SENSORIA

Software Engineering for Service-Oriented Overlay Computers

A Systematic Approach to Developing Service-Oriented Systems

White Paper

Integrated Project funded by the European Community under the
"Information Society Technologies" Programme (2002—2006)
016004

Authors

Martin Wirsing, Giulio Carizzon, Stephen Gilmore, Laszlo Gönczy, Nora Koch, Philip Mayer, Claudio Palasciano

Revision

2007-10-17 (Final)

Contract Start Date

September 1, 2005 Duration: 48 months

Projects Partners

Coordinator: Ludwig-Maximilians-Universität München
Università di Trento | University of Leicester | Warsaw University | TU Denmark at Lyngby
Università di Pisa | Università di Firenze | Università di Bologna | ISTI Pisa | Universidade de Lisboa
University of Edinburgh | ATX Software SA | Telecom Italia Lab | Imperial College London
Cirquent (FAST GmbH) | Budapest University of Technology and Economics | S&N AG
University College London | Politecnico di Milano

Abstract

Service-Oriented Computing is an emerging paradigm where services are understood as autonomous, platform-independent computational entities that can be described, published, categorised, discovered, and dynamically assembled for developing massively distributed, interoperable, evolvable systems and applications. Various approaches to engineering service-based systems are in use today; however, critical problems concerning among others automated composition, performance, security and safety, are still not solved satisfactorily, requiring more fundamental studies ranging from essential research questions to practical development and deployment issues.

In order to address these challenges, the IST-FET Integrated Project SENSORIA aims at developing a novel comprehensive approach to the engineering of service-oriented software systems where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach, supporting semi-automatic development and deployment of self-adaptable (composite) services.

In this white paper, we give a short introduction into Service-Oriented Computing and problems associated with current approaches. Afterwards, we present the SENSORIA approach to the development of service-oriented systems. This includes a new generalised concept of service, new semantically well defined modelling and programming primitives for services, new powerful mathematical analysis and verification techniques for assessing system behaviour and quality of service properties, and a suite of tools supporting model-based transformations, development, and deployment of service-oriented systems as well as reengineering legacy software into services.

Contents

1 Introduction

The last decades have shown tremendous advances in the field of Information Technology (IT). In fact, with online access to vast amounts of information, states offering their services for citizens on the World-Wide Web, and software in control of critical areas such as flight control systems, information systems now lie at the very foundations of our society. But it is the business sector where IT has had the most impact: Driven by the need to stay competitive in an environment of rapid change in global markets and business models as well as local regulations and requirements, organizations now strongly depend on a functional and efficient IT infrastructure which is flexible enough to deal with unexpected changes while still offering stability for business processes and connections to customers, partners, and suppliers.

With the emerging trend of automating complex and distributed business processes as a whole, many organizations face the problem of integrating their existing, already vastly complicated systems to reach yet another layer of sophistication in the interconnection of business value, business processes, and information technology. Many IT systems are difficult to adapt and work with and progress in the world of businesses is often impeded by the difficulty of changing the existing IT infrastructure.

In order to deal with these challenges, a new approach to software development and integration was devised. This approach, called Service-Oriented Computing (SOC), provides the opportunity for organizations to manage their heterogeneous infrastructures in a coherent way, thus gaining new levels of interoperability and collaboration within and across the boundaries of an organization. In addition, SOC promises new flexibility in linking people, processes, information, and computing platforms.

The IST-FET Integrated Project SENSORIA is a European Community funded project that develops methodologies and tools for dealing with Service-Oriented Computing. It addresses major problems found in current approaches to SOC and provides mathematically founded and sound methodologies and tools for dealing with the amount of flexibility and interoperability needed in these next-generation infrastructures [5][16].

This white paper is structured as follows: We will provide a general overview of Service-Oriented Computing in chapter 2. In chapter 3, we will introduce the project SENSORIA, starting from the overall aims of the project and moving on to research topics and results. Finally, chapter 4 will conclude this paper.

2 Service-Oriented Computing

2.1 Introducing SOC

Service-Oriented Computing (SOC) is a new computing paradigm for the development and management of large IT systems for business organizations. Applications based on SOC employ *services* as their basic computational entities, a service being defined as an independent and autonomous piece of functionality which can be described, published, discovered, and used in a uniform way, in most cases even without knowledge of the underlying implementation and actual physical location. The functionality a service can provide ranges from answering simple requests like a dictionary translation to complex functionality like a bank transaction system. Existing pieces of software may be wrapped as services, thus offering their functionality within the context of SOC systems.

Key to the SOC approach is the ability to *compose* existing services to form new, higher-level functionality. These compositions – also called orchestrations – of services form another service in its own right, which can again be composed into even more high-level compositions. As services can be dynamically located and invoked by other services (in particular, by service compositions), the general organization of a SOC-based infrastructure is *loosely coupled*. In this way, SOC enables building so-called *overlay computers* which may span organizations, states, and continents.

Due to the dynamic nature of service interconnection and communication, service compositions may be re-evaluate their partners at any moment, thus allowing for replacement of failing or unsafe services and thereby reacting very quickly to a changing environment. This architecture allows SOC-based systems to provide robust service in difficult operational conditions.

Service-Oriented Computing is of enormous strategic importance for society and key enabling technology for IT-rich areas of modern life. Several of the key concepts of Service-Oriented Computing are directly connected to economic benefits to IT-dependent organizations. Moving to SOC saves investments, as existing software may be wrapped and thus made available in a more decoupled way. By using services as the basic computational entities, developers can grow new applications by merely combining existing services rather than having to create a new monolithic application from scratch, while at the same time keeping the system loosely coupled. Also, the flexible nature of service interactions allows organizations to quickly react to changes in its environment by replacing services or reconfiguring/reorganizing service compositions.

The field of Service-Oriented Computing touches many existing pragmatic approaches to software development as well as research areas of computer science, and in many cases poses hard problems to both IT and business professionals alike. Although we are beginning to see SOC-based systems in action, fulfilling the overall promises of SOC is still ongoing work for both industry and research [11].

2.2 Answering the SOC Challenge

The implementation of Service-Oriented Computing requires a certain design of the application landscape. This design is known by the term Service Oriented Architecture (SOA) – an architectural paradigm which describes the fundamental design of service-oriented software. A SOA can be implemented using several available technologies such as Web Services [15], Grid Computing [6], the OSGi framework [13], the Microsoft Distributed Component Object Model (DCOM) [10], and the OMG Common Object Request Broker Architecture (CORBA) [3].

While component approaches like OSGi, DCOM or CORBA are still heavily dependent on vendor platforms and a tight coupling of client and server [8], Web Services and Grid Computing move forward into the domain of development and execution of business processes that are distributed over a network and available via standard interfaces and protocols. However, all of these approaches suffer from fundamental problems which have not yet been solved satisfactorily. In particular:

- Specification and querying of services, in particular with regard to the correct behaviour of a service along with the data exchanged, is still error-prone due to a lack in expressivity and verifiability of service behaviour in current technologies and standards. However, the inter-organisational nature of business processes and automated interactions between services leave no room for misunderstandings due to different meaning of service specifications or data exchanged between applications [14].
- Current composition architectures have an additional problem: They depend on the perpetual availability of the services and lack the flexibility to replace a failed service with an alternative [12][4][17]. However, due to the linked nature of services across a network, fall-back mechanisms are vital to ensuring continued operation in case of non-availability of single services.
- Service implementations currently in use are often prone to network errors and security attacks [9]. Major security concerns for service implementations have not been solved satisfactorily. In particular, confidentiality of data, authentication or authorisation of participating parties, integrity, non-repudiation, and identity management are still open issues [8][2].
- Measuring and reacting to performance issues at design and runtime is still in an immature state; lacking systematic support for sensitivity and throughput analysis which can be used for large-scale applications [1]. Also, scaling analysis along with increasing system size and complexity remains a major research challenge, as large systems have different requirements with regard to analysis methods and algorithms [7].
- In general, a major problem of current service engineering approaches is missing support for analysis, validation and verification at every stage of the development process. In many areas, SOA software development is still done in an ad-hoc manner; and methods and tools are lacking for systematically constructing, analyzing and deploying service-oriented systems.

Many of these issues touch the fundamental approach to Service-Oriented Computing and Service-Oriented Architectures and thus can only partly be solved by merely adding to existing standards. Therefore, solutions must be found by looking beyond standard technologies to new ways of dealing with services and Service-Oriented Computing.

In the next chapter, we will detail how we answer these challenges within SENSORIA.

3 SENSORIA

The core aim of the EU SENSORIA project is the production of new knowledge for systematic and scientifically well-founded methods of service-oriented software development. SENSORIA provides a comprehensive approach to the visual design, formal analysis, and automated deployment of service-oriented applications. The SENSORIA techniques enable service engineers to model their applications on a very high level of abstraction using service-oriented extensions of the standard UML or standard business process models, and to transform those models to be able to use formal analysis techniques as well as generate executable code.

3.1 Research Themes

The research themes of SENSORIA range across the whole lifecycle of software development from requirements to deployment including reengineering of legacy systems. SENSORIA methods and tools rely on mathematical theories and methods that, ensuring the correctness of each step, allow a semi-automatic design process. Realistic case studies for different important application areas including telecommunications, automotive, e-learning, and e-business are defined by the industrial partners to provide continuous practical challenges for the new techniques of services engineering and for demonstrating the research results (Figure 1).

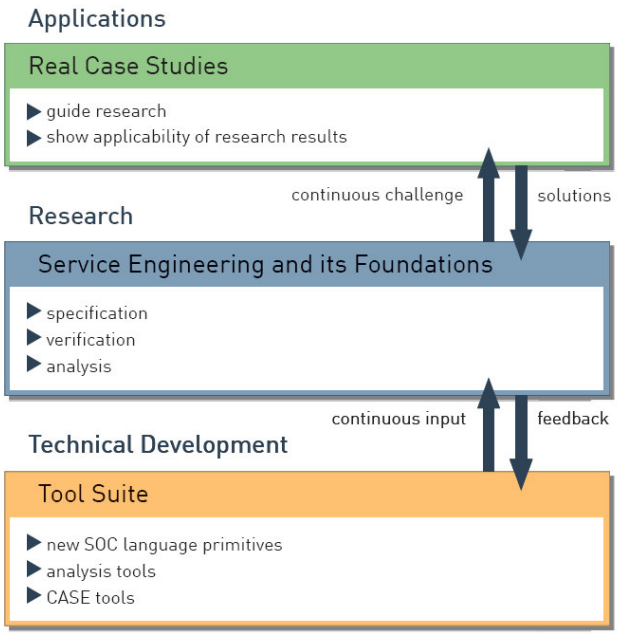

Figure 1: SENSORIA Innovation Lifecycle

As the main innovative aspect of SENSORIA, high-level service models are first translated into hidden formal representations by automated model transformations. Our tools are able to perform checks of functional correctness of services, early performance analysis, prediction of quantitative bottlenecks in collaborating services, and verification of service level agreements. Finally, results of the mathematical analysis are made available in the models to provide feedback to the developer.

The interplay of the different research themes and activities of SENSORIA are illustrated in Figure 2.

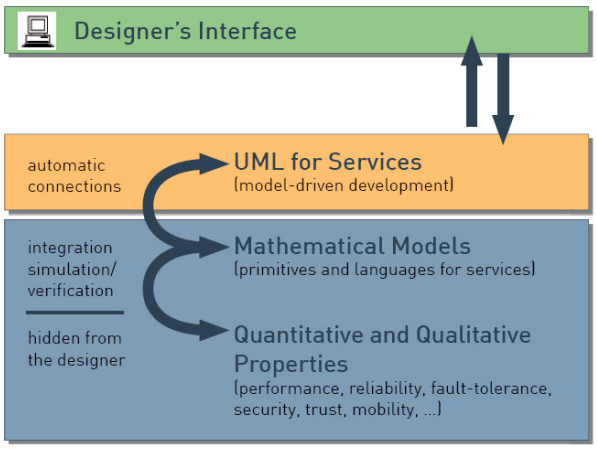

Figure 2: The SENSORIA Approach

The figure shows a high-level overview of the three main research topics of SENSORIA:

- **Modelling Service-Oriented Software.** The definition of *adequate linguistic primitives* for modelling and programming service-oriented systems enables *model-driven development* for implementing services on different global computers, and for transforming legacy systems into services using systematic re-engineering techniques. *Modelling front-ends* allow designers to use high-level visual formalisms such as the industry standard UML or domain-specific modelling languages to precisely capture domain-specific requirements. This allows developers to focus on what matters most, i.e. creating business applications based on Service-Oriented Architectures. Later on, automated *model transformations* allow generation of formal representations from engineering models for further development steps.
- **Formal Analysis of Service-Oriented Software Based on Mathematically Founded Techniques**. *Mathematical models* for service computing formalise different aspects of overlay computers: at this level, services are seen as *abstract computational entities*, modelled in a platform-independent architectural layer. The mathematical models, hidden from the developer, enable q*ualitative and quantitative analysis* supporting the service development process and providing the means for reasoning about functional and non-functional properties of services and service aggregates. These techniques may be integrated into several software process models such as the Model-Driven Architecture (MDA), the Rational Unified Process (RUP), or Test-Driven Development (TDD). SENSORIA results include powerful mathematical analysis techniques; in particular *program*

analysis techniques, type systems, logics, and process calculi for investigating the behaviour and the quality of service of properties of global services. These techniques are then tailored to several specific purposes: Firstly, they can be used to reveal performance bottlenecks or interactions leading to errors or violation of service contracts. Secondly, they are used to deal with security issues like confidentiality, integrity, non-interference, access control, and trust management. Finally, for critical services, deep semantic analysis may be used for certification.

- **Deploying and Runtime Issues of Service-Oriented Software**. The development of *sound engineering techniques for global services* includes deployment mechanisms with aspects like runtime self-management, service-oriented middlewares, and model-driven deployment as well as reengineering legacy systems into services, thus enabling developers to take the last mile to the implementation of Service-Oriented Architectures.

The added value of SENSORIA comes from the availability of sound engineering techniques supported by mathematical foundations, languages with formal semantics and associated analysis methods, and the ability to automate many of the development steps currently done by hand in the design of Service-Oriented Software.

3.2 Technologies and Tools

The SENSORIA development approach laid out above is not only provided through methodologies and mathematical foundations, but also through a concrete set of tools aiding developers in all the steps necessary to model, implement, test, validate, and deploy service-oriented software. Various tools are developed within SENSORIA which are integrated into the main SENSORIA Case Tool, an integration platform which offers an Eclipse-based environment for employing fully-customizable tool chains for the entire model-driven workflow (modelling, development, analysis and deployment).

The SENSORIA CASE Tool enables developers to:

- Use the SENSORIA tools for SOA development in a homogeneous environment
- Discover and publish new tools for SOA development
- Orchestrate tools via scripts, thereby automating SOA development

Figure 3 shows how the SENSORIA CASE Tool may be extended with tools to offer an integrated development environment for Service-Oriented Systems.

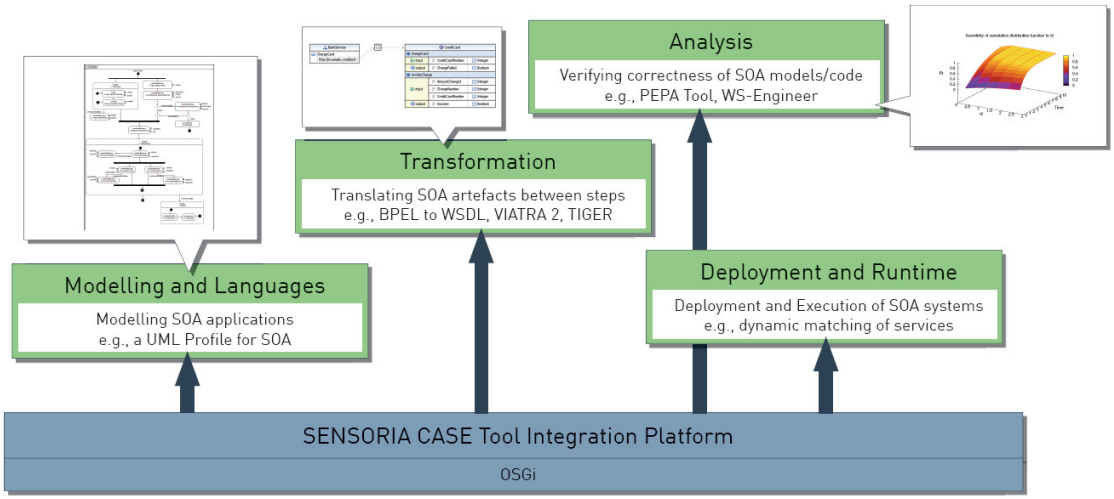

Figure 3: The SENSORIA CASE Tool

As can be seen from the figure, the SENSORIA CASE Tool Integration Platform enables developers to use the tools of SENSORIA in combination for modelling services, transforming SOA artefacts between development steps, analyzing SOA models or code, and finally deploying and running SOA systems.

4 Summary

Service-Oriented Computing and the Service-Oriented Architecture as the main architectural design pattern for Service-Oriented Software are having a huge impact on IT-based business organizations across the world. However, as we have shown, there are still many open issues regarding the development, analysis, and deployment of such software, some of which touch the very foundations of Service-Oriented Computing.

As a remedy, the EU project SENSORIA is developing a novel comprehensive approach to the engineering of service-oriented software systems where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach, thereby supporting semi-automatic development and deployment of self-adaptable (composite) services. The impact of SENSORIA on services development will be to bring mathematically well-founded modelling technology within the reach of service-oriented software designers and developers.

By offering these techniques and tools, we hope to allow IT-dependent organizations to move to a higher and more mature level of SOA software development. In particular, we hope to contribute to an increased quality of service of SOA applications, measured both in qualitative and quantitative terms. As SENSORIA methods are portable to existing platforms, application of these methods is possible while keeping existing investments.

To learn more, please visit our website www.sensoria-ist.eu.

References

[1] Chandrasekaran, S., Miller, J., Silver, G., Arpinar, I., Sheth, A. Performance Analysis and Simulation of Composite Web Services. Electronic Markets 13(2) (2003)

[2] Coetzee, M., Eloff, J.H.P. Web Services Access Control Architecture Incorporating Trust, Internet Research Volume 17 Number 3 2007 pp. 291-305

[3] Common Object Request Broker Architecture (CORBA). Object Management Group, http://www.omg.org/, last visited: October 2007

[4] Dahlem, D., Nickel, L., Sacha, J., Biskupski, B., Dowling, J., Meier, R.. Towards Improving the Availability of Service Compositions, Presented at the Inaugural IEEE Int. Conf. on Digital Ecosystems and Technologies (DEST'07)

[5] FET Global Computing Initiative, http://cordis.europa.eu/ist/fet/gc.htm, last visited: August 2007

[6] Foster, I., Kesselman, C. The Grid: Blueprint for a New Computing Infrastructure. Elsevier Series in Grid Computing. Morgan Kaufmann Publishers, 2004.

[7] Hillston, J (2005). Fluid-flow approximation of PEPA models. QEST 2005, IEEE Society Press.

[8] Joshi, P., Singh, H., Phippen, A.D. Web Services: Measuring Practitioner Attitude, Internet Research Volume 14 Number 5 2004 pp. 366-371

[9] Malek, M.; Harmantzis, F. Security management of Web Services. Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP Volume 2, pp. 19-23, April 2004.

[10] Microsoft COM (Component Object Model) technology. http://www.microsoft.com/com/, last visited: October 2007

[11] Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F. Service-Oriented Computing Research Roadmap. March 1, 2006. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/services-research-roadmap_en.pdf

[12] Phippen, A.D., Taylor, J., Allen, R. Issues in moving from Web Services to Service Orientation, Internet Research, Volume 15 Number 5 2005 pp. 518-526

[13] The OSGi Alliance. http://www.osgi.org/, last visited: October 2007

[14] The Web Services Roadmap. http://roadmap.cbdiforum.com, last visited: August 2007

[15] Weerawarana, S., Curbera, F., Leymann, F., Tony Storey Ferguson, D. F. (2005). Web Services Platform Architecture. Prentice Hall PTR

[16] Wirsing, M., Clark, A., Gilmore, S., Hölzl, M., Knapp, A., Koch, N., Schroeder, A. Semantic Based Development of Service-Oriented Systems, FORTE 2006, LNCS 4229, pp. 24–45, 2006.

[17] Yana, J., Kowalczykb, R., Linb, J., Chhetrib, M., Zhang, S.J. Autonomous Service Level Agreement Negotiation for Service Composition Provision, Future Generation Computer Systems, Volume 23, Issue 6, July 2007, pp. 748-759