifically, we show how the S&N Credit request scenario from the Finance case study is translated into the Alice-Bob notation. This translation is practical as it underlines the assumptions that protocol analysts usually take, but also because the Alice-Bob notation is the common notation cryptographic protocols are presented in, thus it may ease readability for more experienced readers.

Experience: The case study, in its original form, is in a high-level formalism, in which the underlying key establishment and communication protocols are hidden. However, in order to allow validation of the entire system, we must take all of these underlying protocols into account. Thus we accompany the high-level formulation with low-level specifications of the applied protocols, and then combine these into a low-level specification of the entire system. The result of this research is recorded in [YNNÖ07].

Benefits: The S&N Credit Request combines protocols, that are individually considered secure, into one larger scheme. Former study has shown that security of classical protocols is not compositional; i.e. attacks existed when two or more protocols were used in combination. The formalisation we show in this work allows one to investigate whether the S&N Credit Request avoids such combinatorial attacks.

Feedback: The case study assumes that protocols are running on a structured network in the sense that some services are located in a highly secured zone which cannot be accessed by the attacker. This poses new requirements to the analysis, as classical protocol analysis usually assumes that the attacker controls the entire network and thus these approaches cannot be applied to the case study.

3 Specifying and Analysing SOC Applications with COWS

Aim: By means of two scenarios of the Automotive case study, i.e. On road assistance and Automotive security, we aim at showing that some techniques developed for COWS [LPT07a] can be successfully used to analyse COWS specifications.

Experience: The On road assistance scenario lends itself to a number of different kinds of analysis. In [FGL$^+$08, LPT08], we verify *functional properties* of the described system formally specified using the service specification language COWS. The properties are described by means of SocL [FGL$^+$08], a logic specifically designed to capture peculiar aspects of service-oriented applications, and automatically verified by using CMC [FGL$^+$08, CMC], an on-the-fly model checker for SocL formulae. Some examples of functional properties verified over the system are as follows:

- all garage services contacted by the 'orchestrator' process are *responsive*, i.e. they always guarantee a response to each received request;
- a garage or rental car request can be processed only after the driver's credit card has been successfully charged;
- before looking for a tow truck, a garage must be found;
- if renting a car succeeds and finding a tow truck fails, then the order of the rental car must be cancelled (because the car must be redirected to the driver's current location).

In [LPT08], we check *confidentiality properties* of the same system. Indeed, the type system for COWS introduced in [LPT07b] permits expressing and forcing policies regulating the exchange of data among interacting services and ensuring that, in that respect, services do not manifest unexpected behaviours. This enables us to check confidentiality properties, e.g., that critical data such as credit card information are shared only by authorized partners. A relevant confidentiality property for the On road assistance scenario is that a driver in trouble must be assured that information about his credit card and his location cannot become available to unauthorized users. As another example, the on road services could want to guarantee that critical data sent to the in vehicle services, such as cost and quality of the service supplied, are not disclosed to competitors.

The Automotive security scenario is suitable to validate 'security' techniques. In the full version of [LPT07b] we apply to this scenario the same type based approach used for the other scenario and check the following confidentiality property: information about location of the car of a driver in trouble, communicated by its on-board computer system to a trustworthy automobile association, can be accessed only by trustworthy members, i.e. driver's location cannot become available to unauthorized users.

Benefits: Modelling SOC systems and applications by using the formal specification language COWS enables their analysis by means of methods and tools designed for COWS specifications. By applying to the On road assistance and Automotive security scenarios two different analysis techniques, namely model and type checking, we have verified some properties of interest for the described systems.

Feedback: Our experiments on the automotive scenarios witness that meta-theories, proof techniques and analytical tools developed for 'classical' process calculi, such as model checking and type systems, can be tailored to the needs of service-based applications. This 'proof technology' can eventually pave

the way for the development of automatic property validation tools, like the model checker CMC. Therefore, process calculi, such as e.g. COWS, might play an important role in laying rigorous methodological foundations for specification and validation of SOC applications.

4 Design and Verification of Long-Running Transactions

Aim: We designed and implemented a middleware called Java Signal Core Layer [BFM+05, FGS06b, FGST07] (JSCL). A distinguished feature of JSCL is the strict interplay among formal semantic foundations, implementation pragmatics and experimental evaluation of the resulting programming constructs. More precisely, all available programming facilities have a clear semantics. Indeed, at the abstract level, the middleware takes the form of the *Signal Calculus* [FGS06a, FGS06b, FGST07] (SC). SC is a variant of the π-calculus with explicit primitives to deal with event notification and component distribution. At the implementation level, JSCL takes the form of a collection of Java API equipped with a standard development environment (an Exclipse plug-in). The JSCL API's are available at `www.tao4ws.net`.

We aim at experimenting our framework in the design, implemention and verification of properties of services coordination policies (orchestration and choreography). Our main goal is to show that the formal mechanisms underlying the SC-JSCL framework allow a more robust development of coordination policies for services.

Experience: To illustrate some of the features and design facilities made available by the SC-JSCL framework, we consider the Car repair scenario of the Automotive case study.

The focus of our experiments is on the design and implementation of the workflow of the coordination, taking into account the possibility of handling long-running transactions (LRTs) in the style of SAGA compensations. The results of our experiments are presented in [CFGS08, FGST08a, FGST08b].

Benefits: The main benefits of the approach can be summarized as follows

- *The integration of semantic-based verification techniques to manage properties of LRTs*. Our verification technique is based on the notion of *bisimulation*, an observational equivalence, that not only represents the behaviour of sets of components interacting with each other but also that of isolated subsystems. Bisimilarity allows us to distinguish isolated components that behave differently when 'plugged' into different service networks. The Car repair scenario has been designed and implemented by considering the transactional requirements given at specification level via the BPMN notation. We formally proved that the constraints on transactional isolation are maintained in the involved components. The verification of the Car repair scenario is done by checking that it is bisimilar to a 'magic' property, i.e. an abstracted design that models properties of interest.

- *BPMN and LRTs*. The SC-JSCL environment allows the design of LRTs exploiting the BPMN design notation. This has the main benefits that designers can specify LRTs in the familiar and semi-formal BPMN while leaving their precise specification to the underlying formal machineries. Moreover, the JSCL design can be automatically compiled into executable Java programs.

Feedback: The application of the methodology to the Car repair scenario has allowed us to develop a *model-driven development* methodology and certain *refactoring* for LRTs. Indeed, as a matter of fact, BPMN designs (i) neglect distribution aspects of the transactional activities, (ii) does not specify if activities are atomic or consisting of hidden sub-activities, (iii) delegate activities or compensations.

Our refactoring rules address these crucial issues of the deployment phase. Arguably, refactoring does not have to alter the high level meaning of the designer since refactoring rules preserve the intended semantics. Our refactoring rules are proved sound by showing that they preserve (weak) bisimularity.

5 Relational Analysis for Delivery of Services

Aim: Process algebras facilitate abstract models of a number of features of concurrent and distributed computation. Many use the notion of channel to provide end-to-end guarantees ensuring secure communications taking place. It is then a requirement on the transporting processes that they correctly implement the intended end-to-end communication. Our aim is to show how to use static analysis for demonstrating that the service requests of the system are correctly distributed by the multiplexer process.

Experience: In the Road accident scenario of the Automotive case study, all messages between the car and the service centre are communicated over a multiplexed, wireless channel. The multiplexer takes care of distributing messages correctly. It is necessary to find a way to formally validate that the multiplexer does the job correctly. We developed in [NNB$^+$08] a relational analysis that can be easily extended to new, emerging global computing calculi. The analysis is implemented and provides results that are sufficiently precise to validate the Road accident scenario.

Benefits: The core benefit of a relational analysis is that the dependency among the various message fields is taken into account, which allows one to validate the correct delivery of services in the case study.

Feedback: The application of our technique to the case study suggests some possibilities for future work. For one, we may investigate the feasibility of developing a mixed independent-attribute and relational analysis. We may also incorporate techniques that can tell distinct recursive instances, e.g. several cars having an accident at the same time, apart, i.e., allow the analysis to find out that the SOS messages contain the correct position of each car.

6 Relational Analysis of Correlation

Aim: In service-oriented computing, *message correlation* is a mechanism for ensuring that subsequent messages will be treated as a single ongoing conversation between interacting parties. Correlations thus allow e.g. for implementation of sessions or long-running transactions.

As an example, the Road accident scenario from the Automotive case study is used. This scenario describes an accident service to which a number of cars subscribe. When in-car sensors in one of subscriber vehicles detect abnormal conditions the service center is notified of the identity and the GPS coordinates of the vehicle. The service then asks the driver to confirm or to revoke the alarm. If the driver confirms, or fails to respond, an ambulance is called to the place of the accident. An obvious difficulty is that the service center must be able to handle possibly many concurrent service sessions without undesired interferences. More specific behavioural properties of the scenario one may want to guarantee are:

- The ambulance is not called if the driver confirms that she is well.
- Messages sent to the ambulance always contain GPS information.
- If several cars employ the accident service at once, then only the positions of the ones confirming the accident are reported to the ambulance.

Experience: The study of process calculi is a challenging avenue for the development and application of static analysis. Such analyses often tend to fall between two extremes: rather simple 0-CFA analyses, where no context information is taken into account, expressing only rudimentary properties, or extremely powerful relational and polyhedral analyses. We consider it an important achievement of our work, in [BNNP08], that we populate the middle ground between the two extremes. This approach can be specified almost as naturally as simpler independent attribute analyses, that is easily extendable to new, emerging global computing calculi. The analysis is implemented and provides results that are sufficiently precise to validate the Road accident scenario.

Benefits and Feedback: To the best of our knowledge, [BNNP08] is the first paper presenting a development of static analysis for a COWS-like calculus (more precisely, for a calculus similar to Fusion Calculus). Also, it seems to be the first static analysis of correlation.

7 A Flow-Sensitive Analysis of Privacy Properties

Aim: In this work we consider service-oriented architectures where many components interact with one another using a wireless network. We are interested in questions like:

- Can I be sure that I do not get unsolicited information from some service? — unless I give my permission?
- Can I be sure that information I send to some service never is leaked to another service? — unless I give my permission?

We then developed a static program analysis and show how is can be used to give privacy guarantees, like the ones requested above, to the Automotive case study.

Experience: In recent years, static program analyses techniques have been developed for analysing security properties for systems expressed in process calculi. The focus has been on analysing the configurations of the systems and control flow analyses techniques have played a major role. They have been flow-insensitive as well as context-insensititive and thus, from a static analysis point of view, the techniques have been fairly simple. Nonetheless, from a security point of view, surprisingly powerful results have been obtained. It is characteristic for these analyses that they produce a single abstract configuration approximating all the concrete configurations of a given system. The security property of interest has then been validated with respect to this abstract configuration: if the abstract configuration could not exhibit the unwanted behaviour, then none of the concrete configurations could exhibit it either. However, there are cases where this is not enough. To provide guarantees like the ones hinted above as aim, it seems that several abstract configurations are needed in order to distinguish between whether or not certain actions have been performed; also, it seems that we would need information about the order in which these configurations might be encountered when the system is running.

We develop in [NN07] such an analysis for the π-calculus: given a process it will construct a finite automaton, where the states abstract the potential configurations of the system, and the transitions of the automaton faithfully reflect the semantics of the process. We show how our analysis can be used to validate privacy properties.

Benefits: The analysis presented is flow sensitive as well as context sensitive. Applying the technique to the Automotive case study has allowed us to validate several interesting properties related to privacy:

- Sometimes it must be ensured that certain messages are only received after a certain login activity has taken place.
- Sometimes it must be ensured that messages only reach the intended recipients.
- Sometimes it is allowed to "declassify" information for a certain purpose.

As shown in [NN07], it is rather straightforward to validate the properties by inspecting the transitions of the automaton as well as information associated with the states.

Feedback: The analysis technique developed in the work is powerful enough to provide the required guarantees; Further work is needed to develop it for the hardware programming language VHDL, in which the original system was developed.

8 Logic-Based Conflict Detection for Distributed Policies

Aim: Policies are used to describe rules that are employed to modify (often distributed) system behaviour at run-time. Typically policies are created by many different people and added to the system at different times. This leads naturally to inconsistency between the policies, a problem that has been recognised and termed *policy conflict*. In [MRS07, GMRS07a] we present a novel formal semantics for distributed policies expressed in the APPEL language. The semantics is given in terms of formulae in ΔDSTL(x), an extension of temporal logic to deal with global applications. The work includes a semantics-based technique to detect policy conflicts and a consideration of conflict resolution. The aim of applying our technique to the Automotive case study was to validate it in the distributed case.

Experience: In APPEL a policy consists of a number of policy rules. Each of these consists of an optional trigger, an optional condition, and an action. The applicability of a rule depends on whether its trigger has occurred and whether its conditions are satisfied. The applicability of a policy depends on the applicability of the composing policies and on the grouping operators used (**seq**uential, **par**allel, **g**uarded and **u**nguarded choice). Juxtaposition of policies is also a policy, with the same semantics of parallel composition. The APPEL language is one of the ingredients of StPowla, a workflow-based approach to business process modelling [GMRS07b, MvB$^+$08].

As an example we consider the Car repair scenario from the Automotive case study. Assume a car equipped with a diagnostic hardware/software system that continuously reports on the status of the vehicle. When the car experiences some major failure (e.g. engine overheating, exhausted battery, flat tires) an embedded service is invoked to provide the user with a tow truck that carries his damaged car to a garage for repair. A potential conflict can be detected in this scenario.

P_1 : If a car fault happens, then tow truck and garage services are automatically selected.

Policy P_1 is defined by the car manufacturer, and it is deployed in the diagnostic system. This scenario can be modified to take care of user needs. We introduce a new policy to accommodate the fact that usually a driver already knows and trusts some of the garage and towing truck services of his own town.

P_2 : If a car fault happens in the driver's town, then he wants to select the services to be used.

Note also that this is a typical situation where it is not feasible to operate at design-time: the manufacturer policy is already deployed in the system; the user one is added later. We state these policies in APPEL:

$$P_1 = \texttt{when car\_accident do autom\_select\_serv}$$
$$P_2 = \texttt{appliesTo driverTown when car\_accident do manually\_select\_serv}$$

and derive their semantics. We need to add a location variable to the unlocated policy P_1, which for clarity we call AnyTown:

$$\mathcal{G}[\![P_1]\!] = \mathsf{AnyTown}\ \Delta car\_accident\ \text{LEADS\_TO}\ \mathsf{AnyTown}\ autom\_select\_serv$$
$$\mathcal{G}[\![P_2]\!] = \mathsf{driverTown}\ \Delta car\_accident\ \text{LEADS\_TO}\ \mathsf{driverTown}\ manually\_select\_serv$$

According to ΔDSTL(x) semantics, in $\mathcal{G}[\![P_1]\!]$ there is an implicit universal quantification on AnyTown. A legal instance of $\mathcal{G}[\![P_1]\!]$ is when location driverTown binds the location variable AnyTown. Then in two steps we derive that in $driverTown$ the services are selected automatically and manually at the same time, leading to a conflict.

As discussed in [GMRS07a], a strategy to reduce conflicts when a new policy is introduced is to define its relation with other policies applying to the same system. But this might be boring. To reduce the user work to essentials, he can first simply justappose the policies, then use the conflict detection mechanism and define relations only to solve the detected conflicts. Usually, he should assign a priority

to the most recently introduced policy. In APPEL this can be done with the operator **seq** that implements a priority: in P_1 **seq** P_2 we determine whether P_1 is applicable; if so we apply it, otherwise we check P_2.

Back to the car repair example, we note that policy P_2 has a natural priority with respect to the default policy P_1. We thus compose them in the sequence P_2 **seq** P_1, where $\mathcal{G}[\![P_2 \textbf{ seq } P_1]\!] =$

$$\mathsf{driverTown}\ \Delta car\_accident \ \text{LEADS\_TO}\ \mathsf{driverTown}\ manually\_select\_serv$$
$$\mathsf{AnyTown}\ \Delta car\_accident \wedge \sim \mathsf{driverTown}\ true \ \text{LEADS\_TO}\ \mathsf{AnyTown}\ autom\_select\_serv$$

Now, in locality n the first policy does not apply, while an instance of the second formula does. In location driverTown the first policy can apply while the second cannot, thus avoiding the conflict.

Benefits: What makes policy-driven systems appealing, is that the policies can be updated by the user, which is not an IT expert. Moreover, they can be updated without interrupting the System operation, thus providing the user with great adaptability. In the Car repair scenario, the user may, for instance, want to add its own policies after buying the car. However, when policies are added by an end-user, possibly in different time instants, the problem of policy conflict is significantly increased.

Feedback: The case study introduces distribution issues (a car is a moving entity) and distinguished policies might be applied in distinguished locations. We thus need to consider the situation in which a policy can be applied only to part of the system, rather than to the system as a whole. The need for such an extension arises when the system runs on physically distributed machines, and it is necessary/convenient to differentiate the policies for the various machines. Additionally, it is often convenient/necessary to distinguish the roles of the humans interacting with the system as parts to which different policies apply.

To address the distribution of policies we have: borrowed the core concept of *location* from the works on mobility; included localisation of policies, something that APPEL naturally provides through its *appliesTo* notation, and in our recent work extended the semantics of APPEL which was initially defined only for the non-distributed fragment.

9 Modelling the Credit Request Scenario in CaSPiS

Aim: We have applied the type-based techniques developed in [AB08a] to the Credit request scenario of the Finance case study. Our aim was to test the effectiveness of our technique against a concrete and nontrivial example. More specifically, we were interested in: (a) modelling rigorously the case study in CaSPiS, a service-oriented calculus introduced in [BBDL08]; (b) proving analytically a non-trivial property of the system modelled in this way, namely *client progress*. Basically, client progress states that any client invoking a service will never get stuck because of communication mismatches between any two participants involved in the process of serving its request.

Experience: We have modelled the Credit request scenario in CaSPiS$^-$, a sub-calculus of CaSPiS, in order to apply the type-based analysis described in [AB08a]. Some details of the modelling process are given in [AB]. CaSPiS (*Calculus of Sessions and Pipelines* [BBDL08]) is centered around the notions of *session* and of *pipeline*. A session is started when a service is invoked and contains the caller and callee's protocols running in parallel. Pipelines can be used for passing values across sessions. More sophisticated programming primitives can be encoded as combinations of these two. These primitives are viewed as natural tools for structuring client-service interaction and orchestration. Given the service-oriented nature of CaSPiS, we have first interpreted each actor/activity found in the case study's high level UML description as a separate service. Each of these services has then been modelled as CaSPiS process. The flow of activity among these processes is orchestrated by means of CaSPiS' wait-from and if-then-else constructs. The scenario can be viewed as a composition of all the identified services, plus a few clients. Internal computations, which are not relevant to this kind of analysis, have been

omitted. Some other details have been simplified to meet certain limitations of CaSPiS$^-$ (see below). The behaviour of the whole system has been abstracted by means of CCS types as described in [AB08a]. Well-typedness of the system has been checked, which ensures that interaction mismatches cannot occur. In particular, a client invoking the credit portal is guaranteed to eventually complete its own interaction protocol and receive a reply to its request as expected.

Benefits: Process calculi, being defined algebraically, are inherently compositional and, therefore, convey in a distilled form the paradigm at the heart of service-oriented computing. Generally speaking, on the scientific side, the use of formal and mathematically solid analysis techniques may lead to a deeper understanding of the considered system and of its properties. These considerations apply to our techniques as well. On the long run, one could also expect benefits on the engineering or even business sides, since the use of formal methods helps early detection of specification and implementation anomalies, or, dually, validation of desired properties, before actual deployment takes place. In this specific case, static analysis—more precisely a behavioural type system—is used to guarantee client progress, a property that can well be considered of paramount importance in a banking scenario.

Feedback: In the specification phase, some fine details of the Credit request scenario have left out, due to limitations of CaSPiS$^-$ and of the type system. In particular, the task lists, from which employees and supervisors are supposed to pick up the next task, were not implemented: the data to serve a credit request are passed to employees and supervisors directly at call time. A key point here is that services in CaSPiS$^-$ are *stateless*, which makes it is impossible to model a list as a service. This problem is intrinsic to the approach followed in the type system. These difficulties have recently motivated us into considering more sophisticated type-based techniques. Some preliminary results, relative to a somewhat simpler language (π-calculus), are reported in [AB08b].

10 A Methodology to Check Services' Correctness and Replaceability

Aim: In [BBCG07] we introduced a behavioural equivalence for OWL-S describing Web services represented by means of a novel kind of Petri nets, namely, OCPR nets. This equivalence establishes whether two service behaviours are equivalent, hence enabling the possibility of addressing prominent issues in Service-Oriented Computing, such as, e.g., the incremental development of services (i.e., to check whether two different versions of a service are equivalent), the publication of correct service specifications (i.e., to check whether a (complex) service implements a given specification), and the replaceability of services (i.e., to check whether a service s which takes part in a composition $C[s]$ can be replaced with a different service r without changing the behaviour of C, i.e., guaranteeing the equivalence between $C[s]$ and $C[r]$). These issues have been instantiated on the Finance case study, addressing the externalisation of the consumer rating service, and well as the presentation of a public specification that would not allow the consumer to be aware of the internal workings of the credit institute.

Experience: In [BBCG08a] we outlined a methodology addressing, in particular, the publication of correct service specifications and the replaceability of (sub)services, whose key ingredients are the decidable bisimulation equivalence introduced in [BBCG07], a formal encoding from OWL-S to OCPR nets (introduced in [BBCG08b] and implemented in [BCI07]), and the definition of an hiding operator. Specifically, given (the OWL-S descriptions of) a service $S1$, its (verified correct) public specification and a service $S2$, the methodology is able to check whether replacing a subcomponent of $S1$ with $S2$ does not change the behaviour described in the specification. It thus translates the subcomponent of $S1$ and $S2$ into OCPR nets and it checks whether such nets are equivalent by closing those data places that do not occur in the public specification of $S1$. As an example, [BBCG08a] presents a concrete scenario inspired by the Finance case study. The scenario centres on the OWL-S description of the CreditPortal

web service, granting loans to bank customers. Briefly, CreditPortal implements three step: (1) authentication of the customer and upload of her/his personal data, (2) evaluation of the customer credit request, and (3) formulation of the loan offer. In the first step, after logging into the bank system, the customer has to upload information regarding balance and offered guarantees. In the second step, CreditPortal evaluates the customer reliability and computes a rating of the credit request. In the last step, CreditPortal either decides to grant the loan to the customer and to build an offer or it rejects the credit request.

Benefits: Such an OWL-S description however describes the full behaviour of CreditPortal, while it is reasonable that the CreditPortal provider wants to publish a simpler specification of the service by hiding unnecessary and/or confidential details of its implementation. The methodology outlined in [BBCG08a] can be suitably employed to check whether the simpler CreditPortal specification respects the actual CreditPortal behaviour, i.e., whether CreditPortal implements its abstract specification. Moreover, the CreditPortal provider may want to enhance its service and hence decide to externalise the CreditPortal section which computes customer rating. Supposing that some services, named RatingOne and RatingTwo, which compute customer rating are available, then, the methodology can be employed to check whether the CreditPortal section to be externalised can be replaced with RatingOne or RatingTwo affecting neither the internal of CreditPortal nor the correctness of its public specification.

Feedback: The need of addressing the asymmetry of the matching of services (i.e., to check whether a (composition of) service(s) matches a query that specifies the behaviour of the desired service (composition) to be found) moved us to start discussing about *simulation*, thus taking into account the chance of satisfying query with an overspecified service.

11 A Model Checking Approach for the Analysis of SOC Applications

Aim: The emerging trend in industry is to use UML (state diagrams). We introduced the action/state-based branching-time temporal logic UCTL with the aim of using the full potential of specification languages (like UML) that allow both action and state changes to be modelled. Such logics have the advantage of more ease of expressiveness w.r.t. pure action- or state-based logics and their use often leads to a reduced state space, small memory needs, and less time spent for verification.

Experience: We defined UCTL as an extension of action-based CTL (ACTL); UCTL includes both CTL and ACTL. The service-oriented logic SocL used in Section 3 of this deliverable is a recent specialisation of UCTL meant to capture peculiar aspects of services. To allow the efficient verification of UCTL formulae, we developed the prototypical on-the-fly model checker UMC.

We showed the usefulness of our approach in [BFGM08] by applying it to a scenario from the Telecommunications case study. More precisely, in the design phase of aSOAP, an asynchronous extension of the web service communication protocol SOAP. Starting from a reference model of aSOAP provided by Telecom Italia, we described aSOAP as communicating UML state machines, expressed several behavioural properties on this UML model of aSOAP in UCTL, and verified them with UMC.

Telecom Italia envisions aSOAP to operate in a Client-Server architecture with an additional web service Proxy placed in between the Client and Server side. This Proxy must guarantee that various attempts to contact either side are made in case of temporary unavailability of the respective side.

We used UMC v3.3 on an ordinary PC to showed that, for instance, the property

Whenever a Client C reaches a deadlock, then C is either in state 'Wait' (for a Server response), or in state 'Deferred' (the Server has deferred the response)

can be shown to hold by verifying the action/state-based UCTL formula

$$AG\ (\ (\neg EF < C1 : > true) \Rightarrow (C1.state = Wait \vee C1.state = Deferred)\ ).$$

Benefits: Our experience demonstrated that formally modelling and verifying a reference model for the aSOAP protocol provided by Telecom Italia may actually increase the confidence in the design. This use of formal methods in the design phase of an asynchronous extension of SOAP helps to eventually arrive at a proposal of which we can guarantee that it satisfies certain desirable properties.

Feedback: Due to the similarities of both goals (UML verification) and techniques (on-the-fly model checking) it would definitively be interesting to compare our model and results with an equivalent (using the textual UML format) model in the HUGO framework [KMR02]. One of the main differences between our approach and that of HUGO (apart from the real-time aspects, not considered by us) probably lies in the kind of logic used to specify the properties. The HUGO/SPIN approach is based on a state-based linear-time logic (LTL), while our approach relies on a action/state-based branching-time logic (UCTL).

12 Formal Verification of an Automotive Scenario in SOC

Aim: The general goal of our research is to develop formal reasoning mechanisms and analytical tools for checking whether the services resulting from a composition meet desirable correctness properties and do not manifest any unexpected behaviours. The aim of this specific experience is to verify *a priori*, thus before implementation, certain design issues related to functional requirements (like service responsiveness and reliability) of a scenario from the Automotive case study.

Experience: We considered a scenario that offers a car driver on road assistance to contact a garage, a tow truck and a rental car when stranded with a malfunctioning vehicle. To this aim, the requirements model of this scenario from the Automotive case study [KB07]—a high-level UML specification that makes use of domain-specific extensions [KMH+07] like stereotypes to deal with compensations—was formally defined as a set of communicating UML state machines. It is important to note that we implemented one of the many possible operational models that can be defined in accordance with the scenario's requirements model of [KB07]. In fact, we made a number of assumptions w.r.t. this model (for details see [BGKM08]). The on-the-fly model checker UMC was subsequently used to verify a set of correctness properties formalised in the action- and state-based temporal logic UCTL [BFGM08].

For instance, we considered a service to be *responsive* if it guarantees a response to each received request. An example of service responsiveness is expressed by the UCTL formula

$$AG\,[\,requestCardCharge\,]\;\; A\,[\,true\;\;{}_{true}U_{\,chargeResponseOK \vee chargeResponseFail}\;\; true\,],$$

which states that each time action `requestCardCharge` takes place, always at a certain moment action `chargeResponseOK` or `chargeResponseFail` takes place. More intuitively: If the `Car` requests the `Bank` to charge a credit card, then the `Bank` will surely reply with a notification of either a successful or a failed attempt to charge the credit card.

We verified the above formula with UMC v3.4 on an ordinary PC, which showed the formula holds. We verified a number of further behavioural properties (service coordination, service reliability, uniqueness of response, etc.) expressed in UCTL over our implementation of the On road assistance scenario.

Benefits: Our verification experience allowed us to conclude that the requirements model of the On road assistance scenario is well designed in [KB07] and, moreover, to show the usefulness and feasibility of a formal approach to specify and rigorously analyse a system design, also in industrial contexts.

Feedback: There is room for improvement of our model checker: regarding the UML support of the tool, regarding further optimisations of the on-the-fly model-checking algorithm, and regarding the overall usability and user-interface issues.

13 Formal Verification of Secure Service Contracts

Aim: Our starting point is the foundational calculus for service orchestration, called λ_{req} [BDF06a, BDF06b, BDF08], with primitive constructs to describe, select and compose services. Services are modelled as functions and service descriptions as a suitable types and effect system. Our type and effects for services extend the standard WSDL notion of published interfaces. Besides the standard WSDL attributes, our effects add semantic information about service behaviour. This additional attribute formally is a context-free grammar that over-approximates the run-time behaviour of services. The semantic extension of service description impacts on service discovery and selection. The service selection mechanism demands satisfaction of a *secure service contract*: the abstraction of service behaviour must satisfy the security policy specified by the contract. Operationally, a service selection results in a sort of *call-by-contract*: the service repository is searched for a service with a functional type matching the request type. Moreover, the corresponding behavioural abstraction H must respect the secure service contract specified in the request. The notion of call-by-contract calls for a matchmaking algorithm based on formal semantic abstractions. To this aim, we have developed a static analysis technique to detect the viable strategies to solve the call-by-contracts involved in a service orchestration. Our static machinery exploits model checking techniques to determine the orchestration plans driving the selection of those services that match the security contracts on demand. A plan formalises how a call-by-contract invocation is resolved into the actual service binding. In other words, our planning strategy is a semantic-based method to identify which services the orchestrator must choose to guarantee security of the overall application.

We aim to present and validate a design methodology that fully integrates *secure service contracts*. With such contracts we mean safety properties expressing regular properties of service behaviours, thus allowing enforcement of history-based security policies. We argue that *awareness* of security contracts in the early stages of software development is crucial to guarantee and certify higher security levels. Indeed, formal verification of security contracts allows designers to evaluate the impact of such contracts on the overall software architecture without committing to specific low level technological details.

Experience: To illustrate some of the features and design facilities made available by our framework, we consider the Car repair scenario of the Automotive case study. The main focus is not on the structure of the overall system architecture, but rather on the design of the workflow of the service orchestration, taking into account the specific driver policies and the security service contracts on demand. The design methodology and its application to the Car repair scenario are presented in [BDFZ07, BDFZ08].

Benefits: The proposed methodology features dynamic and static semantics, allowing for formal reasoning about systems. Static analysis and model checking techniques provide the designer with useful information to assess and fix possible vulnerabilities. The main benefits of the approach are as follows:

- *The integration of a verification technique to study the composition properties of service networks*. A static analysis is used to infer an abstraction of the behaviour of a network. This abstraction is then model-checked to construct a correct orchestrator that coordinates the running services in a secure manner. Secure orchestration will also allow for improving the overall performance by avoiding unnecessary dynamic security checks while executing services. Studying the output of the model checker may highlight possible design flaws and suggest how the calls by contract and the security policies can be revised. All of the above machinery is completely mechanizable and we implement a tool to support our methodology. The fact that the tool is based on strong theoretical grounds positively impacts the reliability of our approach.

- *A study of various planning and recovering strategies*. We discuss several situations in which one needs to take a decision before proceeding with the execution. For instance, when a planned service disappears unexpectedly, one can choose to replan, i.e. to adapt the current plan to the new network configuration. Depending on the boundary conditions and on past experience, one can choose among different tactics. We discuss the feasibility, advantages, and costs of each of them.

Feedback: The application of the methodology to the Car repair scenario allowed us to properly identify and characterize a taxonomy of aspects for web service orchestration. See [BDFZ08] for a detailed description of the taxonomy. Moreover, the problem of choosing the appropriate services for a block of requests prompted us to understand which are the strategies for constructing or repairing a plan. Although one might defer service selection as much as possible, thus only performing it when executing a request, it is usually advantageous to decide how requests can be resolved in advance, i.e. to build a plan. This is because 'early planning' can provide better guarantees than late service selection. For instance, consider a block with two consecutive requests: r_1 and r_2. It might be that if we choose to resolve r_1 with a particular service s_1, later on we will not be able to find safe choices for r_2. In this case, we get stuck and we must somehow escape from this dead end, possibly with some expensive compensation (e.g., cancelling a reservation). Early planning, instead, can spot this kind of problem and try finding a better way, typically by also considering r_2 when taking a choice for r_1. We identified some strategies to construct or repair a plan. As a matter of fact, no strategy is always better than the others since all have advantages and disadvantages. The choice of a given strategy depends on many factors, some of which lie outside our formal model (e.g., availability of services, cost of dynamic checking, etc.).

14 Specification of a Requirement for an Automotive Scenario with MoSL

Aim: The aim was to validate the continuous stochastic logics for distribution and mobility for the logical characterisation of non-functional requirements of complex systems like, e.g., performance and dependability ones. In particular, the *Mo*bile *S*tochastic *L*ogic, MoSL [DKL$^+$07b], developed in the context of the SENSORIA project, has been used.

Experience: MoSL has been used for the specification of a responsiveness requirement for the Airbag scenario. The informal, global requirement which has been taken into consideration is the following.

> When an accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys, the following happens. First, an automated message is sent to the Accident Assistance Server which contains the vehicle's GPS data, the vehicle identification number and a collection of sensor data. Then, the Accident Assistance Server places a call to the driver's mobile phone. If, due to his injuries, the driver is unable to answer the call, and the severity of the accident is confirmed also by the sensor data, the emergency services are alerted and the vehicle location is communicated to them. Incoming car are also alerted.

The following methodology has been adopted: the requirement is first decomposed into simpler ones; then, each of them is made more precise by making (1) time bounds explicit and (2) statements realistic using probabilistic information. Each of these more concrete and realistic requirements is then formalised as a MoSL formula, where time-bounded until $\mathcal{U}$ is used to deal with time bounds while the probabilistic path operator $\mathcal{P}$ takes care of probabilistic issues. As we will see in the example below, a typical MoSL time bounded until formula is of the form $F1\ \mathcal{U}^{<t}_{site:act}\ F2$. Informally, the intended meaning is that a given run r of the system satisfies the formula if and only if a state s satisfying *F2* can be reached in r, from the initial state of r, by time t via states all satisfying *F1*; moreover, the action by which s is entered must be *act* and such an action must be executed at address *site*. Then, the meaning of the probabilistic formula $\mathcal{P}_{>p}(F1\ \mathcal{U}^{<t}_{site:act}\ F2)$ is that the probability of the set of such runs must be larger than p.

We show the above sketched process by applying it to one of the (sub-)requirements of the responsiveness requirement. Suppose it has been detected that the driver is seriously injured, according to the criteria described above. In this case the Accident Assistance Server is supposed to send, by means of a proper output action, an alert to the Emergency Service. In order for the rescue operations to be effective, ideally we would need to ensure that the Emergency Service is *always* alerted within a certain maximal time, say t_{alert}. In practice, we accept that the alert is sent by time t_{alert} in more than 99.7% of the cases.

Under the assumption that the variables car_{id} and gps are bound to the relevant car identifier and GPS data, the above requirement can be formalised as follows:

$$\mathcal{P}_{>0.997}(true\ \mathcal{U}^{<t_{alert}}_{AAS:OUT(car_{id},gps)@ES}\ true),$$

where *AAS* stands for the (address of) the Accident Assistance Server and *ES* stands for (the address of) the Emergency Server. The notation $AAS:OUT(car_{id}, gps)@ES$ means that an output action sending the pair (car_{id}, gps) to address *ES* is executed at address *AAS*. No automatic verification has been performed w.r.t. the Automotive case study because no detailed system model was available. A model-checking algorithm has been developed for (an extension of) MoSL that is described in [DKL$^+$07b] and is the subject of SENSORIA deliverable D4.2b.

Benefits: To assess performance and dependability issues, typically long-run or transient probabilities of Continuous Time Markov Chains (CTMCs) are determined; we showed how performance and dependability guarantees can be formally specified using MoSL; this allows the use of recent techniques to determine performance and dependability guarantees in a fully automated manner using model checking.

A clear advantage of the approach to performance and dependability assessment mentioned above is the completely *formal*, actually logical, characterisation of the performance and dependability measures of interest. Informal descriptions of complex measures could in fact be easily misinterpreted and are much more likely to be affected by mistakes. Moreover, with the help of a stochastic model-checker, one can automatically check whether a performance or dependability requirement is indeed fulfilled by a specific system model. It is important to point out that the model-checking tool not only provides a *yes/no* answer, but it provides *also* the actual values of the probabilities of interest. In this sense, stochastic model-checkers include *also* the functionality of traditional CTMCs analysis tools, but such a functionality is embedded in the formal framework which we have already mentioned above.

A second advantage comes from the fact that, in many cases, correctness, and in general functional properties of behaviour, usually captured by temporal logics like CTL, can be characterised by formulae of *stochastic* temporal logics, where the degenerate probability values 0 and 1 are used. This means that we can formulate and automatically check *both* functional *and* non-functional properties of system behaviour in an *integrated* way, within the *same* formalism.

Feedback: The logic proved itself well suited for the formalisation of the requirements of interest, as they were expressed informally in the Automotive case study.

15 Safety and Response-Time Analyses of an Automotive Scenario

Aim: In [CG07, ACF$^+$08] we have used our PEPA related software to analyse the Accident assistance scenario of the Automotive case study. Our goal was to increase the functionality of our software tools in order to enable the sensitivity analysis of rates associated with certain activities performed within this scenario. To this end [CG07] describes work which allowed us to solve a PEPA model of the scenario described in the case study a great number of times varying the rates involved. This was achieved through use of a distributing computing platform. Work done for [ACF$^+$08] integrated our solution techniques with the SENSORIA Development Environment and in addition allowed communication with work done within other SENSORIA WPs in the safety analysis of the same example scenario.

Experience: Because of the complementary work done in [CG07] we were well prepared for the problems presented in the given scenario. The largest difficulty in modelling such a system is determining suitable rates for the incoming requests—that is actions which initiate a service invocation. In the given scenario this uncertainty was present in two locations, the first is the number of crash reports coming into the service provider and the second is the percentage of these which are true emergencies and which

are false alarms. The first area of uncertainty was of less concern to us since we are modelling only the realm of responsibility of the service provider beginning after the service is invoked. The second area of uncertainty is a part of that which we are analysing through the use of sensitivity analysis. By multiplying the rate which an unhurt driver answers the phone by a probability we can vary that probability as though it were a rate. This means that we can then vary the probability over a range of values obtaining results for best, expected and worst case scenarios. This we must do because the service is not yet in operation and hence obtaining actual data for such probabilities is awkward.

Benefits: Rates within a passage often have complicated relationships. In our scenario it seems that increasing any one rate will increase the probability of completing at or before a given time. However in many models this is not the case and increasing some rates may decrease performance. Indeed sometimes it is the ratio between rates which must be either by high or low to increase performance. In our case, if we increase the probability that the driver does not answer the phone and thereby decrease the rate at which the driver may answer the phone then we actually reduce the probability of completing the passage within a given time. These, often complex, relationships between the rates involved mean that one cannot simply solve the model a few times representing the best, expected and worst case estimates for all involved rates. The benefit in using our technique in parameter sweeping over all the unknown/variable rates is that every configuration is analysed and it is therefore not possible to miss corner cases representing particularly bad (or good) rate configurations that are not obvious from the original model.

The benefit gained from carrying out precise response time analysis based on solid mathematical methods is that it allows service providers to make well-defended statements about the quality of service which they deliver. Without supporting formal analysis any such claims risk over-estimating the speed of response from the service and claiming too much, or under-estimating the speed of response and claiming too little. The former is especially bad because it is potentially litigious, risking exposure to punitive lawsuits. The latter is also undesirable because it could lead to loss of revenue when customers are discouraged by lax guarantees of quality.

Feedback: The application of our PEPA modelling software to the given case study and the appropriate integration with not only the SENSORIA Development Environment but the LTSA safety analyser lead to a more user friendly PEPA development environment. In addition we have made some small modifications to the input language which were invaluable in relating the two models of the same scenario. In particular the use of free strings as a component identifier which we speculate would be a valuable addition to many other modellling languages.

16 Conclusion

This deliverable provides a detailed overview of the applications of techniques, methods and languages developed in WP3 and WP4 to scenarios of the SENSORIA case studies from WP8. As such it gives a good idea of the usefullness of these techniques, methods and languages in the SOC domain. Due to length restrictions, neither the case studies nor the particular techniques, methods and languages are described in full detail in this deliverable. The interested reader is referred to the appropriate deliverables related to WP3, WP4 and WP8 for such details.

17 Relevant SENSORIA Publications and Reports

This deliverable is based on the following publications and reports from SENSORIA:

[AB08a] Lucia Acciai and Michele Boreale. A type system for client progress in a service-oriented calculus. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 642–658. Springer, 2008.

[ACF+08] Ashok Argent-Katwala, Allan Clark, Howard Foster, Stephen Gilmore, Philip Mayer, and Mirco Tribastone. Safety and Response-Time Analysis of an Automotive Accident Assistance Service. In *Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'08), Porto Sani, Greece*, volume 17 of *Communications in Computer and Information Science (CCIS)*. Springer, 2008. To appear.

[BDFZ08] Massimo Bartoletti, Pierpaolo Degano, Gianluigi Ferrari, and Roberto Zunino. Semantics-Based Design for Secure Web Services. *IEEE Transactions on Software Engineering*, 34(1):33–49, 2008.

[BNNP08] Jörg Bauer, Flemming Nielson, Hanne Riis Nielson, and Henrik Pilegaard. Relational Analysis of Correlation. In María Alpuente and Germán Vidal, editors, *Proceedings of the 15th International Static Analysis Symposium (SAS'08), Valencia, Spain*, volume 5079 of *LNCS*, pages 32–46. Springer, 2008.

[BFGM08] Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, and Franco Mazzanti. An Action/State-Based Model-Checking Approach for the Analysis of Communication Protocols for Service-Oriented Applications. In Stefan Leue and Pedro Merino, editors, *Formal Methods for Industrial Critical Systems—Revised Selected Papers of the 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'07), Berlin, Germany*, volume 4916 of *LNCS*, pages 133–148. Springer, 2008.

[BGKM08] Maurice H. ter Beek, Stefania Gnesi, Nora Koch, and Franco Mazzanti. Formal Verification of an Automotive Scenario in Service-Oriented Computing. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany*, pages 613–622. ACM, 2008.

[BBCG08a] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. Compositional Specification of Web Services via Behavioural Equivalence: A Case Study. In Kees M. van Hee and Rüdiger Valk, editors, *Proceedings of the 29th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN'08), Xi'an, China*, volume 5062 of *LNCS*, pages 52–71. Springer, 2008.

[CFGS08] Vincenzo Ciancia, Gianluigi Ferrari, Roberto Guanciale, and Daniele Strollo. Checking Correctness of Transactional Behaviors. In Kenji Suzuki, Teruo Higashino, Keiichi Yasumoto, and Khaled El-Fakih, editors, *Proceedings of the 28th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'08), Tokyo, Japan*, volume 5048 of *LNCS*, pages 134–148. Springer, 2008.

[DKL+07] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. D4.2a: Stochastic logics. SENSORIA Deliverable 4.2a, Feb 2007.

[FGL+08] Alessandro Fantechi, Stefania Gnesi, Alessandro Lapadula, Franco Mazzanti, Rosario Pugliese, and Francesco Tiezzi. A model checking approach for verifying COWS specifications. In José Luiz Fiadeiro and Paola Inverardi, editors, *Proceedings of the 11th International Conference on Fundamental Approaches to Software Engineering (FASE'08), Budapest, Hungary*, volume 4961 of *LNCS*, pages 230–245. Springer, 2008.

[FGST08a] Gianluigi Ferrari, Roberto Guanciale, Daniele Strollo, and Emilio Tuosto. Event Based Service Coordination. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 312–329. Springer, 2008.

[FGST08b] Gianluigi Ferrari, Roberto Guanciale, Daniele Strollo, and Emilio Tuosto. Refactoring Long Running Transactions. In Roberto Bruni and Karsten Wolf, editors, *Proceedings of the 5th*

International Workshop on Web Services and Formal Methods (WSFM'08), Milan, Italy, LNCS. Springer, 2008. To appear.

[LPT08] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. Specifying and Analysing SOC Applications with COWS. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 701–720. Springer, 2008.

[MRS08] Carlo Montangero, Stephan Reiff-Marganiec, and Laura Semini. Logic-based conflict detection for distributed policies. 2008. Submitted.

[NN07] Flemming Nielson and Hanne Riis Nielson. A flow-sensitive analysis of privacy properties. In *Proceedings of the 20th Computer Security Foundations Symposium (CSF'07), Venice, Italy*, pages 249–264. IEEE, 2007.

[NNB+08] Flemming Nielson, Hanne Riis Nielson, Jörg Bauer, Christoffer Rosenkilde Nielsen, and Henrik Pilegaard. Relational Analysis for Delivery of Services. In Gilles Barthe and Cédric Fournet, editors, *Proceedings of the 3rd Trustworthy Global Computing Symposium (TGC'07), Sophia-Antipolis, France*, volume 4912 of *LNCS*, pages 73–89. Springer, 2008.

[YNNÖ07] Ender Yüksel, Hanne Riis Nielson, Christoffer Rosenkilde Nielsen, and Mehmet Bulent Örencik. A secure simplification of the PKMv2 protocol in IEEE 802.16e-2005. In Pierpaolo Degano, Ralf Küsters, Luca Viganò, and Steve Zdancewic, editors, *Informal proceedings of the Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07), Wrocław, Poland*, pages 149–164, 2007.

References

[AB] Lucia Acciai and Michele Boreale. Modeling the Financial Case Study in Caspis. Presentation available at `http://www.pst.ifi.lmu.de:8080/Sensoria/DOWNLOAD/WP2%2FTalk_Pi08%2FWP2-Case-Study-Acciai.pdf`.

[AB08a] Lucia Acciai and Michele Boreale. A type system for client progress in a service-oriented calculus. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 642–658. Springer, 2008.

[AB08b] Lucia Acciai and Michele Boreale. Spatial and behavioral types in the pi-calculus. In Franck van Breugel and Marsha Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08), Toronto, Canada*, volume 5201 of *LNCS*, pages 372–386. Springer, 2008.

[ACF+08] Ashok Argent-Katwala, Allan Clark, Howard Foster, Stephen Gilmore, Philip Mayer, and Mirco Tribastone. Safety and Response-Time Analysis of an Automotive Accident Assistance Service. In *Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'08), Porto Sani, Greece*, volume 17 of *Communications in Computer and Information Science (CCIS)*. Springer, 2008. To appear.

[BBCG07] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. A behavioural congruence for Web Services. In Farhad Arbab and Marjan Sirjani, editors, *Proceedings of the International Symposium on Fundamentals of Software Engineering (FSEN'07), Tehran, Iran*, volume 4767 of *LNCS*, pages 240–256. Springer, 2007.

[BBCG08a] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. Compositional Specification of Web Services Via Behavioural Equivalence: A Case Study. In Kees M. van

Hee and Rüdiger Valk, editors, *Proceedings of the 29th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency (ICATPN'08), Xi'an, China*, volume 5062 of *LNCS*, pages 52–71. Springer, 2008.

[BBCG08b] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. On the use of behavioural equivalences for Web services development. 2008. Submitted.

[BBDL08] Michele Boreale, Roberto Bruni, Rocco De Nicola, and Michele Loreti. Sessions and Pipelines for Structured Service Programming. In Gilles Barthe and Frank S. de Boer, editors, *Proceedings of the 10th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08), Oslo, Norway*, volume 5051 of *LNCS*, pages 19–38. Springer, 2008.

[BCI07] Antonio Brogi, Sara Corfini, and Stefano Iardella. From OWL-S descriptions to Petri nets. In *Proceedings of the 3rd International Workshop on Engineering Service-Oriented Applications (WESOA'07), Wien, Austria*, 2007. To appear.

[BDF06a] Massimo Bartoletti, Pierpaolo Degano, and Gianluigi Ferrari. Security Issues in Service Composition. In Roberto Gorrieri and Heike Wehrheim, editors, *Proceedings of the 8th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'06), Bologna, Italy*, volume 4037 of *LNCS*, pages 1–16. Springer, 2006.

[BDF06b] Massimo Bartoletti, Pierpaolo Degano, and Gianluigi Ferrari. Types and Effects for Secure Service Orchestration. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop, (CSFW'06), Venice, Italy*, pages 57–69. IEEE, 2006.

[BDF08] Massimo Bartoletti, Pierpaolo Degano, and Gianluigi Ferrari. Secure Service Composition. *Journal of Computer Security*, 2008. To appear.

[BDFZ07] Massimo Bartoletti, Pierpaolo Degano, Gianluigi Ferrari, and Roberto Zunino. Secure Service Orchestration. In Alessandro Aldini and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design IV: FOSAD 2006/2007 Tutorial Lectures*, volume 4677 of *LNCS*, pages 24–74. Springer, 2007.

[BDFZ08] Massimo Bartoletti, Pierpaolo Degano, Gianluigi Ferrari, and Roberto Zunino. Semantics-Based Design for Secure Web Services. *IEEE Transactions on Software Engineering*, 34(1):33–49, 2008.

[BFGM08] Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, and Franco Mazzanti. An Action/State-Based Model-Checking Approach for the Analysis of Communication Protocols for Service-Oriented Applications. In Stefan Leue and Pedro Merino, editors, *Formal Methods for Industrial Critical Systems—Revised Selected Papers of the 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'07), Berlin, Germany*, volume 4916 of *LNCS*, pages 133–148. Springer, 2008.

[BFM+05] Roberto Bruni, Gianluigi Ferrari, Hernán Melgratti, Ugo Montanari, Daniele Strollo, and Emilio Tuosto. From Theory to Practice in Transactional Composition of Web Services. In Mario Bravetti, Leïla Kloul, and Gianluigi Zavattaro, editors, *Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WSFM'05), Versailles, France*, volume 3670 of *LNCS*, pages 272–286. Springer, 2005.

[BGKM08] Maurice H. ter Beek, Stefania Gnesi, Nora Koch, and Franco Mazzanti. Formal Verification of an Automotive Scenario in Service-Oriented Computing. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany*, pages 613–622. ACM, 2008.

[BNNP08] Jörg Bauer, Flemming Nielson, Hanne Riis Nielson, and Henrik Pilegaard. Relational Analysis of Correlation. In María Alpuente and Germán Vidal, editors, *Proceedings of the 15th International Static Analysis Symposium (SAS'08), Valencia, Spain*, volume 5079 of *LNCS*, pages 32–46. Springer, 2008.

[CFGS08] Vincenzo Ciancia, Gianluigi Ferrari, Roberto Guanciale, and Daniele Strollo. Checking Correctness of Transactional Behaviors. In Kenji Suzuki, Teruo Higashino, Keiichi Yasumoto, and Khaled El-Fakih, editors, *Proceedings of the 28th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'08), Tokyo, Japan*, volume 5048 of *LNCS*, pages 134–148. Springer, 2008.

[CG07] Allan Clark and Stephen Gilmore. Evaluating Quality of Service for Service Level Agreements. In Luboš Brim, Boudewijn R. Haverkort, Martin Leucker, and Jaco van de Pol, editors, *Proceedings of the 11th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'06) and of the 5th International Workshop on Parallel and Distributed Methods in verifiCation (PDMC'06), Bonn, Germany*, volume 4346 of *LNCS*, pages 181–194. Springer, 2007.

[CMC] CMC. An on-the-fly model checker and interpreter for COWS. Tool available through `http://fmt.isti.cnr.it/cmc/`.

[DKL+07a] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. D4.2a: Stochastic logics. SENSORIA Deliverable 4.2a, February 2007.

[DKL+07b] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. Model checking mobile stochastic logic. *Theoretical Computer Science*, 382(1):42–70, 2007.

[FGL+08] Alessandro Fantechi, Stefania Gnesi, Alessandro Lapadula, Franco Mazzanti, Rosario Pugliese, and Francesco Tiezzi. A model checking approach for verifying COWS specifications. In José Luiz Fiadeiro and Paola Inverardi, editors, *Proceedings of the 11th International Conference on Fundamental Approaches to Software Engineering (FASE'08), Budapest, Hungary*, volume 4961 of *LNCS*, pages 230–245. Springer, 2008. Full version available as technical report through `http://rap.dsi.unifi.it/cows`.

[FGS06a] Gianluigi Ferrari, Roberto Guanciale, and Daniele Strollo. Event Based Service Coordination over Dynamic and Heterogeneous Networks. In Asit Dan and Winfried Lamersdorf, editors, *Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC'06), Chicago, IL, USA*, volume 4294 of *LNCS*, pages 453–458. Springer, 2006.

[FGS06b] Gianluigi Ferrari, Roberto Guanciale, and Daniele Strollo. JSCL: A Middleware for Service Coordination. In Elie Najm, Jean-François Pradat-Peyre, and Véronique Donzeau-Gouge, editors, *Proceedings of the 26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06), Paris, France*, volume 4229 of *LNCS*, pages 46–60. Springer, 2006.

[FGST07] Gianluigi Ferrari, Roberto Guanciale, Daniele Strollo, and Emilio Tuosto. Coordination Via Types in an Event-Based Framework. In John Derrick and Jüri Vain, editors, *Proceedings of the 27th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'07), Tallinn, Estonia*, volume 4574 of *LNCS*, pages 66–80. Springer, 2007.

[FGST08a] Gianluigi Ferrari, Roberto Guanciale, Daniele Strollo, and Emilio Tuosto. Event Based Service Coordination. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 312–329. Springer, 2008.

[FGST08b] Gianluigi Ferrari, Roberto Guanciale, Daniele Strollo, and Emilio Tuosto. Refactoring Long Running Transactions. In Roberto Bruni and Karsten Wolf, editors, *Proceedings of the 5th International Workshop on Web Services and Formal Methods (WSFM'08), Milan, Italy*, LNCS. Springer, 2008. To appear.

[GMRS07a] Stephen Gorton, Carlo Montangero, Stephan Reiff-Marganiec, and Laura Semini. D1.3a: Prototype language for business process modelling— Primitives for business policies and end-user high-level goals. SENSORIA Deliverable 1.3a, March 2007.

[GMRS07b] Stephen Gorton, Carlo Montangero, Stephan Reiff-Marganiec, and Laura Semini. StPowla: SOA, Policies and Workflows. In *Proceedings of the 3rd International Workshop on Engineering Service-Oriented Applications: Analysis, Design, and Composition (WESOA'07), Vienna, Austria*, 2007. To appear.

[KB07] Nora Koch and Dominik Berndl. Requirements Modelling and Analysis of Selected Scenarios of the Automative Case Study. SENSORIA Deliverable 8.2a, September 2007.

[KMH+07] Nora Koch, Philip Mayer, Reiko Heckel, László Gönczy, and Carlo Montangero. UML for Service-Oriented Systems. SENSORIA Deliverable 1.4a, September 2007.

[KMR02] Alexander Knapp, Stephan Merz, and Christopher Rauh. Model Checking Timed UML State Machines and Collaborations. In Werner Damm and Ernst-Rüdiger Olderog, editors, *Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'02), Oldenburg, Germany*, volume 2469 of *LNCS*, pages 395–414. Springer, 2002.

[LPT07a] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. A Calculus for Orchestration of Web Services. In Rocco De Nicola, editor, *Proceedings of the 16th European Symposium on Programming (ESOP'07), Braga, Portugal*, volume 4421 of *LNCS*, pages 33–47. Springer, 2007.

[LPT07b] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. Regulating data exchange in service oriented applications. In Farhad Arbab and Marjan Sirjani, editors, *Proceedings of the International Symposium on Fundamentals of Software Engineering (FSEN'07), Tehran, Iran*, volume 4767 of *LNCS*, pages 223–239. Springer, 2007. Full version available as technical report through `http://rap.dsi.unifi.it/cows`.

[LPT08] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. Specifying and Analysing SOC Applications with COWS. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 701–720. Springer, 2008.

[MRS07] Carlo Montangero, Stephan Reiff-Marganiec, and Laura Semini. Logic-based detection of conflicts in APPEL policies. In Farhad Arbab and Marjan Sirjani, editors, *Proceedings of the International Symposium on Fundamentals of Software Engineering (FSEN'07), Tehran, Iran*, volume 4767 of *LNCS*, pages 257–271. Springer, 2007.

[MRS08] Carlo Montangero, Stephan Reiff-Marganiec, and Laura Semini. Logic-based conflict detection for distributed policies. 2008. Submitted.

[MvB+08] Carlo Montangero, M. Birna van Riemsdijk, Laura Bocchi, Stephan Reiff-Marganiec, and Laura Semini. D1.3b: Prototype language for business process modelling—Formal semantics and cases studies. SENSORIA Deliverable 1.3b, July 2008.

[NN07] Hanne Riis Nielson and Flemming Nielson. A flow-sensitive analysis of privacy properties. In *Proceedings of the 20th Computer Security Foundations Symposium (CSF'07), Venice, Italy*, pages 249–264. IEEE, 2007.

[NNB+08] Flemming Nielson, Hanne Riis Nielson, Jörg Bauer, Christoffer Rosenkilde Nielsen, and Henrik Pilegaard. Relational Analysis for Delivery of Services. In Gilles Barthe and Cédric Fournet, editors, *Proceedings of the 3rd Trustworthy Global Computing Symposium (TGC'07), Sophia-Antipolis, France*, volume 4912 of *LNCS*, pages 73–89. Springer, 2008.

[YNNÖ07] Ender Yüksel, Hanne Riis Nielson, Christoffer Rosenkilde Nielsen, and Mehmet Bulent Örencik. A secure simplification of the PKMv2 protocol in IEEE 802.16e-2005. In Pierpaolo Degano, Ralf Küsters, Luca Viganò, and Steve Zdancewic, editors, *Informal proceedings of the Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07), Wrocław, Poland*, pages 149–164, 2007.