Ralf H. Reussner
Judith A. Stafford
Clemens A. Szyperski (Eds.)

LNCS 3938

Architecting Systems with Trustworthy Components

International Seminar
Dagstuhl Castle, Germany, December 2004
Revised Selected Papers

5-53

Springer

TP311.5-53
A673
2004

Ralf H. Reussner Judith A. Stafford
Clemens A. Szyperski (Eds.)

Architecting Systems with Trustworthy Components

International Seminar
Dagstuhl Castle, Germany, December 12-17, 2004
Revised Selected Papers

Springer

Volume Editors

Ralf H. Reussner
Universität Karlsruhe (TH)
Fakultät für Informatik
Am Fasanengarten 5, 76131 Karlsruhe, Germany
E-mail: reussner@ipd.uka.de

Judith A. Stafford
Tufts University
Department of Computer Science
161 College Avenue, Medford, MA 02155, USA
E-mail: jas@cs.tufts.edu

Clemens A. Szyperski
Microsoft
One Microsoft Way, Redmond, WA 98053, USA
E-mail: cszypers@microsoft.com

Library of Congress Control Number: 2006905119

CR Subject Classification (1998): C.2.4, D.1.3, D.2, D.4.5, F.2.1-2, D.3, F.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-540-35800-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-35800-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11786160 06/3142 5 4 3 2 1 0

Preface

Software components are most generally viewed as a means of software re-use and, as such, much past research has been devoted to the study of problems associated with integrating components into cohesive systems. However, even when a collection of trustworthy components have been successfully assembled the quality of the resultant system is not guaranteed. In December 2004, 41 experts on this topic from around the world, from research as well as industrial organizations, came together at Dagstuhl to discuss pressing issues related to *architecting software systems from trustworthy components.*

During the course of the cold, yet sunny, December days in Dagstuhl, discussion sessions addressed topics such as compositional reasoning on various system-level properties (such as deadlocks, live-locks etc.), compositional prediction models for different quality attributes (such as performance or reliability), blame analysis, interaction protocols, and composition frameworks. Using the liberal form of Dagstuhl Seminars, the days of the seminar were filled mostly with discussion in a variety of settings: in working sessions, around the table at meals, small groups in a corner, and also all together in the main meeting room.

Component software technologies attract much attention for their promise to enable scaling of our software industry to new levels of flexibility, diversity, and cost efficiency. Yet, these hopes collide with the reality that assemblies typically suffer from the proverbial "weakest link" phenomenon. If a component is used in a new compositional variation, then it will likely be stressed in a new way. Asserting useful properties of assemblies based on the used composition schema and theory requires a firm handle on the properties of both the components being composed and the communication mechanisms (connectors) that bind them. For such assertions to hold, these composition elements must meet their advertized properties, even if used under circumstances not explicitly envisaged by their developers. A component or connector that fails to do so becomes a weak link of its hosting assembly and may cause the entire assembly to fail to meet its advertized properties.

In contrast, components that promise to be a strong link in their assemblies can be called 'trustworthy' and ways to get to the construction and proper use of such components was the subject of this seminar. Transitively, the seminar was also concerned with trustworthy assemblies: assemblies that reliably meet their requirements based on trustworthy components and solid composition methods.

The weakest link phenomenon is not a new observation, but the recent trends to move to dynamic and late composition of non-trivial components exasperate the problem. A concrete example promising deep widespread relevance are Web services. The problem space is complex and multi-faceted. Practical solutions will have to draw on combined insights from a diverse range of disciplines, including component software technology, software engineering, software architecture, dependable systems, formal methods, as well as areas such as type systems and proof-carrying code.

A lot of good, and sometimes even groundbreaking, work has been performed in the focus area of this seminar, but many problems remain open. To spark discussions, a small set of core problems was prepared by the organizers:

- Measurement and normalization of extra-functional properties
- Modular reasoning over extra-functional properties
- capture of component requirements in interfaces and protocols
- Interference and synergy of top-down and bottom-up aspects
- Duality of componentization and architecture
- System properties (non-deadlocks, liveness, fairness, etc.)
- Opportunities for correctness by construction/static checking

All of these problems are considered hard today and yet, all of them, if solved appropriately, promise the creation of key stepping stones toward an overall approach yielding trustworthy components as well as trustworthy compositions. It is likely that any such approach supports a multitude of more specialized disciplines and methods, targeting different requirement profiles at the assembly level; for example, those with tight resource management or that rely on real-time characteristics.

Most of the time at Dagstuhl was used for focused discussions in break-out groups; the abstracts of the break-out groups as well as position papers submitted by all participants are available on the seminar Website. In this volume of *Lecture Notes on Computer Science* we present extended papers reflecting work of seminar participants. Among the articles are ten peer-reviewed papers and five invited papers of outstanding researchers whose work is related to the Dagstuhl seminar but were not able to attend. The peer-reviewed papers were submitted by participants after the conclusion of the workshop and were selected based upon the high quality of scholarly work, their timeliness, and their appropriateness to the goals of the seminar, some reflecting ongoing collaboration that grew out of the seminar.

We would like to gratefully acknowledge the friendly and very helpful support of the Dagstuhl administration staff, Alfred Hofmann from Springer for his support during the preparation and publication of the LNCS volume and Klaus Krogmann for preparing the final manuscript for Springer.

Karlsruhe, Medford, and Redmond
February 2006

Ralf Reussner
Judith Stafford
Clemens Szyperski

Organization

Architecting Systems with Trustworthy Components

(Dagstuhl Seminar 04511)

Organizers

Ralf Reussner, Universität Karlsruhe (T.H.), Germany
Judith Stafford, Tufts University, USA
Clemens Szyperski, Microsoft Corp., USA

Participants

Uwe Aßmann, TU Dresden, Germany
Colin Atkinson, University of Mannheim, Germany
Steffen Becker, University of Oldenburg, Germany
Jan Bredereck, TZI, Bremen, Germany
Antonia Brogi, Università di Pisa, Italy
Christian Bunse, Fraunhofer IESE, Germany
Ivica Crnkovic, Mälardalen University, Sweden
Viktoria Firus, University of Oldenburg, Germany
Kathi Fisler, Worcester Polytechnic Institute, USA
Felix C. Freiling, RWTH Aachen, Germany
Sabine Glesner, Universität Karlsruhe (T.H.), Germany
Gerhard Goos, Universität Karlsruhe (T.H.), Germany
Ian Gorton, NICTA, Australia
Lars Grunske, The University of Queensland, Australia
Christine Hofmeister , Lehigh Univ. - Bethlehem, USA
Jens-Holger Jahnke, University of Victoria, Canada
Jean-Marc Jézéquel, IRISA (Univ. Rennes & INRIA), France
Bernd Krämer, FernUniversität in Hagen, Germany
Shriram Krishnamurthi, Brown Univ. - Providence, USA
Juliana Küster-Filipe, The University of Birmingham, UK
Stig Larsson, ABB - Västerås, Sweden
Nicole Levy, University of Versailles, France
Raffaela Mirandola, University of Rome TorVergata, Italy
Sven Overhage, Universität Augsburg, Germany
Frantisek Plasil, Charles University, Czech Republic
Iman Poernomo, King's College London, UK
Alexander Romanovsky, University of Newcastle, UK
Christian Salzmann, BMW Car IT, Germany
Thomas Santen, TU Berlin, Germany

Heinz Schmidt, Monash University, Australia
Jürgen Schneider, IBM - Böblingen, Germany
Asuman Sünbül, SAP Research Labs - Palo Alto, USA
Massimo Tivoli, Univ. degli Studi di L'Aquila, Italy
Kurt Wallnau, Software Engineering Institute, USA
Wolfgang Weck, Independent Software Architect, Switzerland
Rob van Ommering, Philips Research - Eindhoven, The Netherlands
Willem-Jan van den Heuvel, Tilburg University, The Netherlands

Invited Contributions

Antonia Bertolino, ISTI CNR Pisa, Italy
Manfred Broy, TU Munich, Germany
Bertrand Meyer, ETH Zurich, Switzerland
Wolfgang Pree, University of Salzburg, Austria
Heike Wehrheim, University of Paderborn, Germany

Publication date: May 2006

¥573.00元

Table of Contents

Audition of Web Services for Testing Conformance to Open Specified Protocols*

Antonia Bertolino[1], Lars Frantzen[2], Andrea Polini[1], and Jan Tretmans[2]

[1] Istituto di Scienza e Tecnologie della Informazione "Alessandro Faedo", Consiglio Nazionale delle Ricerche, via Moruzzi, 1 – 56124 Pisa, Italy
{antonia.bertolino, andrea.polini}@isti.cnr.it
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands
{lf, tretmans}@cs.ru.nl

Abstract. A Web Service (WS) is a type of component specifically conceived for distributed machine-to-machine interaction. Interoperability between WSs involves both data and messages exchanged and protocols of usage, and is pursued via the establishment of standard specifications to which service providers must conform. In previous work we have envisaged a framework for WS testing. Within this framework, this paper focuses on how the intended protocol of access for a standard service could be specified, and especially on how the conformance of a service instance to this specified protocol can then be tested. We propose to augment the WSDL description with a UML2.0 Protocol State Machine (PSM) diagram. The PSM is intended to express how, and under which conditions, the service provided by a component through its ports and interfaces can be accessed by a client. We then propose to translate the PSM to a Symbolic Transition System, to which existing formal testing theory and tools can be readily applied for conformance evaluation. A simple example illustrates the approach and highlights the peculiar challenges raised by WS conformance testing.

1 Introduction

Service Oriented Architecture (SOA) is the emerging paradigm for the realization of heterogeneous, distributed systems, obtained from the dynamic combination of remote applications owned and operated by distinct organizations. Today the Web Service Architecture (WSA) certainly constitutes the most relevant and widely adopted instance of such a paradigm.

A Web Service (WS) is essentially characterized by the capability to "support interoperable machine-to-machine interaction over a network" [5]. This capability is achieved due to the agreement of all major players on the usage of uniform

* This work has been supported by the European Project TELCERT (FP6 STREP 507128), by Marie Curie Network TAROT (MRTN-CT-2004-505121), and by the Netherlands Organization for Scientific Research (NWO) under project: STRESS – Systematic Testing of Realtime Embedded Software Systems.

WS interfaces, coded into the standard machine-processable Web Service Definition Language (WSDL) format [9], and of the Simple Object Access Protocol (SOAP) [18] for WS communication. Moreover, WSA interconnects service providers and service requesters via a standard Service Broker called the UDDI (Universal Description and Discovery Integration)[10]. The information in this catalog follows the yellow, green or white pages paradigms open technology, and defines a common mechanism to publish and retrieve information about available Web Services.

From a methodology viewpoint, WSA builds on the extensive framework of the Component-Based Software Development (CBSD) paradigm, of which it can be considered an attractive successor. Where in fact CBSD pursued the development of a composite system by the assembly of pre-existing (black-box) components, WSA chases the dynamic composition of services at client requests. The two paradigms share the underlying philosophy of developing building blocks (either components or services) of a system for external generalized reuse, whose implementation details are hidden behind a published interface.

By building on the extensive results of CBSD, WSs can today rely on a much more mature culture for compositional development, as testified by the emergence of established standard access and communication protocols. On the other hand, by exacerbating the aspects of loose coupling, distribution and dynamism, WSs have also inherited the most challenging issues of the component-based approach, directly descending here from the need of dynamically composing the interactions between services whose internal behavior is unknown. This fact brings several consequences on the trustability and reliability of WSA; in particular, it calls for new approaches to validate the behavior of black-box components whose services are invoked by heterogeneous clients in a variety of unforeseen contexts.

Although similar problems have been encountered and tackled in the area of software components, testing of WSs is even more difficult since the different machines participating in the interaction could be dispersed among different organizations, so even a simple monitoring strategy or the insertion of probes into the middleware is not generally feasible. Moreover, the notion of the WSA establishes rigid limitations on the kind of documentation that can be provided and used for integrating services. In particular, a service must not include information on how it has been implemented. This obviously is desirable to enable the decoupling between requesters and providers of services, but obviously makes integration testing more difficult.

Speaking in general, it is clear that the capability of testing a software artefact is strongly influenced by the information available [3]. In fact, different kinds of testing techniques can be applied depending on the extent and formalization degree of the information available. The technique to be applied will also be different depending on the quality aspects to be evaluated, e.g. functionality, performance, interoperability, etc.

In CBSD, different proposals have been made to increase the information available with software components [24], following what we generally refer to as the metadata-based testing approach [25]. Fortunately, as already said, today the area of WS can rely on a more mature attitude towards the need for standardized documentation, with respect to the situation faced by early component developers, and in fact the interaction among WSs is based on a standardized protocol stack and discovery service. Current practice is that the information shared to develop interacting WSs is stored in WSDL files. However, such documents mainly report signatures (or syntax) for the available services, but no information concerning specific constraints about the usage of the described service can be retrieved. Obviously, this way of documenting a service raises problems regarding the capability of correctly integrating different services. In particular, the technology today relies on the restrictive assumption that a client knows in advance the semantics of the operations provided by a service or other properties of it [1].

To facilitate the definition of the collaborations among different services, various approaches are being proposed to enrich the information that should be provided with a WS. Languages such as the Business Process and Execution Language for Web Services (BPEL4WS) and the Web Service - Choreography Description Languages (WS-CDL) are emerging [1], which permit to express how the cooperation among the services should take place. The formalized description of legal interactions among WSs turned out to be instrumental in verifying interoperability through the application of specific conformance evaluation instruments.

We claim that it would be highly useful to attach this description in the form of an XML Metadata Interchange (XMI [29]) file, since in this form it can be easily reused by UML based technologies. XMI is becoming the de facto standard for enabling interaction between UML tools, and it can be automatically generated from widespread UML editors such as IBM Rational Rose XDE or Poseidon.

It is indeed somewhat surprising how two broad standardization efforts, such as the UML and the WSA, are following almost independent paths within distinct communities. Our motivating goal is the investigation of the possibility to find a common ground for both communities. Hence our proposal is that the WS description (including the WSDL file) will report some additional information documented by the WS developer in UML, and in particular, as we explain below, as a Protocol State Machine, that is a UML behavior diagram newly introduced into the latest version of this language [11]. In this way an XMI file representing the associated PSM could be inserted in the UDDI registry along with the other WS documentation. Moreover, as we show in this paper, the PSM provides a formal description of the legal protocol for WS interaction, and following some translation step it can be used as a reference model for test case derivation, applying well established algorithms from formal testing theory.

The framework for automatic testing of WSs presented in this paper has been specifically defined considering technologies related to the WS domain. It will probably be straightforward to apply a similar approach also in a Component Based (CB) setting when the necessary information is provided as data attached to the component. WSs can be considered as being an extreme consequence of the CB paradigm, in which the developer of a system looses the control, also at run time, of the "assembled" components.

The paper is structured as follows: Section 2 provides an overview of the different flavors of the interoperability notion for WSs, and in particular introduces WS conformance testing; Section 3 presents PSMs and their proposed usage for WS protocol specification; Section 4 synthesizes the general framework we propose for WS testing, and Section 5 outlines related work. In Section 6 a short survey of formal approaches to conformance testing is given, before focusing on the specific formalism which we are going to exploit for WS conformance testing, called Symbolic Transition Systems (STSs). In Section 7 we relate the PSM specification to the presented STS one. Finally, Section 8 summarizes our conclusions and mentions the work remaining to be done.

2 Interoperability of Web Services

Web Services are cooperating pieces of software that are generally developed and distributed among different organizations for machine-to-machine cooperation, and which can act, at different times, as servers, providing a service upon request, or as clients, invoking some others' services. The top most concern in development of WSA is certainly WS interoperability. Actually, WS interoperability is a wide notion, embracing several flavors, all of them important. Without pretending to make a complete classification, for the sake of exposition in our case we distinguish between two main kinds of interoperability issues. A first type of interoperability refers to the format of the information stored in the relevant documents (such as WSDL files, UDDI entry), and to the format of the exchanged SOAP messages. This interoperability flavor is briefly presented below in Section 2.1, in which the approach defined by the WS-I consortium (where the "I" stands for Interoperability) to ensure this kind of interoperability is outlined. A second interoperability issue, discussed in Section 2.2, is instead relative to the correct usage of a WS on the client's side, in terms of the sequencing of invocations of the provided services. Certainly, other kinds of heterogeneity hindering correct interactions of WSs can be identified. For instance, in [16] the authors report about an interesting experience in integrating externally acquired components in a single system. As they highlight, different assumptions made by the different components, such as who has to take the control of the interaction, often prevent real interoperability.

2.1 Data and Messaging Conformance

As said, a first factor influencing the interoperability of WSs is obviously related to the way the information is reported in the different documents (such as SOAP messages, WSDL files, UDDI entries) necessary to enable WS interactions, and to the manner this information is interpreted by cooperating WSs.

This concern is at the heart of the activities carried on by the WS-I consortium, an open industry organization which joins diverse communities of Web Service leaders interested in promoting interoperability. WS-I provides several resources for helping WS developers to create interoperable Web Services and verify that their results are compliant with the WS-I provided guidelines. In particular, WS-I has recently defined several profiles [12] that define specific relations that must hold among the information contained in different documents. In so doing the interactions among two or more WSs are enabled. Besides, WS-I has defined a set of requirements on how specific information reported in these files must be interpreted.

Just to show the kind of interoperability addressed by WS-I we report below without further explanation a couple of examples from the specification of the Basic Profile. Label R1011 identifies a requirement taken from the messaging part of the profile, which states:

```
R1011 - An ENVELOPE MUST NOT have any element children of soap:
Envelope following the soap:Body element.
```

The second example has been taken from the service description part and describes relations between the WSDL file and the related SOAP action:

```
R2720 - A wsdl:binding in a DESCRIPTION MUST use the part
attribute with a schema type of "NMTOKEN" on all contained
soapbind:header and soapbind:headerfault elements.
```

Alongside the Basic Profile, the WS-I consortium also provides a test suite (that is freely downloadable from the WS-I site) that permits to verify the conformance of a WS implementation with respect to the requirements defined in the profile. In order to be able to verify the conformance of the exchanged messages, part of the test suite acts as a proxy filtering the messages and verifying that the conditions defined in the profile are respected.

The WS-I profile solves many issues related to the representation of data, and to how the same data are represented in different data structures. Another kind of data-related interoperability issue not addressed by the WS-I profile concerns instead the interpretation that different WSs can give for the same data structure. Testing can certainly be the right tool to discover such kind of mismatches. The testing theory presented in Section 6 and its application to the domain of WSs, exerted in Section 7, provides a formal basis for the derivation of test cases that permit to verify that a single implementation of a WS correctly interprets the exchanged data.

2.2 Protocol Conformance

A different interoperability flavor concerns the correct usage of a WS on the client's side, in terms of the sequencing of invocations of the provided services. A correct usage of a WS must generally follow a specified protocol defining the intended sequences of invocations for the provided interface methods, and possibly the pre- and post-conditions that must hold before and after each invocation, respectively.

This is the kind of interoperability we focus on in the remainder of this paper. Generally speaking a protocol describes the rules with which interacting entities