simpA-WS: An Agent-Oriented Computing Technology forWS-based SOA Applications

Alessandro Ricci, Claudio Buda, Nicola Zaghini, Antonio Natali, Mirko Viroli, Andrea Omicini
DEIS, Università di Bologna Alma Mater Studiorum
Cesena (FC), 47023 Italy
Emails: {a.ricci, claudio.buda, nicola.zaghini, antonio.natali, mirko.viroli, andrea.omicini}@unibo.it

***Abstract*— This document briefly describes simpA-WS, a Java-based agent-oriented computing technology to flexibly and effectively implement WS-I compliant SOA/WS applications—i.e. Web-Service applications with a Service-Oriented Architecture—both on the user side and the service side.**

I. Introduction

simpA-WS is a Java-based technology that makes it possible to build WS-I SOA/WS compliant applications adopting an agent-oriented style in designing and developing the systems. simpA-WS is based on of simpA model and technology [1], an *agent-oriented* extension of the Java Object-Oriented computational model providing high-level abstractions for designing and developing complex software systems.

On the one side, simpA-WS provides a framework API to build *user applications* in terms of sets of agents that flexibly interact and use Web Services compliant with the WS-I Basic Profile [12], represented as *artifacts* in agent workspaces. On the other side, simpA-WS provides an API framework and a middleware for building WS-I compliant Web Services in terms of set of agents as providers of the services.

A. Human Cooperative Working Environments as Background Metaphor

To tackle the complexity of modern and future software systems, simpA introduces *agents* and *artifacts* as high-level abstractions to design and build distributed / concurrent software systems. This is the A&A (Agents and Artifact) conceptual model [16], [15], [10], based on inter-disciplinary studies involving Activity Theory and Distributed Cognition as main conceptual background frameworks [9].

A&A metaphors are taken from human cooperative working environments, where "systems" are composed by individual autonomous entities which pro-actively carry on some kind of activities or tasks, both individual and cooperative, typically requiring forms of interaction and coordination with other individuals. A fundamental aspect of such a picture is the context —or the *environment*— that makes it possibile to such activities to take place. *Artifacts* designed and built by humans are an essential part of such a context, both as outcome of the activities and as the *tools* exploited by humans to support and realise them. Artifacts can be then resources and objects constructed during the activities, but also whatever tools is used to support humans communication, coordination, and—more generally—cooperative working activities. The overall picture is given by *workspaces* where ensembles of individuals work and interact in a coordinated manner, by communicating and sharing / using the same artifacts, so as to achieve some kind of objective.

A&A brings this metaphor down to software engineering, conceiving a software system as one or multiple workspaces where ensembles of autonomous entities—the agents—execute their working activities and interact by co-constructing, sharing and using artifacts, analogously to the human case.

simpA provides agents and artifacts as basic high-level building blocks to decompose and structure complex systems, in particular:

- agents are provided as first-class abstractions to model and design *pro-active* entities, i.e. entites programmed so as to autonomously execute some kind activity—composed by one or more *tasks*—encapsulating the control of such activities;
- artifacts are provided as first-class abstractions to model and design what is used or constructed by agents during their activities, including resources, tools, devices: any *passive* entity encapsulating some kind of functionality, exposed by a proper interface;
- *workspaces* are the logical place where agents and artifacts are immersed, used to give a topology to the overall activities, and then partition the application environment.

Objects and classes are used as basic abstractions to define data structures to build agents and artifacts.

B. A&A *for Implementing SOA / Web Service Applications*

simpA-WS makes it possible to exploit the A&A approach and simpA for implementing SOA and Web-Service applications, following the basic architectured described in W3C documentation [17](see Fig. 1).

Using simpA-WS both service users and providers are modelled as simpA agents, as pro-active entities that respectively *(i)* need to access and use services in their working activities, encapsulating the business logic of user applications, and *(ii)* process the requests and messages for services, encapsulating the service business logic. In both cases, artifacts are used as high-level mediating entities functioning as interfaces, encapsulating the technology needed to enable the interaction using WS-* standards. In particular (refer to Fig. 2 and Fig. 3):

- on the user side, a specific kind of artifacts – called WS-Artifact — is provided to make it possible for simpA

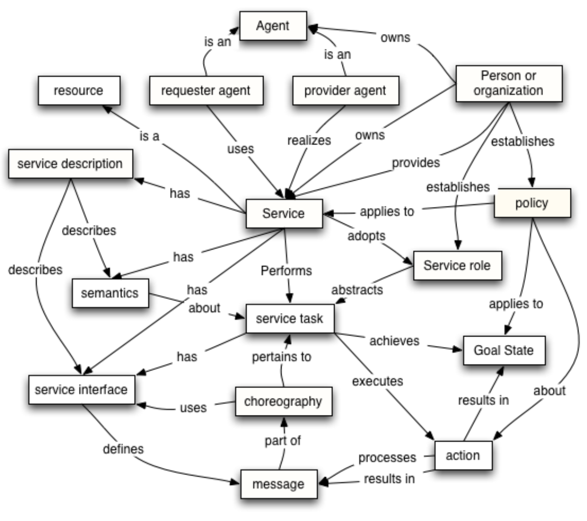

Fig. 1. Web Service Service Model according to W3C

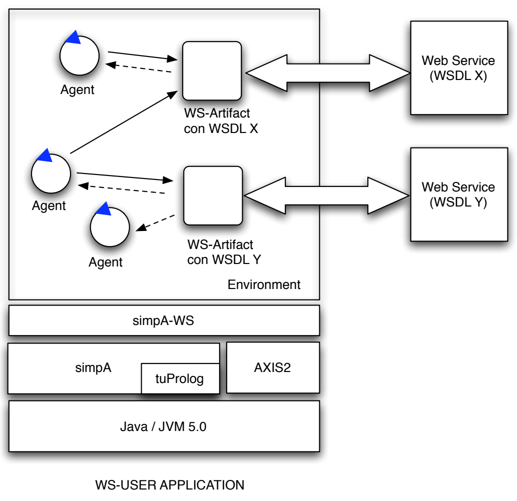

Fig. 2. Abstract architecture of simpA-WS user applications

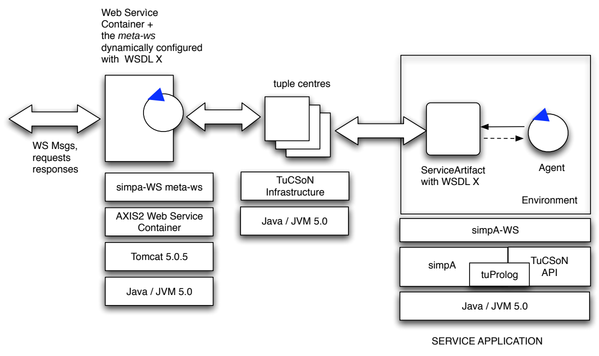

Fig. 3. Abstract architecture of simpA-WS service applications

agents to access and use what ever existing Web Service, by simply creating and using instances of such an artifact;

- on the service side, artifacts called Service-Artifacts are provided to make it possible for simpA agents to get and process the messages delivered to a specific Web Service, again by simply creating and using instances of such an artifact.

The adoption of the agent level of abstraction, and in particular of agents and artifacts basic building blocks, makes it possible to exploit a fully uncoupled approach for modelling and realising interaction with Web Services, as required by true service-oriented architecture [6]. On the one side, agents *use* Web Services by executing operations on WS-Artifact artifacts, by means of fully asynchronous *actions*. On the other side, agents *perceive* possible information or result generated Web Services by observing events — through simpA *sensing* primitives — generated by such artifacts.

II. System Requirements

Currently simpA-WS can run on any Java-based platform, version 5.0. Consequently, it can be seamlessly deployed on machines with different kind of operating systems and hardware configuration. A light-weight version is planned, in order to port simpA-WS—the API for implementing user application in particular—on J2ME, the Java platform for mobile devices, so as to implement user applications on top of J2ME-enabled smart phones / devices.

III. Design and Implementation

Fig. 2 and Fig. 3 provide an overview of the architecture of the simpA-WS technology, respectively on the user side and the service side. Fig. 4 shows a mixed and articulated case, where services are themselves users of other services.

simpA and simpA-WS are fully developed in Java, and exploit apache Axis2 open-source libraries [7] for realising low-level SOAP-based interaction with Web Services on the user side and as Web Service container on the service side, on top of Tomcat apache Java-based Web Server [8].

simpA and simpA-WS internally exploits tuProlog [5] and TuCSoN [11] research technologies, respectively a light-weight Java-based Prolog interpreter and a tuple-based agent coordination infrastructure.

IV. A Sample Application

Among the examples provided with simpA-WS distribution, a supply-chain sample application is provided, following the reference sample application defined by WS-I organisation, available among the deliverables at WS-I Web Site [13]. The supply-chain example is one among the illustrative scenarios defined by the WS-I Sample Applications Working Group to show the benefits of having interoperable Web services applications, and to demonstrate the application of the WS-I profiles to those scenarios.

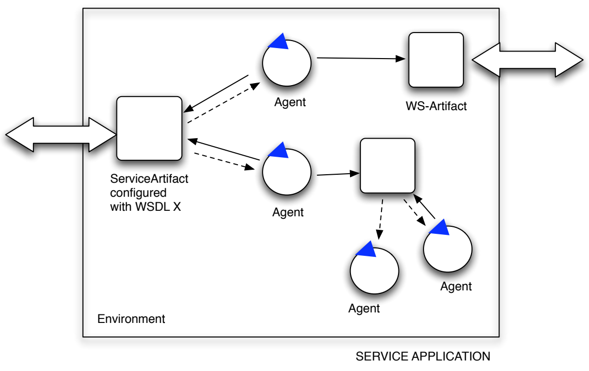

Fig. 4. Abstract architecture of mixed simpA-WS applications

V. simpA-WS in Real-World and Industrial Applications

simpA-WS is one of the technologies experimented for implementing SOA-based applications in the context of STIL ("Strumenti Telematici per l'Interoperabilità delle reti di imprese: Logistica digitale integrata per l'Emilia-Romagna").

STIL is a 2-years project funded by Emilia-Romagna, in the context of the "Iniziativa 1.1 del Piano Telematico Regionale" initiative [14]. The project has been funded to push and improve the research activities in Emilia-Romagna targeted to exploit innovative ICT technologies for the creation of a global digital logistic district. Among the objectives, STIL is dedicated to the creation of virtual organizations grouping together different kind of actors directly or indirectly involved in the logistic supply-chain, providing them effective ICT supports for integrating and innovating their business.

For the purpose, a SOA-based infrastructure has been conceived, designed and implemented to enable interoperability among the different participants. The STIL infrastructure is meant to provide an effective support for enabling communication, coordination and cooperation among the open and heterogeneous kind of WS-based applications and services. simpA-WS is currently experimented as one of the state-of-the-art technologies for implementing the applications and services, and first results are available on STIL web sites [14], [3].

VI. Industrial Support and Distribution

simpA and simpA-WS are open-source projects, and can be freely downloaded and used for research and non-commercial purposes from related web sites [1], [2]. Besides the open-source prototypes, an industrial-version of the technology will be available as commercial product distributed by IRIS [4], a start-up spin-off company hosted in Cesena.

References

[1] The aliCE Research Group. simpA official web site. http://www.alice.unibo.it/projects/simpa.
[2] The aliCE Research Group. simpA-WS official web site. http://www.alice.unibo.it/projects/simpa-ws.
[3] aliCE-Unibo research unit. The STIL-UNIBO project web site. http://www.alice.unibo.it/projects/stil.
[4] The IRIS Company. IRIS official web site. http://www.irislab.eu.
[5] Enrico Denti, Andrea Omicini, and Alessandro Ricci. Multi-paradigm Java-Prolog integration in tuProlog. *Science of Computer Programming*, 57(2):217–250, August 2005.
[6] Thomas Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
[7] The Apache Software Foundation. Axis 2.0 — next generation web services — official web site. http://ws.apache.org/axis2/.
[8] The Apache Software Foundation. Tomcat official web site. http://tomcat.apache.org/.
[9] B. A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.
[10] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. *Agens Faber*: Toward a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Sciences*, 150(3):21–36, 29 May 2006. 1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 April 2005. Proceedings.
[11] Andrea Omicini and Franco Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, September 1999. Special Issue: Coordination Mechanisms for Web Agents.
[12] The WS-I Organization. WS-Basic Profile 1.0 document. http://www.ws-i.org.
[13] The WS-I Organization. WS-I Official web site. http://www.ws-i.org.
[14] The STIL project. The STIL official web site. http://stil.pc.unicatt.it/.
[15] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. *Construenda est* CArtAgO: Toward an infrastructure for artifacts in MAS. In Robert Trappl, editor, *Cybernetics and Systems 2006*, volume 2, pages 569–574, Vienna, Austria, 18–21 April 2006. Austrian Society for Cybernetic Studies. 18th European Meeting on Cybernetics and Systems Research (EMCSR 2006), 5th International Symposium "From Agent Theory to Theory Implementation" (AT2AI-5). Proceedings.
[16] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Programming MAS with artifacts. In Rafael P. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems*, volume 3862 of *LNAI*, pages 206–221. Springer, March 2006. 3rd International Workshop (PROMAS 2005), AAMAS 2005, Utrecht, The Netherlands, 26 July 2005. Revised and Invited Papers.
[17] W3C WS Working Group. Web Services Architecture. http://www.w3.org/TR/ws-arch/.