Eclipse SoaML: a Tool for Engineering Service Oriented Applications

Andrea Delgado, Laura González

Instituto de Computación, Facultad de Ingeniería, Universidad de la República

Julio Herrera y Reissig 565,11300, Montevideo, Uruguay
{adelgado, lauragon}@fing.edu.uy

Abstract. Service-Oriented applications focus on the definition of self-contained independent pieces of software called services providing specific functionality, being nowadays one of the most used implementations to support business process execution. Although languages, technologies and platforms supporting the service-oriented paradigm have evolved within the last decades, it is not the case of service modeling support. Service modeling is a key aspect for the design and specification of services and for automating the different stages following the model driven development vision. SoaML is a standard for service modeling which defines an UML profile and a metamodel extending the UML metamodel, defining explicitly concepts and elements for services specification. The Eclipse SoaML tool provides support for service modeling with UML using the SoaML standard, allowing importing and exporting models in XMI format to interoperate with other tools. From SoaML models the tool also generates the code needed to implement services in Java EE and Web Services.

Keywords: Service Oriented Architectures (SOA), service modeling, service development, standards, SoaML, JEE, WS.

1 Introduction

Service Oriented Computing (SOC) [1] promotes the design of applications based on services that are reusable software components by means of which consumers and providers interact in a decoupled way. A Service Oriented Architecture (SOA) [2][3] is an architectural style that supports service orientation providing the means for designing and specifying services. Services are usually defined as self-contained independent pieces of software providing specific functionality, being nowadays one of the most used implementations to support the automatic execution of business processes. Business Process Management [4][5] provides the means for modeling, executing and assessing business processes and their implementations in organizations.

While languages, technologies and platforms support for the implementation and execution of services is an area that in recent years has evolved considerably, the design and modeling of services is still immature. Service modeling is essential among other things, for automating various stages of software development using

Model Driven Development (MDD) [6][7], such as code generation. Service models can be enhanced with platform specific information to obtain the corresponding code, providing traceability from requirements to code which eases software maintenance. The Service Oriented Architecture Modeling Language (SoaML) [8] standard is based on UML and is one step in the direction of providing support for service modeling by defining specific concepts and stereotypes for modeling services. Currently there is a lack of support for the SoaML standard since there are no more than a few implementations of it which are mostly commercial tools[1].

The Eclipse SoaML tool we present here is composed of two Eclipse plug-ins which provide support for engineering service-oriented applications: the first one is a SoaML editor which implements the SoaML standard allowing service modeling in SoaML and interchange of service models in XMI format; the second one is a code generator for implementing services with Java EE and Web Services which takes as input a SoaML service model and generates the corresponding code. Both Eclipse plug-ins were developed as part of a broader research work regarding the continuous improvement of business processes implemented by services with a model-driven focus, reflected in the definition of the framework MINERVA [9][10][11][12][13].

The rest of the article is organized as follows: in section 2 we describe the SoaML standard, in section 3 we present the Eclipse SoaML Tool we have developed to support SoaML modeling and code generation, in section 4 we discuss related work, and finally in section 5 we present some conclusions and future work.

2 SoaML standard

The SoaML standard for service modeling has been recently released in its version 1.0.1 (May, 2012) and its first beta release was only five years ago (April, 2009). It defines the concepts and corresponding elements and stereotypes needed for modeling service-oriented applications. The most important one is a *Service,* which consist of a value offering according to one or more defined *Capabilities*. It provides an *Interface* with *Operations*, with input and output *Parameters* and associated *Types*, and a *ServiceContract* which defines the roles and interfaces involved in the execution of the service. A *ServicesArchitecture* is a UML *Collaboration* which shows the *Participants* of the network or services community, the *ServiceContracts* of the *Services* for interaction between each other, and the roles each one plays within each service.

Participants may be software components, organizations, or systems that provide and use services; services are provided by means of *Service Ports* and requested by means of *Request Ports* which are specializations of UML *Ports*. Services can be modeled with a *ServiceInterface* or simple UML *Interface*s which define the elements needed to interact with the service. The *ServiceContract* defines the interfaces, roles and choreography that participants agree to use to interact within each other. A *ServiceChannel* models the communication between consumers and providers, and the *MessagesType* allows specifying the information exchanged within the operations.

[1] SoaML implementations, http://www.omgwiki.org/SoaML/doku.php

3 Eclipse SoaML Tool

Both Eclipse plug-ins of the Eclipse SoaML Tool were developed as part of a broader research line as mentioned before, the MINERVA framework. The first one focuses on service modeling and the second one on generating code from these SoaML models. Other tools and proposals developed within this wider work to support our vision based on business processes implemented by services are not shown here.

3.1 SoaML Tool Architecture

The Eclipse environment allows the integration of several tools by means of developing plug-ins which can be directly downloaded and installed in the IDE. Along with the flexibility this provides for integrating specific tools, Eclipse is one of the most used IDEs within the software development community, and provides several options for UML modeling and Web Services generation. It provides extension points to which new plug-ins can be "hooked" reusing existing environment implementations. In Fig. 1 the Eclipse Architecture is shown along with the integration with the SoaML plug-ins composing the SoaML Tool.

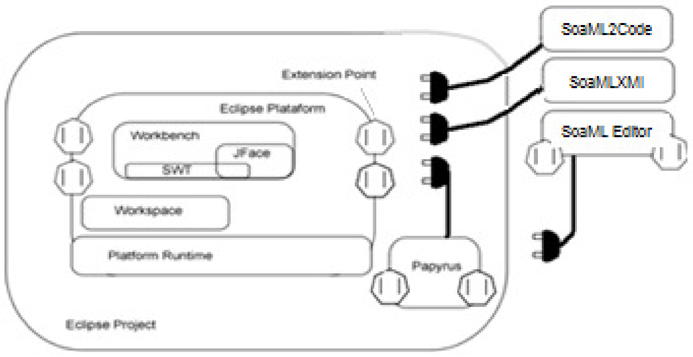


Fig. 1. Eclipse Architecture with extension points and integration of the SoaML Tool

3.2 SoaML editor plug-in

The Eclipse SoaML[2] plug-in for service modeling is developed above the existing Eclipse Papyrus[3] UML 2 modeler, as shown in Fig. 1. Papyrus provides the base UML metamodel and implementation of palette elements, to which we added the SoaML specific extensions and elements. Papyrus is composed of several plug-ins (.jars) implementing the different UML elements and diagrams provided by the editor. For the solution we provide we have reused the code corresponding to the class and composed structures of Papyrus, creating new plug-ins (.jars) for each type of SoaML diagram to be added to the base Papyrus tool.

2 http://www.fing.edu.uy/inco/grupos/coal/en/field.php/Proyectos/EclipseSoaML
3 Papyrus UML, http://www.papyrusuml.org/

SoaML editor functionalities.

The SoaML editor plug-in provides seven diagrams to support the elements defined in the SoaML standard: ServicesArchitecture, Participants (Class and Component), Interfaces, MessagesTypes, ServiceContracts and Capabilities. Fig. 2 shows the options for creating SoaML diagrams where the top ones correspond to UML diagrams from Papyrus and the bottom ones (a) correspond to the SoaML diagrams we have added. In the Model Explorer view from the Papyrus perspective which is shown at the left (b) in Fig. 2, you can see the tree showing the structure and elements of the SoaML model, which are visualized on the diagram layouts to the right (c).

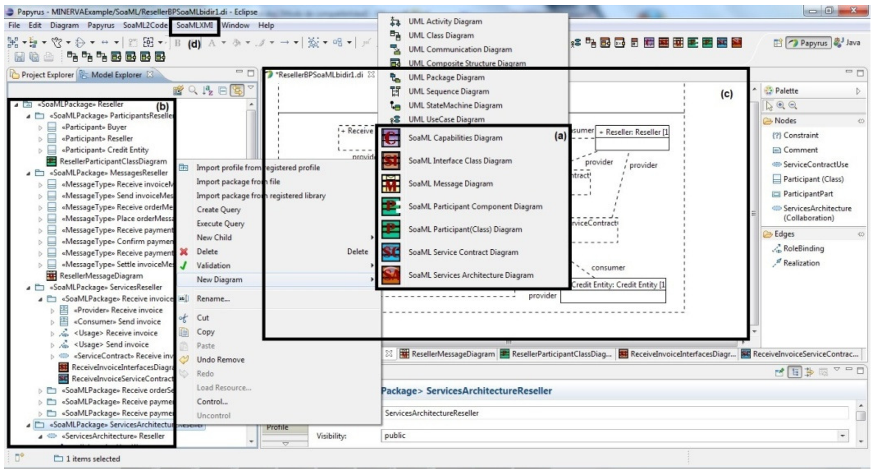

(a)
(b)
(c)
(d)
SoaMLXMI
UML Activity Diagram
UML Class Diagram
UML Communication Diagram
UML Composite Structure Diagram
UML Package Diagram
UML Sequence Diagram
UML StateMachine Diagram
UML UseCase Diagram
SoaML Capabilities Diagram
SoaML Interface Class Diagram
SoaML Message Diagram
SoaML Participant Component Diagram
SoaML Participant(Class) Diagram
SoaML Service Contract Diagram
SoaML Services Architecture Diagram
New Diagram

Fig. 2. Screenshot of the Eclipse SoaML editor showing diagrams and a SoaML model

The Eclipse SoaML plug-in also allows importing and exporting SoaML models in XMI format, to interoperate with other SoaML tools, an option that is provided in the Eclipse menu as shown in Fig. 2 (d). This means that we can import and visualize a service model specified in SoaML with another tool, and we can export a SoaML service model in XMI to be visualized in another tool. Some tools add specific tags that are not included in the SoaML standard (such as MagicDraw[4]) so in these cases the interoperability is affected and the XMI file must be manually "rearranged" to be imported. Interoperability was successfully tested with Modelio early versions[5].

Fig. 3 (a) shows as an example how the SoaML ServicesArchitecture model corresponding to the Reseller business process adapted from [4] is visualized in the SoaML editor. Fig. 3 also shows examples of other SoaML diagrams: (b) Participants showing ports in which services are provided and required, (c) service Interface showing the provider and consumer interfaces with operations, parameters and types for the service "Receive Invoice", and (d) ServiceContract showing the defined roles.

[4] http://www.nomagic.com/products/magicdraw-addons/cameo-soa.html
[5] http://www.modeliosoft.com/en/modelio-store/type/free.html

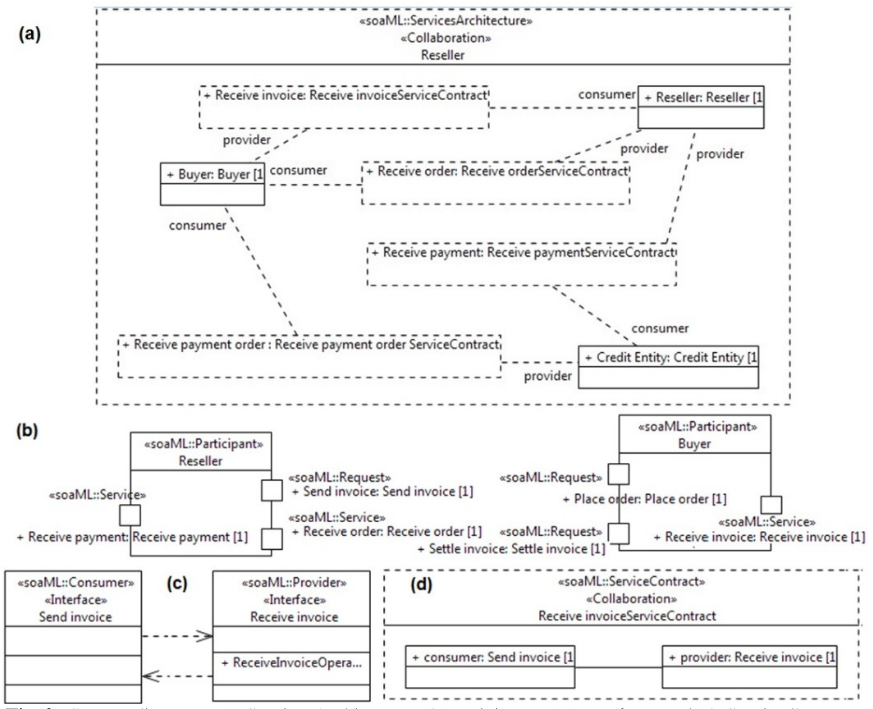

(a)
«soaML::ServicesArchitecture»
«Collaboration»
Reseller
+ Receive invoice: Receive invoiceServiceContract
consumer
+ Reseller: Reseller [1]
provider
+ Buyer: Buyer [1]
consumer
+ Receive order: Receive orderServiceContract
provider
provider
consumer
+ Receive payment: Receive paymentServiceContract
consumer
+ Receive payment order : Receive payment order ServiceContract
+ Credit Entity: Credit Entity [1]
provider
(b)
«soaML::Participant»
Reseller
«soaML::Service»
+ Receive payment: Receive payment [1]
«soaML::Request»
+ Send invoice: Send invoice [1]
«soaML::Service»
+ Receive order: Receive order [1]
«soaML::Participant»
Buyer
«soaML::Request»
+ Place order: Place order [1]
«soaML::Request»
+ Settle invoice: Settle invoice [1]
«soaML::Service»
+ Receive invoice: Receive invoice [1]
«soaML::Consumer»
«Interface»
Send invoice
(c)
«soaML::Provider»
«Interface»
Receive invoice
+ ReceiveInvoiceOpera...
(d)
«soaML::ServiceContract»
«Collaboration»
Receive invoiceServiceContract
+ consumer: Send invoice [1]
+ provider: Receive invoice [1]

Fig. 3. SoaML diagrams: (a) ServicesArchitecture, (b)Participants, (c) Interfaces and (d) ServiceContract

3.3 SoaML2Code plug-in

The Eclipse SoaML2Code[6] plug-in generates specific java projects from SoaML service models, both for service providers and consumers. It requires as input an XMI file containing a SoaML service model, which can be modeled in the same Eclipse SoaML Tool using the SoaML editor plug-in presented above, or another SoaML tool or generated automatically from a business process model in MINERVA framework. Fig. 4 presents the three components that were designed and implemented in order to generate code from SoaML models. It also shows the interaction with the SoaML plug-in or any other tool exporting SoaML models in the XMI or UML formats.

The “SoaML models to Java objects converter” is in charge of parsing the input file with the SoaML model, in order to generate Java objects which represent it. The input file can be an XMI file (version 2.0 or 2.1) or an UML file. The “SoaML models Processor” is in charge of processing the previously generated Java objects and applying to them a set of rules and conditions for invoking the code generators. Currently, the plug-in supports JEE and WS generators, but new generators may be incorporated. The user has different generation options, for example, the type of project to generate (i.e. client or server), the target application server (JBoss AS or Apache

[6] http://www.fing.edu.uy/inco/grupos/coal/en/field.php/Proyectos/EclipseSoaML2Code

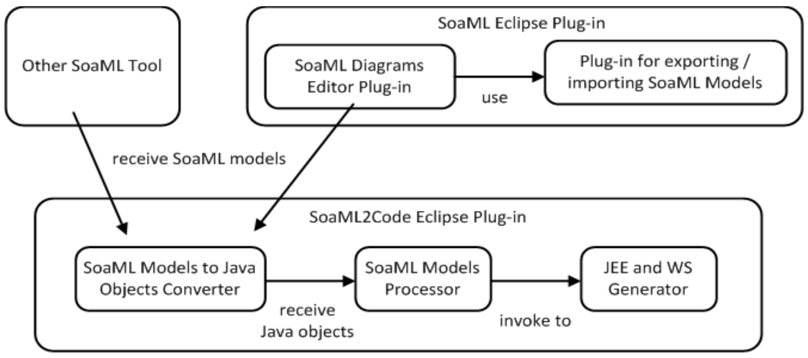

Other SoaML Tool
SoaML Eclipse Plug-in
SoaML Diagrams Editor Plug-in
use
Plug-in for exporting / importing SoaML Models
receive SoaML models
SoaML2Code Eclipse Plug-in
SoaML Models to Java Objects Converter
receive Java objects
SoaML Models Processor
invoke to
JEE and WS Generator

Fig. 4. Components for generating code from SoaML models

Tomcat) and, in case of generating a server project, the type of application which will expose the services (i.e. a JEE application or a Web application). The "JEE and WS Generator" is in charge of generating, based on the SoaML model, the JEE or WS code by invoking the existing generators. Fig. 5 shows the SoaML2Code plug-in, in particular, it shows (a) the available options to generate Java objects from a SoaML model as described above, and on the left side (b) the XMI and Papyrus files corresponding to the SoaML model and layout.

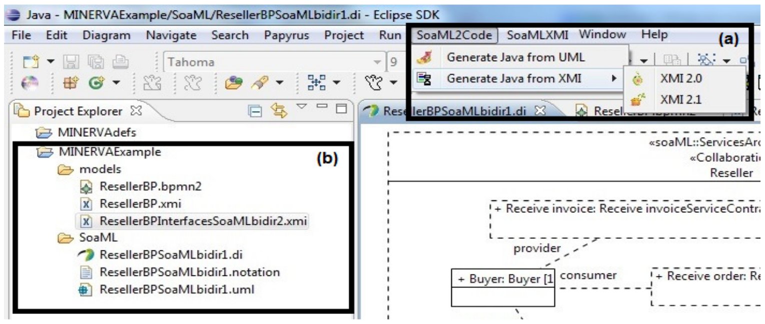
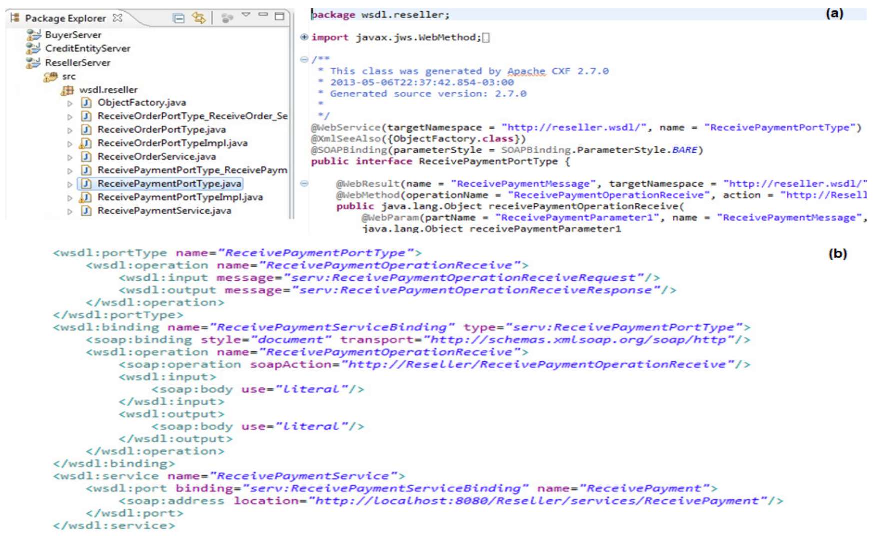

Java - MINERVAExample/SoaML/ResellerBPSoaMLbidir1.di - Eclipse SDK
File Edit Diagram Navigate Search Papyrus Project Run SoaML2Code SoaMLXMI Window Help
(a)
Generate Java from UML
Generate Java from XMI
XMI 2.0
XMI 2.1
Tahoma
Project Explorer
MINERVAdefs
MINERVAExample
(b)
models
ResellerBP.bpmn2
ResellerBP.xmi
ResellerBPInterfacesSoaMLbidir2.xmi
SoaML
ResellerBPSoaMLbidir1.di
ResellerBPSoaMLbidir1.notation
ResellerBPSoaMLbidir1.uml
«soaML::ServicesArc
«Collaborati
Reseller
+ Receive invoice: Receive invoiceServiceContr
provider
+ Buyer: Buyer [1
consumer
+ Receive order: Re

Fig. 5. Eclipse SoaML Tool with the plug-in for code generation from SoaML Models

SoaML2Code functionalities.

From the SoaML diagrams as presented in Fig. 3, the SoaML2Code plugin can be used to generate JEE or WS code by selecting the menu option "Generate Java from JEE" or "Generate Java from UML", as shown in Fig. 5. After selecting one, the generation options (e.g. target application server) are presented in a dialog for the user to choose. For the purpose of this example, we assume that the following options were selected: JBoss AS as the target application server and a Web application to expose the services. Fig. 6 (a), shows on the left a Java project generated for each participant of the SoaML model. Also, for each Port whose interface has the provider role (c.f. Fig. 3), the Java interfaces and classes are generated in order to expose the service as a Web Service. The right side of Fig. 6 (a) shows how the Java code of these interfaces and classes is enriched with several annotations (@WebService, @WebMethod, etc) specified in the JSR 181 (Web Services Metadata Annotations). The SoaML2Code plug-in also generates the WSDL declarations for each of the services. Fig. 6 (b) shows an extract of the WSDL file for the ReceivePayment service.


Package Explorer
BuyerServer
CreditEntityServer
ResellerServer
src
wsdl.reseller
ObjectFactory.java
ReceiveOrderPortType_ReceiveOrder_Se
ReceiveOrderPortType.java
ReceiveOrderPortTypeImpl.java
ReceiveOrderService.java
ReceivePaymentPortType_ReceivePaym
ReceivePaymentPortType.java
ReceivePaymentPortTypeImpl.java
ReceivePaymentService.java
(a)
(b)

```java
package wsdl.reseller;

import javax.jws.WebMethod;

/**
 * This class was generated by Apache CXF 2.7.0
 * 2013-05-06T22:37:42.854-03:00
 * Generated source version: 2.7.0
 *
 */
@WebService(targetNamespace = "http://reseller.wsdl/", name = "ReceivePaymentPortType")
@XmlSeeAlso({ObjectFactory.class})
@SOAPBinding(parameterStyle = SOAPBinding.ParameterStyle.BARE)
public interface ReceivePaymentPortType {

    @WebResult(name = "ReceivePaymentMessage", targetNamespace = "http://reseller.wsdl/"
    @WebMethod(operationName = "ReceivePaymentOperationReceive", action = "http://Resell
    public java.lang.Object receivePaymentOperationReceive(
        @WebParam(partName = "ReceivePaymentParameter1", name = "ReceivePaymentMessage",
        java.lang.Object receivePaymentParameter1
```

```xml
<wsdl:portType name="ReceivePaymentPortType">
    <wsdl:operation name="ReceivePaymentOperationReceive">
        <wsdl:input message="serv:ReceivePaymentOperationReceiveRequest"/>
        <wsdl:output message="serv:ReceivePaymentOperationReceiveResponse"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ReceivePaymentServiceBinding" type="serv:ReceivePaymentPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="ReceivePaymentOperationReceive">
        <soap:operation soapAction="http://Reseller/ReceivePaymentOperationReceive"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="ReceivePaymentService">
    <wsdl:port binding="serv:ReceivePaymentServiceBinding" name="ReceivePayment">
        <soap:address location="http://localhost:8080/Reseller/services/ReceivePayment"/>
    </wsdl:port>
</wsdl:service>
```

Fig. 6. Generated Eclipse projects and Java Code (a) and extract of a WSDL file (b)

4 Related work

To the best of our knowledge there are few implementations of the SoaML standard, mostly commercial ones such as: IBM Rational Architect[7], Visual Paradigm[8], Sparx Systems' Architect[9] and Magic Draw[10]. The last one provides a SoaML editor which can be used together with the open source ModelPro[11] engine to generate code, although ModelPro seems to be a discontinued project. Modelio[12] provides an open source SoaML editor but to be used within their commercial solutions. Our approach is an open source one integrated in the Eclipse environment, which provides easy installation and use within an IDE widely used in the software community, along with existing plug-ins such as Papyrus and WS and Java generators. It was validated by means of case studies based on real service models to support business processes.

7 http://www.ibm.com/developerworks/downloads/r/architect/index.html
8 http://www.visual-paradigm.com/product/vpuml/features/soamlmodeling.jsp
9 http://www.sparxsystems.com.au/products/ea/index.html
10 http://www.nomagic.com/products/magicdraw-addons/cameo-soa.html
11 http://portal.modeldriven.org/project/ModelPro
12 http://www.modeliosoft.com/en/technologies/soa.html

5 Conclusions and future work

We have presented the Eclipse SoaML Tool which provides support for service modeling with SoaML, XMI import and export of SoaML models and generation of code from SoaML models for JEE and WS. We believe it is a useful tool to provide support for service modeling, which is a key aspect in the development of robust SOA applications and for supporting business process execution. Modeling SOAs and automatically generating code from these models, allow registering traceability from business concepts to services and from services to code, easing software maintenance, reuse of services and interchangeability of implementations. We are now working on extending SoaML models with QoS characteristics and generating the related code.

Acknowledgements: This work was partially funded by the Agencia Nacional de Investigación e Innovación (ANII), Uruguay, project ANII-FCE-2011-7341. We would like to thank students Andrés Pastorini, Sofía Larroca, Natalia Maurizio, Victor Fuica and Alejandro Goss for their work within the implementation of the tools.

References

1. Papazoglou, M.; Traverso, P.; Dustdar, S.; Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenge, IEEE Computer Society, (2007)
2. Krafzig, D. Banke, K. Slama, D., Enterprise SOA: Best Practices, Prentice Hall, (2005)
3. Erl, T.,SOA: Concepts, Technology, and Design,Prentice Hall, (2005)
4. Weske, M., BPM Concepts, Languages, Architectures, Springer, (2007)
5. van der Aalst, W.M.P., ter Hofstede, A., Weske, M., Business Process Management: A Survey, In: International Conference on Business Process Management, (2003)
6. Mellor, S., Clark, A., Futagami, T., Model Driven Development - Guest editors introduction, IEEE Computer Society, September/October (2003)
7. Model Driven Architecture (MDA), OMG, http://www.omg.org/mda/specs.htm, (2003)
8. Soa Modeling Language (SoaML), OMG, http://www.omg.org/spec/SoaML/, (2009)
9. Delgado A., Ruiz F., García-Rodríguez de Guzmán I., Piattini M:, MINERVA: Model drIveN and sErvice oRiented framework for the continuous BP improVement & relAted tools, In: 5th Int. Work. Engineering SO Applications (WESOA'09), (2009)
10. Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M.; "Business Process Service Oriented Methodology (BPSOM) with Service generation in SoaML". In 23rd Int. Conf. on Advanced IS Engineering (CAiSE), (2011)
11. Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., "Model Transformations for Business-IT Alignment: From Collaborative BPs to SoaML Service Model", In: 27th ACM Symposium on Applied Computing (SAC), (2012)
12. Delgado; A., Weber, B., Ruiz; F., García; F., Piattini, M., "An integrated approach based on execution measures for the continuous improvement of BPs realized by services", Information and Software Technology Journal, Elsevier, Vol. 56, 2, p. 134–162, (2013-14)
13. Delgado, A, González,L., "Automatic generation of SOAs for Business Process execution: A vision based on models, Latin American Computing Conference (CLEI), (2013)