



Search
for:

Use + - () " "

within

[Search help](#)

[IBM home](#) | [Products & services](#) | [Support & downloads](#) | [My account](#)

[IBM developerWorks](#) > [Web services](#)

developerWorks

From UML to BPEL



Model Driven Architecture in a Web services world

Level: Intermediate

[Keith Mantell](#)

IT Architect, IBM

September 9, 2003

The Business Process Execution Language for Web Services (BPEL4WS or BPEL for short) is an XML-based standard for defining how you can combine Web services to implement business processes. It builds upon the Web Services Definition Language (WSDL) and XML Schema Definition (XSD). This article describes a new tool from part of the Emerging Technologies Toolkit version 1.1 (ETTK) released on alphaWorks (see [Resources](#) for a link), which takes processes defined in the Unified Modeling Language (UML) and generates the corresponding BPEL and WSDL files to implement that process. This capability is used to highlight some of the benefits of the Object Management Groups (OMGs) Model Driven Architecture (MDA) initiative: raising the level of abstraction at which development occurs, which, in turn, will deliver greater productivity, better quality, and insulation from underlying changes in technology.

So much XML ...

Application development has undergone a sea change with the advent of Service Oriented Architecture (SOA), incorporating XML-based standards such as WSDL, Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI), and now BPEL. However, while you can do more powerful things, the size and the complexity of the development task has risen, and developers can find themselves losing sight of the main purpose of their work in a welter of files and syntax. In addition, the standards themselves are still evolving, providing something of a moving target for developers. Thus to support the more rapid adoption of Web services, developers are looking for an answer to the problems of complexity, productivity and technological change.

The UML to BPEL mapping tool is able to take models of processes developed in a UML tool, such as IBM Rational's XDE or Rose, and convert them to the correct BPEL and WSDL files necessary to implement that process. The Emerging Technologies Toolkit version 1.1 (ETTK) is an environment for trying out interesting new technologies, and now comes in two flavors: autonomic and webservices. This article focuses on the latter.

What is BPEL?

BPEL provides an XML notation and semantics for specifying business process behavior based on Web Services. A BPEL4WS process is defined in terms of its interactions with partners. A partner may provide services to the process, require

Contents:

[So much XML ...](#)

[What is BPEL?](#)

[Why UML?](#)

[Extending UML](#)

[The UML Profile for Automated Business Processes](#)

[Mapping to BPEL4WS](#)

[The UML to BPEL mapping demonstrator](#)

[Reflections](#)

[Acknowledgements](#)

[Resources](#)

[About the author](#)

[Rate this article](#)

Related content:

[Business processes in a Web services world](#)

[Specification: Business Process Execution](#)

[Language for Web Services, Version 1.0](#)

[Subscribe to the developerWorks newsletter](#)

[developerWorks Toolbox subscription](#)

Also in the Web services

zone:

[Tutorials](#)

[Tools and products](#)

[Articles](#)

services from the process, or participate in a two-way interaction with the process.

Thus BPEL orchestrates Web Services by specifying the order in which it is meaningful to call a collection of services, and assigns responsibilities for each of the services to partners. You can use it to specify both the public interfaces for the partners and the description of the executable process.

BPEL 1.1 is a recent new version of this specification. In addition to clarifying terminology, it allows finer scoping of variables and the addition of event handlers. For the most recent version of the specification, and an introduction to BPEL, see the first two related content entries.

Why UML?

UML is an OMG standard which provides a visual modeling notation that is valuable for designing and understanding complex systems.

UML has several general advantages: it is the most widely known Object-Oriented (OO) modeling notation, it has a graphical notation which is readily understood, and a rich set of semantics for capturing key features of OO systems. UML is widely used in the development of object-oriented software and has also been used, with customizations, for component-based software, business process modeling, and systems design. This enables the considerable body of UML experience to be applied to the maturing Web services technologies.

Extending UML

The ability to extend or customize UML is essential to MDA; UML can be customized to support the modeling of systems that will be completely or partially deployed to a Web services infrastructure. The scope of this article is mainly centered on stereotypes. Stereotypes are a way of categorizing elements of a model. For instance, if you have a class representing a Customer, you could attach a stereotype of <<entity>> to indicate that it represents a data object (perhaps an Entity Bean). This information can be used to help the readability of the model for a human, and can even be used to change the icon representing the class in a CASE tool such as Rational Rose. In this case, however, you can use it to guide the behavior of a model translator. And remember, you can add stereotypes to most elements in a UML model. You can combine a set of these stereotypes in a Profile. A UML Profile is used to define a specific set of extensions to the base UML in order to represent a particular domain of interest. For instance there are Profiles defined for CORBA and Data Modeling. A profile defines what elements of UML are to be used, how they may be extended, and any *well-formedness rules* to constrain the assembly of the elements.

In the following sections, I'll introduce a UML Profile which supports modeling with a set of semantic constructs that correspond to those in the Business Process Execution Language for Web Services. I'll also describe a mapping to BPEL4WS which can be automated to generate Web services artifacts (BPEL, WSDL, XSD) from a UML model meeting the profile.

The following sections will show actual examples of BPEL, and the UML Profile will highlight the key concepts.

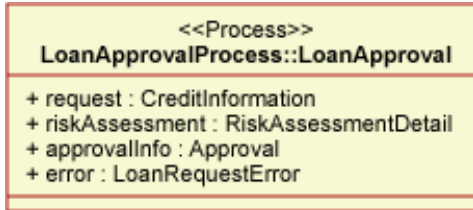
The UML Profile for Automated Business Processes

This section introduces a subset of the UML profile through an example that defines a simple loan approval process. This is the example that you will find in the README file of the ETTK for the transformer. It may be summarized as follows:

"On receiving the loan request, the requested amount is compared to an amount (10000). If the requested amount is lower, then an Assessor service is called, otherwise the Approver service is used. If the Assessor deems the request to be high risk, it is also passed to the Approver. When either the Approver has completed or the Assessor has accepted, the approval information is returned."

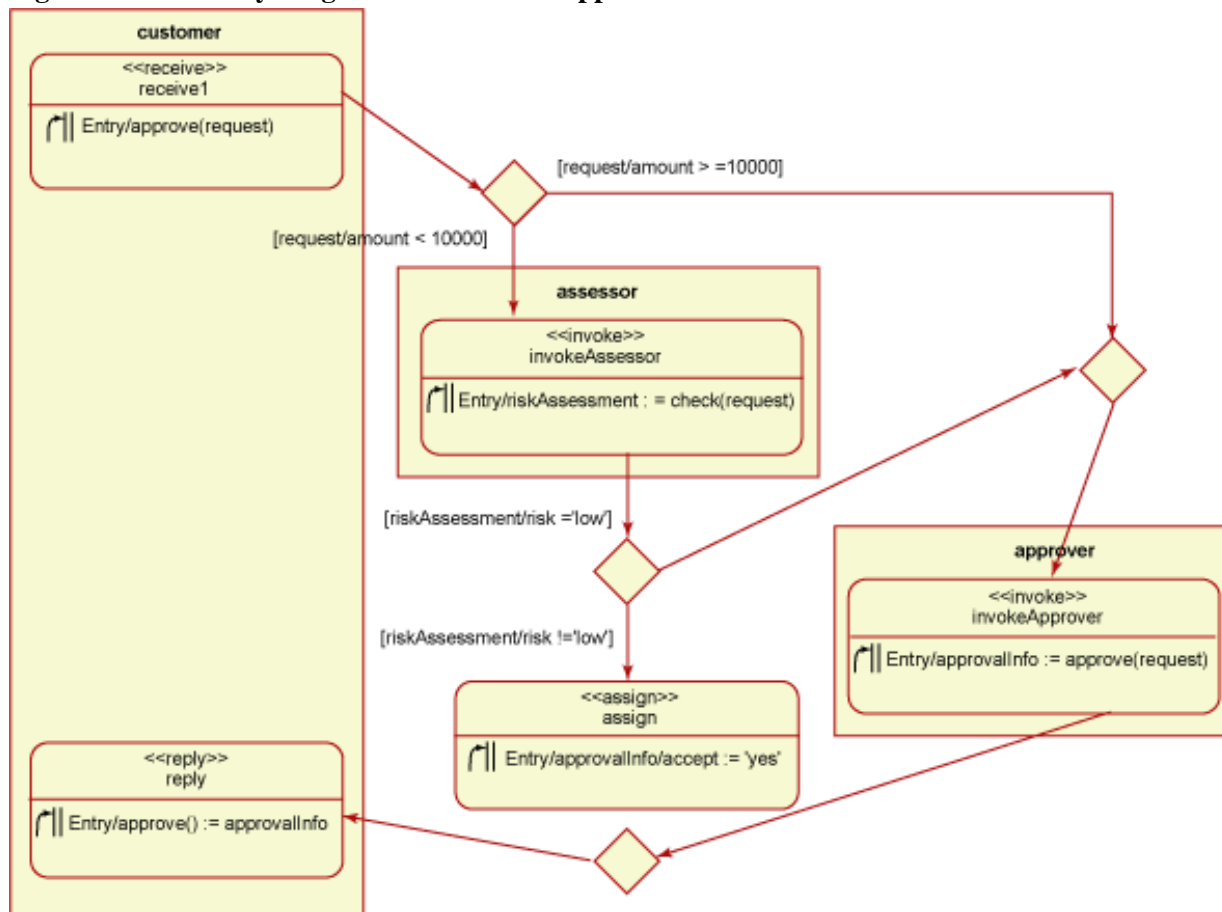
BPEL processes are stateful and have instances, so in BPEL this scenario is implemented as a LoanApproval process which would have an instance for each actual loan application being processed. Each instance has its own state which is captured in BPEL variables. In the UML profile, a process is represented as a class with the stereotype <<Process>>. The attributes of the class correspond to the state of the process (its containers in BPEL4WS 1.0 terminology or variables in BPEL 1.1). The UML class representing the loan approval process is shown in [Figure 1](#).

Figure 1. A UML class used to model a BPEL process



The behavior of the class is described using an activity graph. The activity graph for the loan approval process is shown in [Figure 2](#). The activities, such as `invokeAssessor`, are shown as the rectangles with rounded corners. The actions to be performed are shown as Entry conditions to the activity; for example, `riskAssessment` (a variable) is set to the result of the check service. The partners with which the process communicates are represented by the UML partitions (also known as swimlanes): customer, assessor, and approver. Activities that involve a message send or receive operation to a partner appear in the corresponding partition. The arrows indicate the order in which the process performs the activities. Note that the assignment activity is not in a swimlane; it depicts an action that takes place within the process itself, not requiring an external service.

Figure 2. An Activity Diagram for the Loan Approval Process



The reply activity returns a response back to the customer, completing the execution of the process. Each activity has a descriptive name and an entry action detailing the work performed by the activity.

Mapping to BPEL4WS

The UML profile for automated business processes expresses that complete executable BPEL artifacts can be generated from UML models. [Table 1](#) shows an overview of the mapping from the profile to BPEL covering the subset of the profile introduced in this article.

Table 1. UML to BPEL4WS mapping overview

| Profile Construct | BPEL4WS Concept |
|-----------------------------------------------|-----------------------------------|
| <<process>> class | BPEL process definition |
| Activity graph on a <<process>> class | BPEL activity hierarchy |
| <<process>> class attributes | BPEL variables |
| Hierarchical structure and control flow | BPEL sequence and flow activities |
| <<receive>>, <<reply>>, <<invoke>> activities | BPEL activities |

A cutdown version of the BPEL document that would be generated from the loan approval example in this paper is shown in [Listing 1](#) (much of the detail is omitted here due to space constraints).

Listing 1. Excerpt of the BPEL listing

```

<process name="loanApprovalProcess" ...>

  <variables>
    <variable name="request"
      messageType="loandef:creditInformationMessage" />
    <variable name="riskAssessment"
      messageType="asns:riskAssessmentMessage" />
    ...
  </variables>
  ...

  <flow>

    <receive name="receive1" partner="customer"
      portType="apns:loanApprovalPT"
      operation="approve" variable="request"
      createInstance="yes">
      <source linkName="receive-to-assess"
        transitionCondition=
          "bpws:getVariableData('request', 'amount')<10000"/>
      <source linkName="receive-to-approval"
        transitionCondition=
          "bpws:getVariableData('request', 'amount')>=10000"/>
    </receive>

    <invoke name="invokeAssessor" partner="assessor"
      portType="asns:riskAssessmentPT"
      operation="check"
      inputVariable="request"
      outputVariable="riskAssessment">
      <target linkName="receive-to-assess"/>
      <source linkName="assess-to-setMessage"
        transitionCondition=
          "bpws:getVariableData('riskAssessment', 'risk')='low'"/>
      <source linkName="assess-to-approval"
        transitionCondition=
          "bpws:getVariableData('riskAssessment', 'risk')!='low'"/>
    </invoke>

```

```

<assign name="assign">
  <target linkName="assess-to-setMessage"/>
  <source linkName="setMessage-to-reply"/>
  <copy>
    <from expression="'yes'"/>
    <to variable="approvalInfo" part="accept"/>
  </copy>
</assign>

...

<reply name="reply" partner="customer" portType="apns:loanApprovalPT"
  operation="approve" variable="approvalInfo">
  <target linkName="setMessage-to-reply"/>
  <target linkName="approval-to-reply"/>
</reply>
</flow>
</process>

```

A link to the complete profile can be found in the [Resources](#).

The UML to BPEL mapping demonstrator

A technology demonstrator supporting an end-to-end scenario from a UML tool (such as Rational XDE) through to a BPEL4WS runtime (BPWS4J) is available from IBM alphaWorks as part of the ETTK. The mapping implementation is built as an Eclipse plugin and takes the industry standard file format for exchange of UML models (XMI) as input. BPEL4WS artifacts along with the required WSDL and XSD artifacts are generated.

Let's take a look at the demonstrator. To play along you need to install a few pre-requisites. The details of these can be found on the ETTK page (see [Resources](#) for a download link). In summary you need the following:

- Rose or XDE (I use XDE v2003 in this article)
- Eclipse 2.0+ or WebSphere Studio Application Developer(WSAD) 5.0+
- WebSphere Application Server (WAS) 5.0+ or Apache Tomcat 4.1.24+
- The ETTK itself (which includes BPWS4J).

I will assume that you have a suitable environment for the next part.

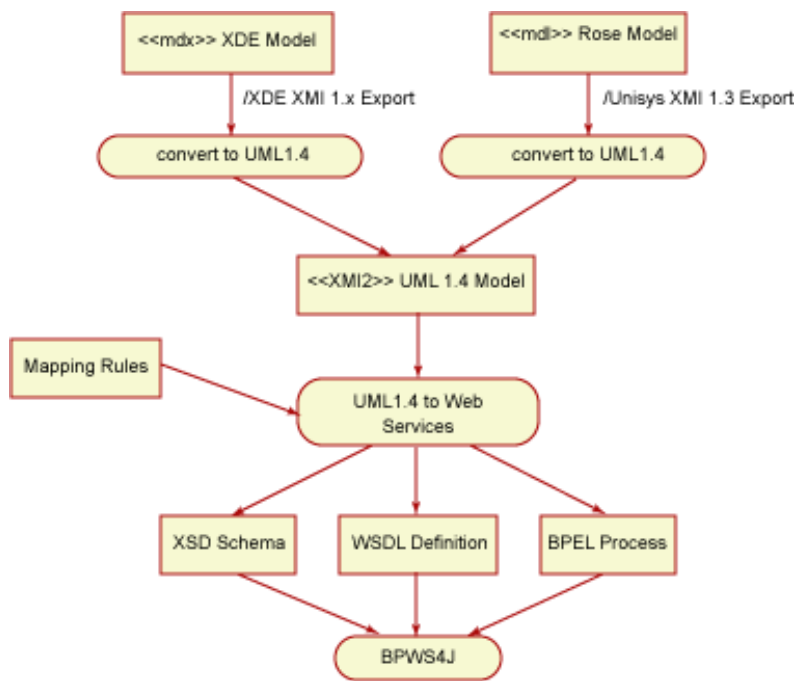
The demonstrator comes with a set of sample files for different scenarios, such as a Loan Approval or Purchase Order process. The sample files are of two main types: UML model files which can be opened and modified with Rose or XDE, and XML files containing the XMI version of the UML models, which are exported by Rose or XDE. In [Figure 3](#) you can see that this corresponds to the models in Rose or XDE themselves, or the XMI output of these tools.

[Figure 3](#) uses a UML Activity Diagram to show the overall process of transforming the files; isn't UML useful?

The boxes represent artifacts (usually files) while the ellipses represent an action or activity. The main stages are:

- Building and exporting the UML model to XMI (using Rose or XDE)
- Generating the BPEL, WSDL, and XSD files
- Deploying and running these on the BPWS4J runtime - trying it out.

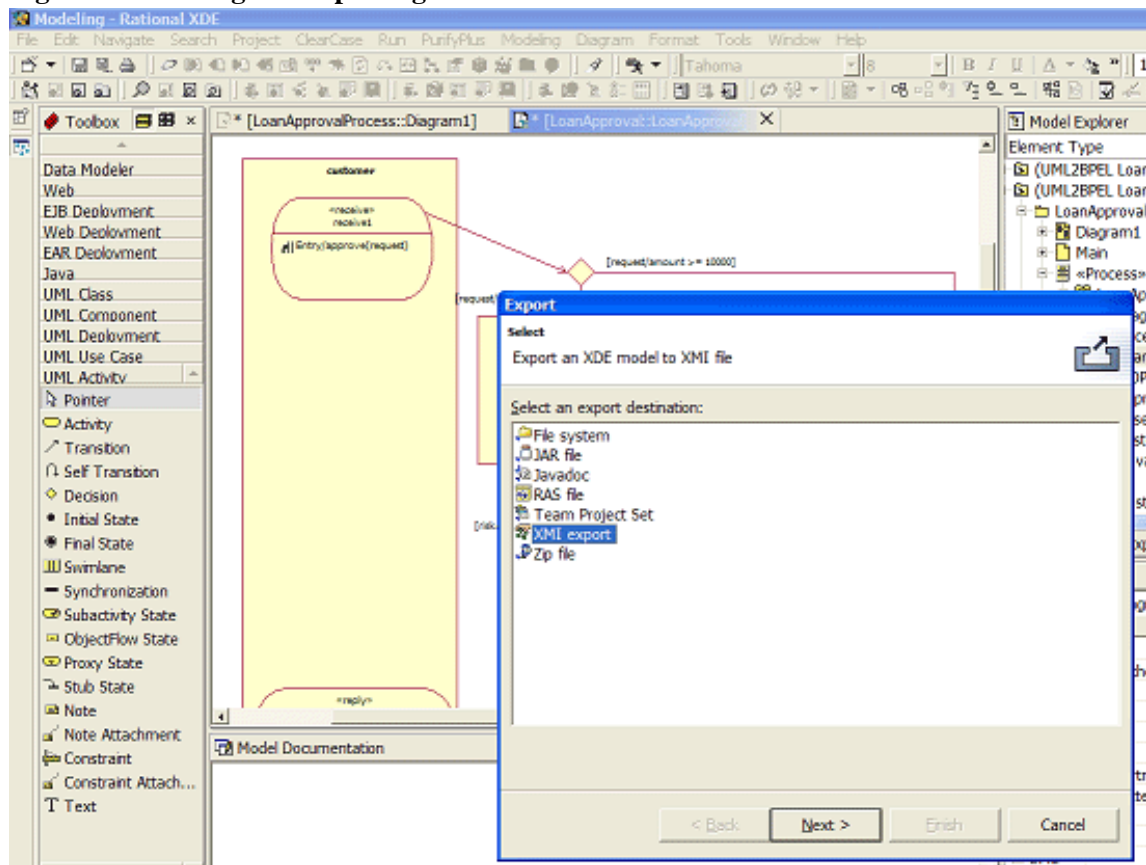
Figure 3. Developing a process



Building and exporting the UML Model

If you have either Rose or XDE you can open one of the UML files (.mdl or .mdx); you have seen examples from the Loan Approval example in this article.

Figure 4. Modeling and exporting with XDE



In [Figure 4](#) the UML model is exported to XMI - in XDE you need v2003 or newer. Note that the beauty of this approach is that you are using a general purpose, unmodified UML tool and UML's standard extension

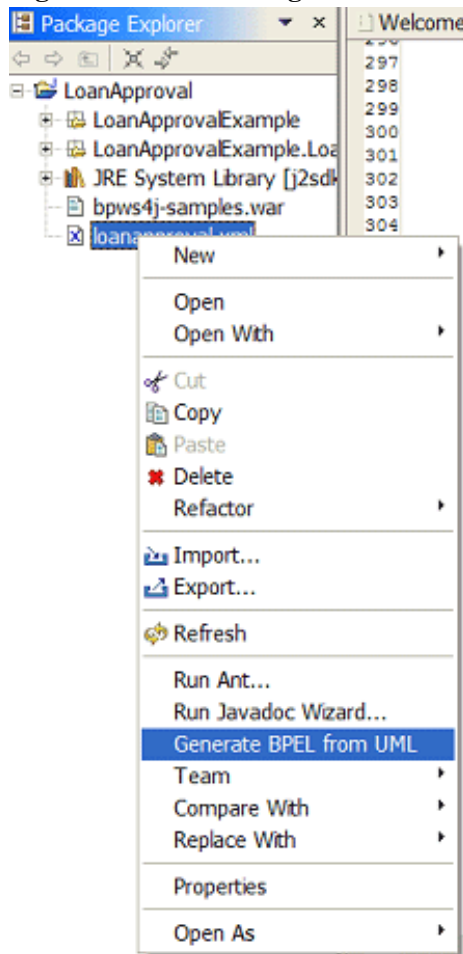
mechanisms to build and export the process definition.

Once the Model is complete, it can be exported from the File --> Export menu, selecting XMI Export, as shown in [Figure 4](#).

Generating the BPEL and WSDL

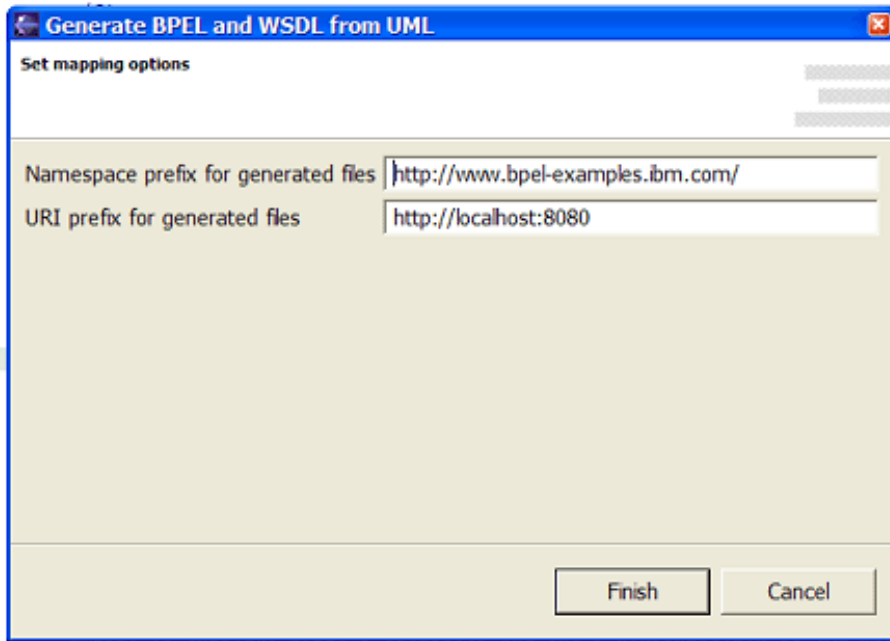
Within Eclipse (or WebSphere Application Developer (IE)), create a Java Project, and import the XMI file you created in the previous section (or use the sample XMI file provided in the ETTK). Then select the XMI file and select "Generate BPEL from UML" from the right-click menu, as shown in [Figure 5](#).

Figure 5. Transforming the XMI file



Next you are offered the chance to change the various prefixes; these should be modified to match the settings for your setup, the defaults are fine if you just use the ETTK install, as shown in [Figure 6](#).

Figure 6. Setting the prefixes



After clicking "Finish," a number of files will appear in the project, including the main BPEL and WSDL files, plus WSDL files for the LoanAssessor and LoanApprover services and the data definitions.

Trying it out

At this point the generated files should be ready to deploy. In this article you deploy to Apache Tomcat, but it is equally possible to use WebSphere Application Server.

Navigate to the BPEL4WS deployment panel (which is <http://localhost:8080/bpws4j/admin/index.html>) and then enter the WSDL file which corresponds to the main services (which is Loan ApprovalProcess.wsdl) and the BPEL file, (which is ProcessOrder.bpel) in the respective fields, as shown in [Figure 7](#).

Figure 7. Deploying the process



After clicking "Continue Deployment", insert any files required by the different roles in the process. In this example there are two extra roles: LoanAssessor and LoanApprover.

Figure 8. Deploying the services

IBM Business Process Execution Language for Web Services Java Runtime

Configure Processes

List

Deploy

Un-deploy

Deploy a Process: Partner Identification

Please enter the name of the WSDL file which corresponds to each of the following partners in the business process.

approver C:\loanapproval\loanapprover.wsdl

assessor C:\loanapproval\loanassessor.wsdl

Once the process has been deployed (see also [Figure 8](#)), then running the sample is the same as for the BPWS4J sample: invoke the LoanApprovalSample script that is appropriate for your OS:

```
LoanApprovalSample [soap-address] first-name last-name amount
```

For example,

```
LoanApprovalSample http://localhost:80/bpws4j/soaprpcrouter John Doe 10
```

From the UML activity diagram, you know that the amount 10000 is significant!

Reflections

This article has introduced a UML profile for automated business processes with a UML to BPEL translator. The profile allows developers to use normal UML skills and tools to develop Web services processes using BPEL4WS. This approach enables service-oriented BPEL4WS components to be incorporated into an overall system design utilizing existing software engineering practices. Additionally, the mapping from UML to BPEL4WS permits a model-driven development approach in which BPEL4WS executable processes can be automatically generated from UML models.

This approach highlights how the notion of MDA can be applied to other areas and at higher levels of abstraction, and insulate the developer from changes in the technology.

There are various scenarios in which this approach will be helpful:

- First, masking changes in versions of the technologies used in a project. BPEL can (and has) changed version. Unless major new semantic changes are introduced which the process wishes to use, the implementation can be regenerated with no other intervention by the developer. These transformation techniques can also be used to protect the developers investment in the original UML models: when a new version of UML is available (for instance moving from version 1.4 to version 2) the UML models themselves can be transformed.
- Second, it is possible to produce a UML profile which can generate other processes implementations such as BPML.
- Third, it shows that artifacts for multiple technologies can be generated from a single model. In this example there was BPEL and WSDL, but non-XML outputs can also be generated, such as Java output.
- Last, the transformations can be bi-directional -- in this example was a reverse mapping to support the import of existing BPEL4WS and WSDL artifacts and the synchronization of UML models and BPEL4WS artifacts with changes in either being reflected in the other.

Acknowledgements

Thanks to Gary Flood, Catherine Griffin, and Tracy Gardner for input and comments on this article.

Resources

- Download the [ETTK](#) from alphaWorks.
- See a [live running install of the ETTK](#) on *developerWorks*.
- Take the tutorial, "[Implementing Web services with the ETTK](#)" by James Snell. (*developerWorks*, April 2003)
- Check out the [MDA](#) resources page at the OMG, which has many articles and pointers to conferences and much more information.
- Browse the [UML](#) resources page at the OMG, which has many articles and pointers to conferences and much more information.
- Find the UML Profile in <ettk install directory>\wstk\services\demos\uml2bpel\docs\UMLProfileForBusinessProcess1.1.pdf.
- Read the [BPEL 1.1 Specification](#) (*developerWorks*, May 2003).
- Get an excellent overview of BPEL4WS in "[Business processes in a Web services world](#)" by Frank Leymann and Dieter Roller (*developerWorks*, August 2002).

About the author

Keith works for IBM at its lovely Hursley Park development site in the UK. He is currently working in the Model Driven Business Integration group. He has worked in application development in many areas including Retail, Banking, and Insurance, often using UML and other techniques to enhance quality, productivity, and communication with stakeholders. You can contact Keith at [keith_mantell at uk.ibm.com](mailto:keith_mantell@uk.ibm.com).



What do you think of this document?

Killer! (5) Good stuff (4) So-so; not bad (3) Needs work (2) Lame! (1)

Comments?