Semantic Software Engineering Tools

Alexander Paar
Universität Karlsruhe
Am Fasanengarten 5
76128 Karlsruhe, Germany
+49 – 700 – 2539 7227
AlexPaar@ieee.org

ABSTRACT
Recently, the paradigm of software engineering has shifted significantly to service orientation based on Web services. Web Services Description Language interface specifications provide sufficient information to physically access a service. However, these interface descriptions are semantically bleak. This work introduces a number of tools, which were developed to augment strict syntactic service descriptions with semantic information in order to elucidate the meaning of processed data and provided functionality. Semantic Web technologies such as DAML+OIL were supplemented with natural language support for usability improvements both at design- and at runtime.

Categories and Subject Descriptors
D.2.3 [**Software Engineering**]: Coding Tools and Techniques – o*bject-oriented programming, program editors, standards, structured programming, top-down programming.*

General Terms
Documentation, Design, Languages.

Keywords
Semantic software engineering, automatic service lookup and integration, C#, WSDL, DAML+OIL.

1. INTRODUCTION
During the nineties, object orientation of software source code made possible component orientation of applications. Lately, novel component oriented runtime environments have paved the way for service oriented infrastructures. Since there may be a considerable number of service providers, which offer very similar functionality, it tends to be difficult to choose the most appropriate service and to guess the appropriate operations by interpreting syntactic operation names as provided by state of the art Web service interface descriptions. We used Semantic Web technologies like DAML+OIL [1] for constructing ontologies, which present the meaning of processed data and provided functionality. These ontologies were used to annotate both syntactic Web service descriptions as well as object oriented C# source code. This demonstration introduces the set of tools that was developed in order to implement the idea of semantic software engineering. Web Services Description Language [2] documents were annotated with semantic information using a WSDL annotator. A Microsoft Visual Studio [3] add-in was developed in order to annotate C# source code based on a set of DAML ontologies. The Microsoft SOAP toolkit [4, 5] was extended to preserve semantic annotations when C# proxy classes are automatically created from WSDL files as well as when annotated Web service descriptions are automatically generated from annotated C# source code. Annotated Web services may be invoked in a declarative manner both at design- as well as at runtime. Declarative service requests that are embedded with common C# source code are resolved by a Visual Studio add-in. A programmer may even dictate such requests using natural language input. A service activator was developed in order to handle declarative requests issued at runtime.

OOPSLA'03, October 26–30, 2003, Anaheim, California, USA.
ACM 1-58113-751-6/03/0010.

2. ANNOTATION TOOLS

2.1 DAML Annotator
Ontologies representing real-life knowledge may become very extensive. Hence, it is often difficult to look up particular entities. A *DAML Annotator* was developed to supplement DAML descriptions with natural language information as follows.

```
<geography:california rdf:ID="california">
<rdfs:phrasing>California|The Golden State
</rdfs:phrasing></geography:california>
```

The above example shows phrasings for a DAML instance that presents California. Such phrasings are accepted as input for service invocation tools introduced in section 3. Multilingual phrasings could further improve user experience. Future database servers will support such features by introducing full-text queries for synonyms. As a result, a DAML+OIL entity would have to provide only a minimal set of phrasings. The actual vocabulary would be provided by database servers.

2.2 WSDL Annotator
A *WSDL Annotator* was developed in order to augment syntactic interface definitions as provided by WSDL files with semantic meanings. WSDL port types refer to message definitions in the messages section. Each port operation's function signature comprises one input- and one output message. Such a message consists of certain XML types. Using the WSDL Annotator, parameters of Web method signatures may unambiguously be declared as representing particular ontological entities (e.g. an input string may be defined as the name of a US state). Ontological annotations in semantic *SWSDL* files are incorporated as XML tags. SWSDL files may then be referenced by UDDI [6] registries.

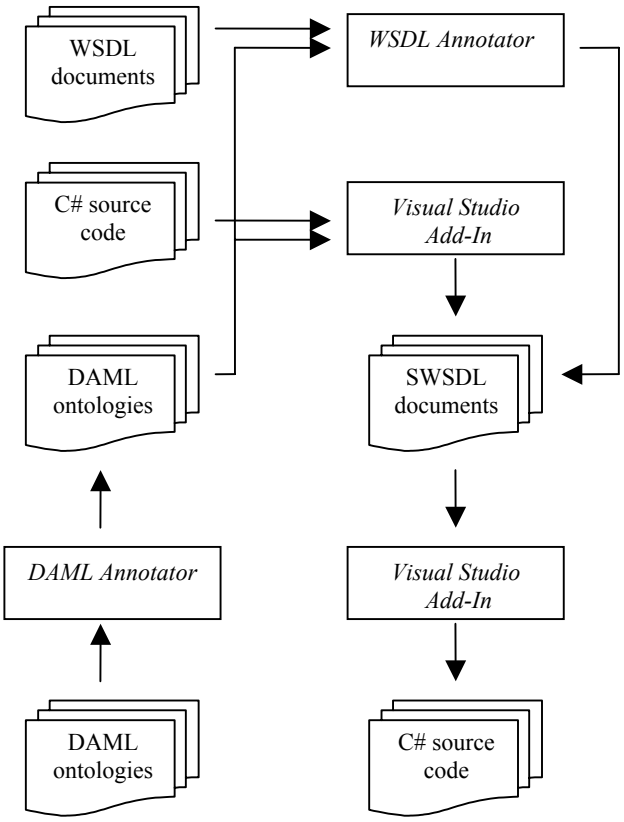

Figure 1. Annotation tools.

2.3 Visual Studio Annotation Add-In

Instead of subsequently enriching existing WSDL descriptions, one may provide semantic type information during programming time, too. The code snippet below shows how declarative programming techniques were employed to annotate C# source code. The *Visual Studio Annotation Add-In* supports roundtrip engineering. SWSDL documents are automatically generated from annotated program source code. C# proxy classes are automatically generated from SWSDL files.

```
public
[DAMLAnnotation("DemographyOntology#Population")]
int GetPopulation
([DAMLAnnotation("GeographyOntology#Country")]
string strCountry) { // function body }
```

Annotated C# Web methods may not only be called with fully qualified DAML identifiers but also with natural language phrasings. During compilation, `DAMLAnnotation` attributes are expanded to code that checks DAML repositories in order to find the canonical DAML ID of a particular instance, which can be processed by the function implementation.SERVICE INVOCATION TOOLS

2.4 Visual Studio Add-In

Annotated Web services that are described by SWSDL interface specifications may be invoked in a declarative manner. The following lines of code depict how such statements may be embedded with normal C# source code. The novel C# keyword `ontology` was introduced in order to permit the use of types in a DAML namespace, such that, one does not have to qualify the use of a type in that namespace

```
ontology "GeographyOntology","DemographyOntology";
p = Invoke(#population?, #state="California");
```

Declarative statements may contain natural language phrasings as introduced in section 2.1. The above service call is dynamically resolved at runtime. It is not bound to a particular WSDL interface description. In especially, it could be resolved to a number of Web services whose parameter namings are totally different.

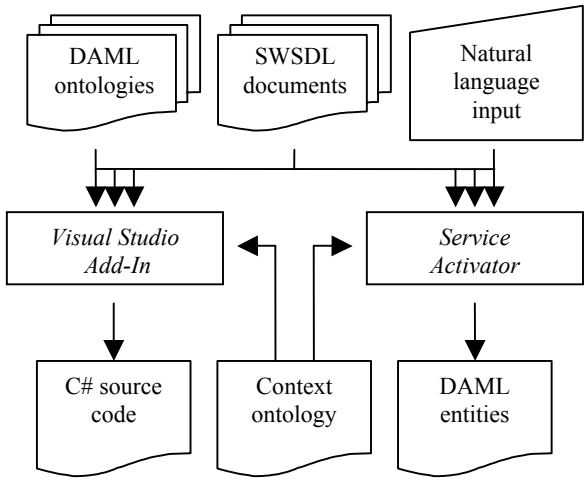

Figure 2. Invocation tools.

2.5 Service Activator

A *Service Activator* was developed as a front-end application for annotated Web services. It may be used to validate and test source code that contains declarative statements. Similar requests may return different results depending on the user's current location. A *Context Server* loads location related ontologies into client applications via TCP/IP remoting techniques. These location aware ontologies are then used to translate natural language input into fully qualified DAML entities.

3. REFERENCES

[1] Defense Advanced Research Projects Agency, The DARPA Agent Markup Language, http://www.daml.org

[2] World Wide Web Consortium, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl

[3] Microsoft Corporation, Microsoft Visual Studio, http://msdn.microsoft.com/vstudio

[4] World Wide Web Consortium, Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP

[5] Microsoft Corporation, Microsoft SOAP Toolkit 3.0, http://msdn.microsoft.com/soap

[6] Oasis Technical Committee, Universal Description, Discovery and Integration of Web Services, http://www.uddi.org