BPMN 2.0 Metamodel Implementation for Eclipse: Get it and Use it

Applies to:

Business Process Modeling, Business Process Management. For more information, visit the Business Process Modeling homepage

Summary

The new BPMN 2.0 specification provides a basis for interoperability between Process Modeling tools and engines from different vendors. For Eclipse as important tool platform, a metamodel implemented on the Eclipse Modeling Framework (EMF) allows to read and write standard compliant BPMN 2.0 files and offers a good API to develop any kind of tools. SAP committed such a metamodel implementation to the Eclipse community. This article explains how you can download and use it.

Author: Reiner Hille-Doering

Company: SAP

Created on: 12 July 2010

Author Bio

Reiner works for SAP since 1997. He has developed several products for Microsoft and Eclipse technology integration and BPM. He is an architect for BPM products, member of OMG task forces and Eclipse committer.

Table of Contents

Introduction

The Business Process Model & Notation (BPMN), a standard for graphically modeling business processes and bringing them to execution, is in its latest version 2.0 a huge step forward for the whole business process management community. BPMN 1.x was quite successful – mainly in the business analyst's community – as it provides easy understandable shapes such as activities, gateways or events, to visualize the most relevant processes running in businesses. However, BPMN 1.x had some limitations and drawbacks that are addressed in BPMN 2.0:

- BPMN 2.0 has a metamodel. A metamodel is a formal way to specify a modeling language. The BPMN metamodel defines all BPMN entities with all their attributes and relations. Entities could be visual shapes like activities or gateways, but also invisible stuff, such as web service operations or data structures.
  BPMN 1.x didn't have a metamodel but only defined the shapes and some of their attributes.
- BPMN 2.0 defines two official file formats that ensure interoperability between tools and engines from different vendors. One file format is based on XML and defined by an XML Schema (XSD) that is part of the BPMN 2.0 standard. The other file format is based on XMI (Extensible Model Interchange), which is typically used by some model repositories. I will explain more details about the two file formats, the technical backgrounds and problems in my next articles.
- BPMN 2.0 defines the execution semantics for each shape, thus nailing down how a process is executed - assuming it is completely defined.
- BPMN 2.0 defines several new shapes or shape refinements, e.g. Human Tasks, Automated Task, Event Subprocesses, Data Stores and much more. Also data flow is fully specified now.
- BPMN 2.0 comes with several new diagram types or extensions of the existing ones, such as Conversations and Choreographies that are useful in some situations, e.g. if the user is more interested, how two processes interact, and not how they work internally.

All in all enough reasons for fast adoption. Let's focus on the second point now: interoperability between tools and engines based on standardized interchange formats. However, for interoperability in practice, it is real implementations that count.

BPMN Metamodel Implementation for Eclipse on MDT

As many BPMN modeling tools including SAP's Process Composer run on the Eclipse platform, it makes sense to implement the BPMN 2.0 on the "Eclipse Modeling Framework" (EMF). EMF provides many useful APIs and feature to work with any kind of structured (meta-) data. As usual in Model Driven Development, the core for each program using EMF is a metamodel, also called "Ecore model".

As mentioned before, the BPMN 2.0 standard already contains a metamodel. However, it comes in the so-called CMOF (Common Meta Object Facility) format that cannot be directly used in Eclipse. Therefore SAP implemented the BPMN 2.0 metamodel on EMF. Using it you get a lot features for free from EMF:

- EMF generates a Java interface and an implementation class for each entity of the BPMN 2.0 metamodel. The generated code provides an easy to use APIs to create and manipulate BPMN models in memory.
- EMF has many built-in services, e.g. to search/query objects, transactions, validation,… .
- EMF also automatically generates a tree-based editor with nice property sheet support. Such an editor is a nice play ground to understand BPMN 2.0 files and is a great diagnostic feature.
- The generated code is a good basis for import/export tools, graphical editor and so forth.
- The generated EMF metamodel has build-in support to read/write models from both official file formats XML and XMI. E.g. you can load any BPMN file into memory and let your code work on it.

Of course also the Eclipse community is aware that an open-source reference implementation of the BPMN 2.0 metamodel makes sense. Therefore the subproject "BPMN2" of Eclipse project "Model Development Tools" (MDT) has already been founded several months ago. The project was quite silent so far, because the

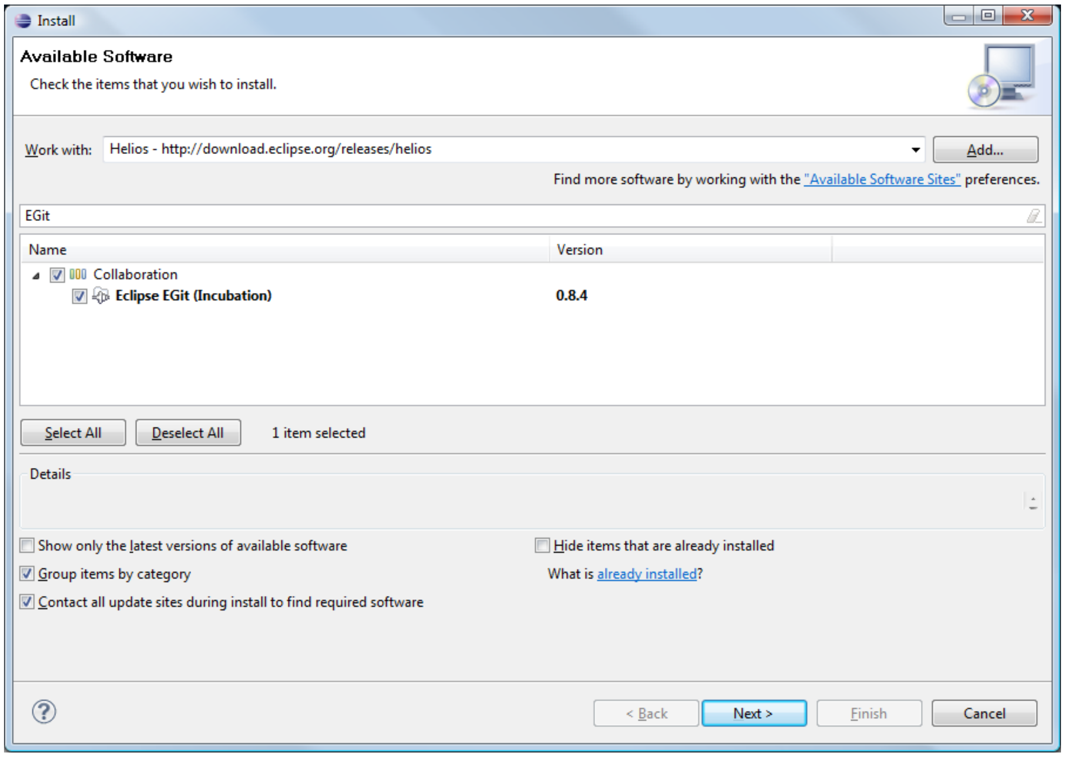
BPMN 2.0 standard was not yet completed. But now, the almost official spec text, metamodel and schema is available; so time has come to fill the MDT/BPMN2 project with life. Quoting from the Wiki of the project:

"The objectives of the BPMN2 component are to provide
- an open source "reference" implementation of the BPMN 2.0 specification
- an EMF-based foundation on which business process modeling tools can be built
- a basis for integrating and interchanging artifacts between business process modeling tools
- a forum for engaging the community in validation of the BPMN 2.0 specification
- an opportunity for increased collaboration between Eclipse and the OMG."

Get it

Now you might want to see our BPMN 2.0 metamodel implementation in action. As the project is quite young, there is neither an Eclipse version available that contains the code nor can you install it via a public update site. Fortunately, you can download the sources with Git (one of the Eclipse source code repositories) and use it in your target Eclipse.

1. Install an Eclipse from http://www.eclipse.org/, best you use a Eclipse Modeling Tools Helios version.
2. Install EGit from the Helios update site. EGit is an Eclipse client for the Git and allows you to download the code from the repository.


Install
Available Software
Check the items that you wish to install.
Work with:
Add...
Find more software by working with the "Available Software Sites" preferences.
Name
Version
Select All
Deselect All
Details
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

Work with: Helios - http://download.eclipse.org/releases/helios

EGit

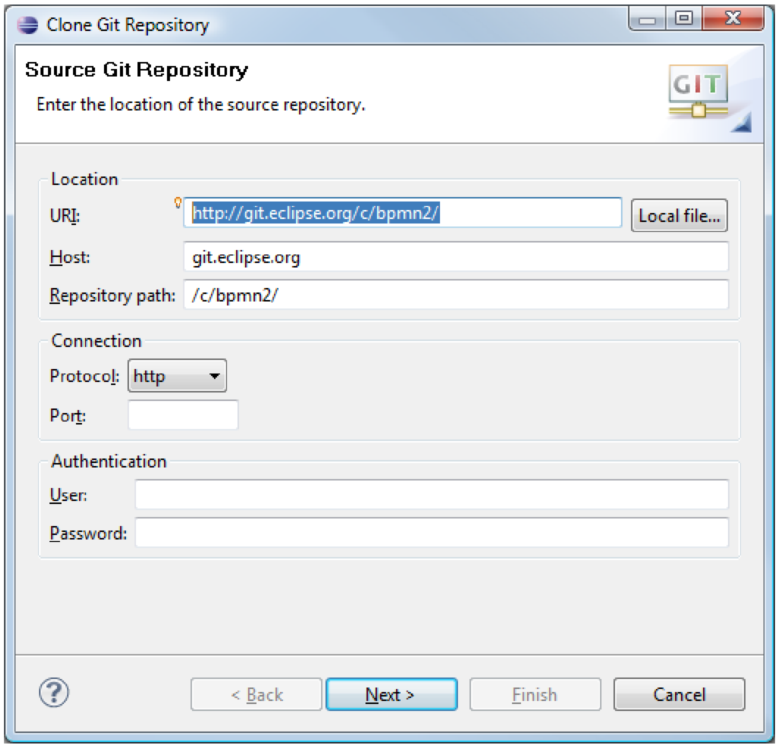
Name	Version
☑ Collaboration	
☑ Eclipse EGit (Incubation)	0.8.4

1 item selected

☐ Show only the latest versions of available software
☑ Group items by category
☑ Contact all update sites during install to find required software
☐ Hide items that are already installed

3. After restarting Eclipse, go to "Git Repository" Perspective, right-click in the "Git Repositories" view and select "Import Git Repository".

4. Enter http://git.eclipse.org/c/bpmn2/ into URI field:



Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI:	http://git.eclipse.org/c/bpmn2/
Host:	git.eclipse.org
Repository path:	/c/bpmn2/

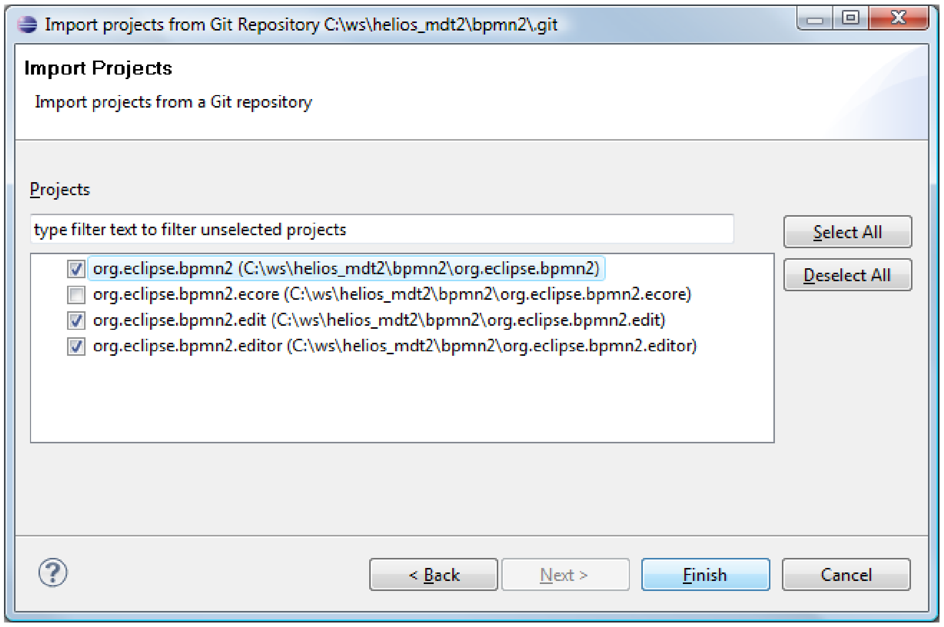
Connection

Protocol:	http
Port:	

Authentication

User:	
Password:	

5. Press “Next” twice and then “Finish”.
6. In the Repository Explorer, right-click “Working directory” and choose “Import Projects…”.
7. In the Wizard click “Next” and then choose all projects except “org.eclipse.bpmn2.ecore” (this one will be deleted soon). Press Finish:



Import projects from Git Repository C:\ws\helios_mdt2\bpmn2\.git

Import Projects

Import projects from a Git repository

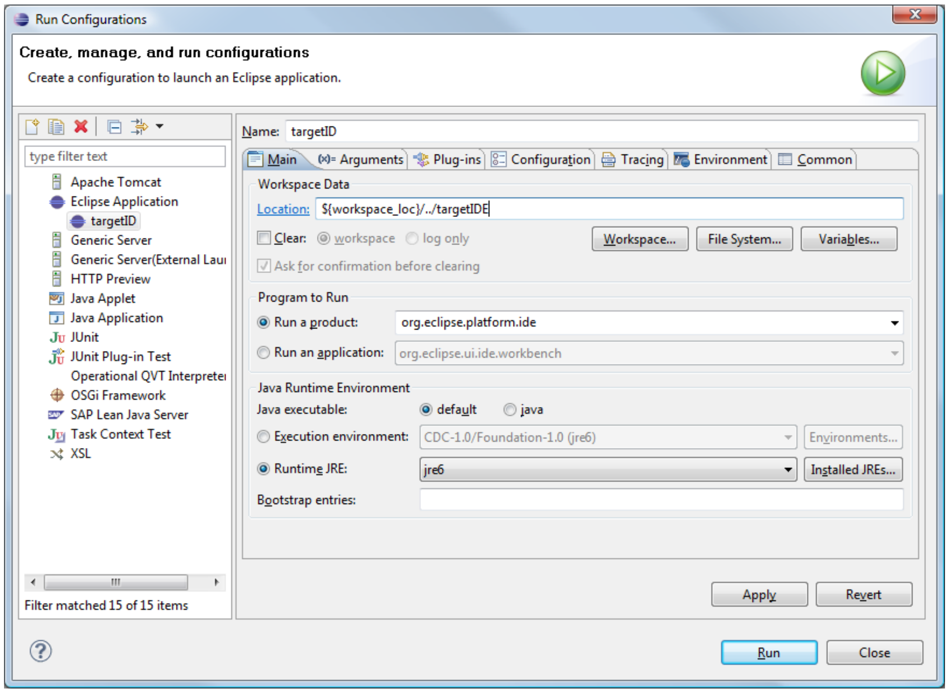
Projects

type filter text to filter unselected projects

- [✓] org.eclipse.bpmn2 (C:\ws\helios_mdt2\bpmn2\org.eclipse.bpmn2)
- [ ] org.eclipse.bpmn2.ecore (C:\ws\helios_mdt2\bpmn2\org.eclipse.bpmn2.ecore)
- [✓] org.eclipse.bpmn2.edit (C:\ws\helios_mdt2\bpmn2\org.eclipse.bpmn2.edit)
- [✓] org.eclipse.bpmn2.editor (C:\ws\helios_mdt2\bpmn2\org.eclipse.bpmn2.editor)

8. If you switch to “Java” or “Plug-In Development” perspective, you should see the 3 plugins without compilation errors.

9. As you want to see the effect of the imported plug-ins in a target IDE, create a new Run configuration (if you don't have one so far), e.g. like this one.



Name: targetID

Workspace Data
Location: ${workspace_loc}/../targetIDE
Clear: workspace log only
Ask for confirmation before clearing

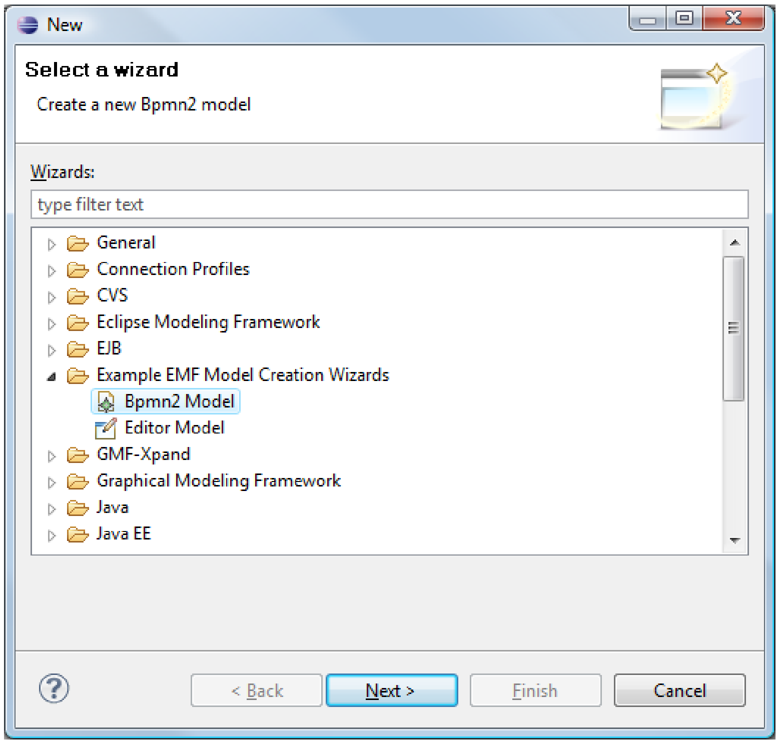
Program to Run
Run a product: org.eclipse.platform.ide
Run an application: org.eclipse.ui.ide.workbench

Java Runtime Environment
Java executable: [✓ default] java
Execution environment: CDC-1.0/Foundation-1.0 (jre6)
Runtime JRE: jre6
Bootstrap entries:

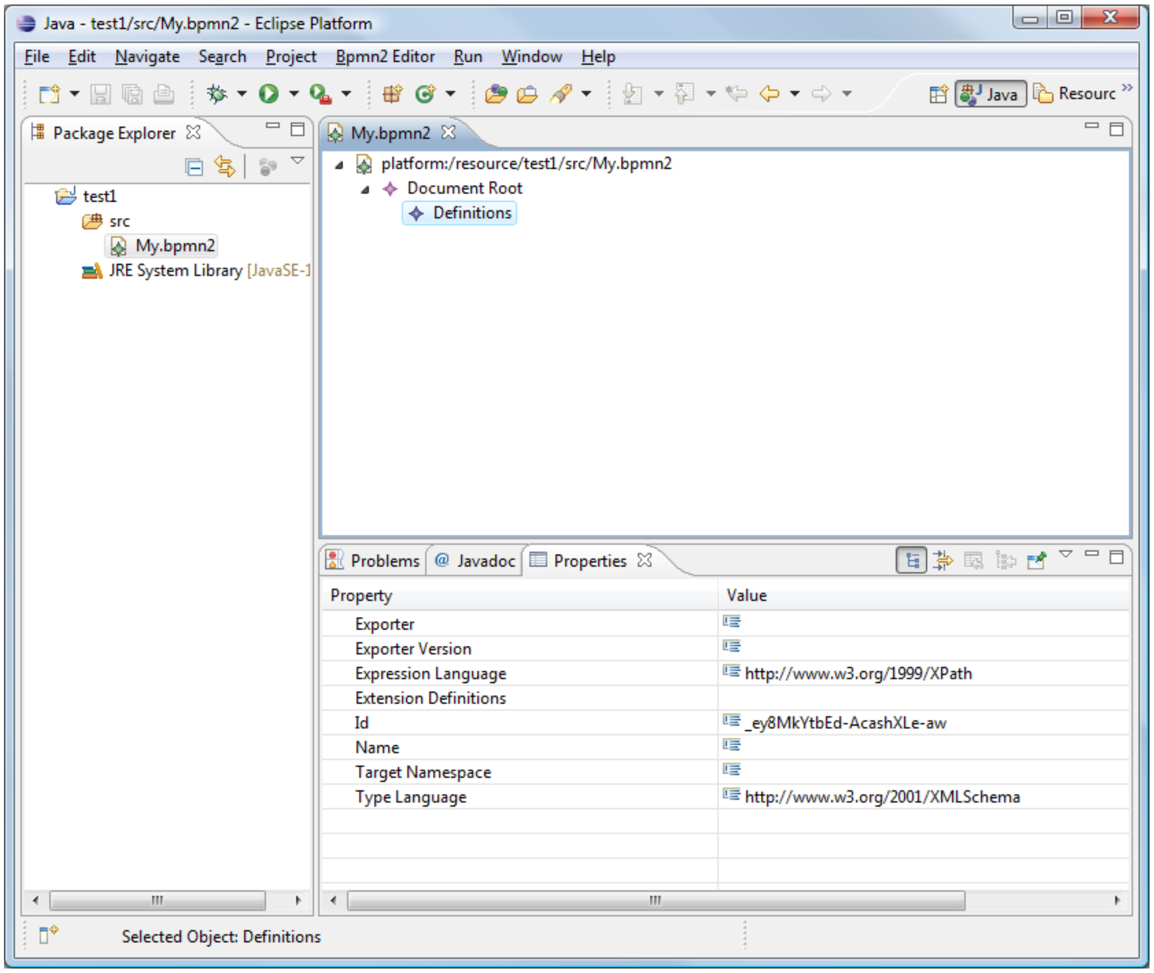
Run it

Of course you want to see what you got. So run your target IDE and create an arbitrary project, e.g. a Java project.

Then right-click a folder and choose "New" → "Other…" (or hit Ctrl+N).



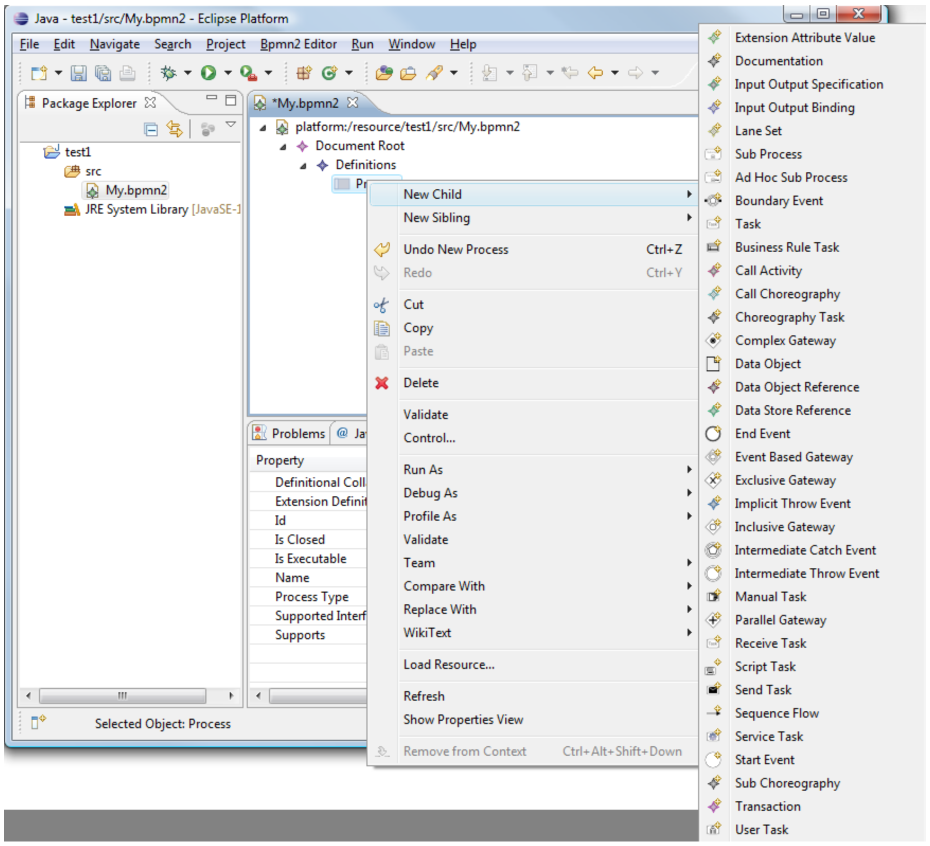
Click "Next" and fill in a name for your new BPMN file, if you like. And you see the simple, tree-based BPMN 2.0 editor.


Java - test1/src/My.bpmn2 - Eclipse Platform
File Edit Navigate Search Project Bpmn2 Editor Run Window Help
Java
Resourc
Package Explorer
test1
src
My.bpmn2
JRE System Library [JavaSE-1
My.bpmn2
platform:/resource/test1/src/My.bpmn2
Document Root
Definitions
Problems
Javadoc
Properties
Selected Object: Definitions

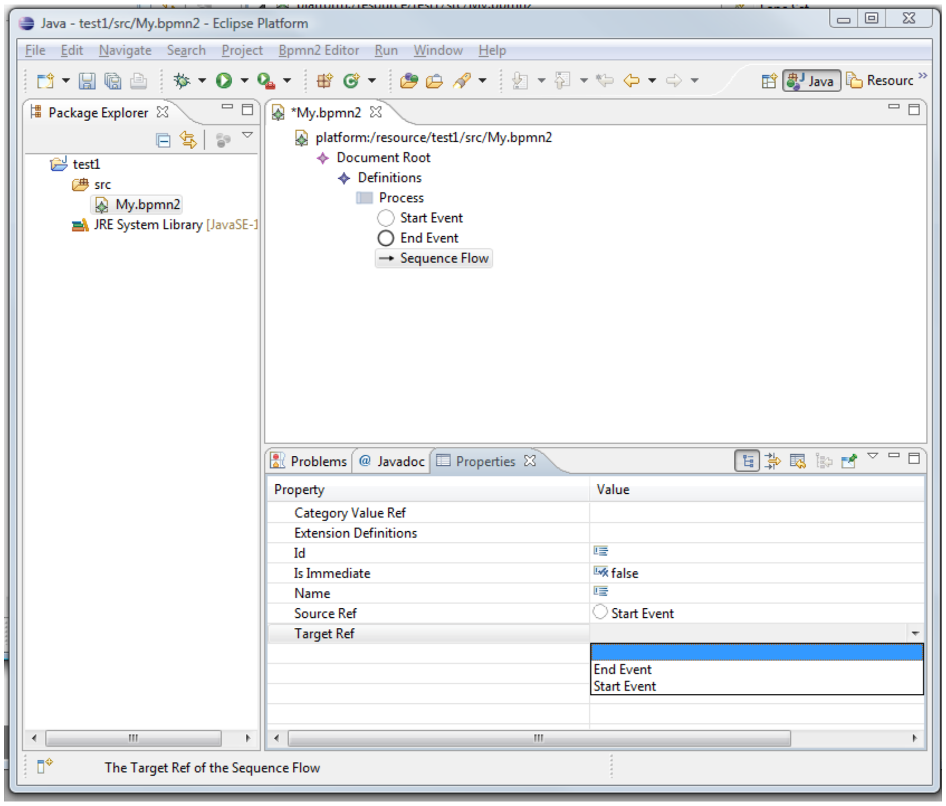
Property	Value
Exporter	
Exporter Version	
Expression Language	http://www.w3.org/1999/XPath
Extension Definitions	
Id	_ey8MkYtbEd-AcashXLe-aw
Name	
Target Namespace	
Type Language	http://www.w3.org/2001/XMLSchema

The new file is automatically populated with a Definitions object, as this is mandatory for BPMN 2.0. In a BPMN 2.0 file all content is (indirectly) contained in the Definitions root object.

You can right-click on the Definitions node and add children based on what the metamodel allows. Note that we already partly have the correct icons for the BPMN elements. The metamodel plug-in contains the BPMN icons in different sizes and even as high-quality SVG vector graphics.

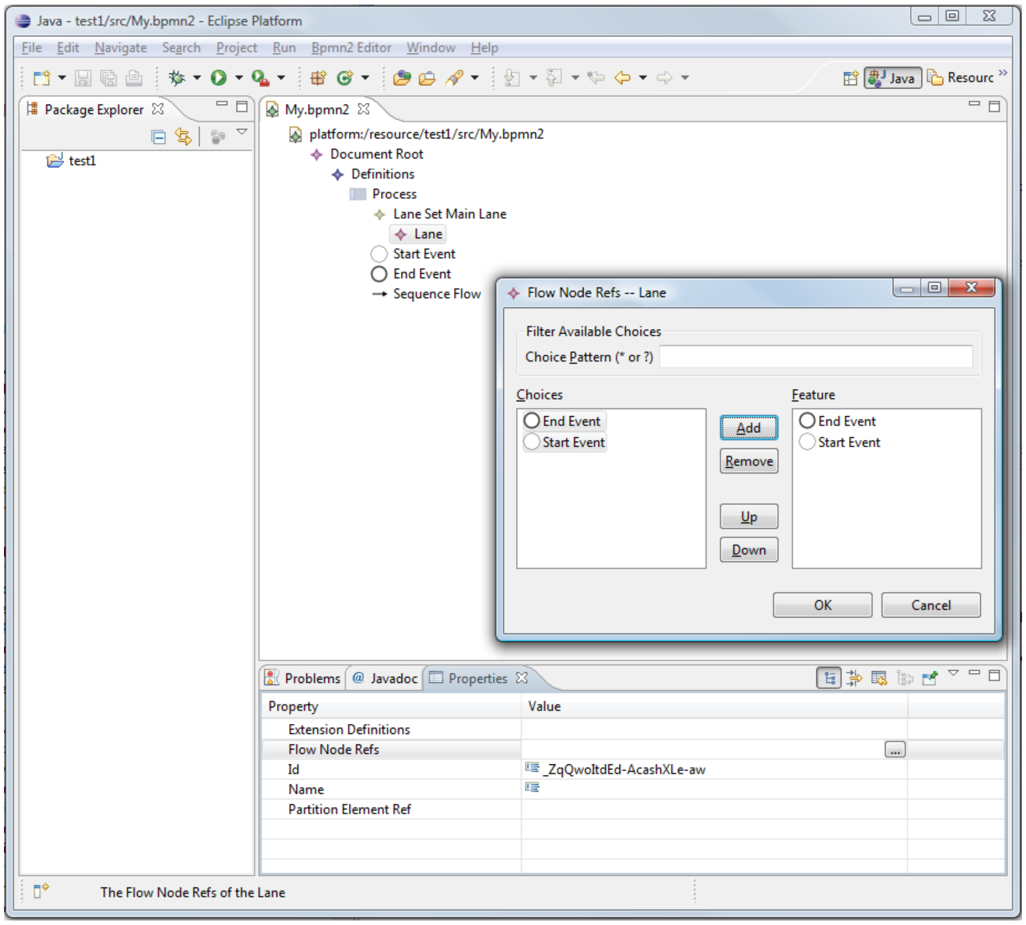


As the EMF metamodel knows all members of each BPMN element and there type information, the editor can nicely help you to fill the properties as show below. E.g. the selected SequenceFlow has two references for source and target node. As they the metamodel specifies that both are of type FlowNode, the property sheet automatically suggests the possible model elements in a drop-down box:


Java - test1/src/My.bpmn2 - Eclipse Platform
File Edit Navigate Search Project Bpmn2 Editor Run Window Help
Java
Resourc
Package Explorer
test1
src
My.bpmn2
JRE System Library [JavaSE-1
*My.bpmn2
platform:/resource/test1/src/My.bpmn2
Document Root
Definitions
Process
Start Event
End Event
Sequence Flow
Problems
Javadoc
Properties
The Target Ref of the Sequence Flow

Property	Value
Category Value Ref	
Extension Definitions	
Id	
Is Immediate	false
Name	
Source Ref	Start Event
Target Ref	
	End Event
	Start Event

EMF Edit also has nice features to populate object collections. E.g. if you want to add Start and End Events to a new lane, create a Lane Set with a Lane and go to the “Flow Node Refs” property editor, again guided by the type information in the metamodel:



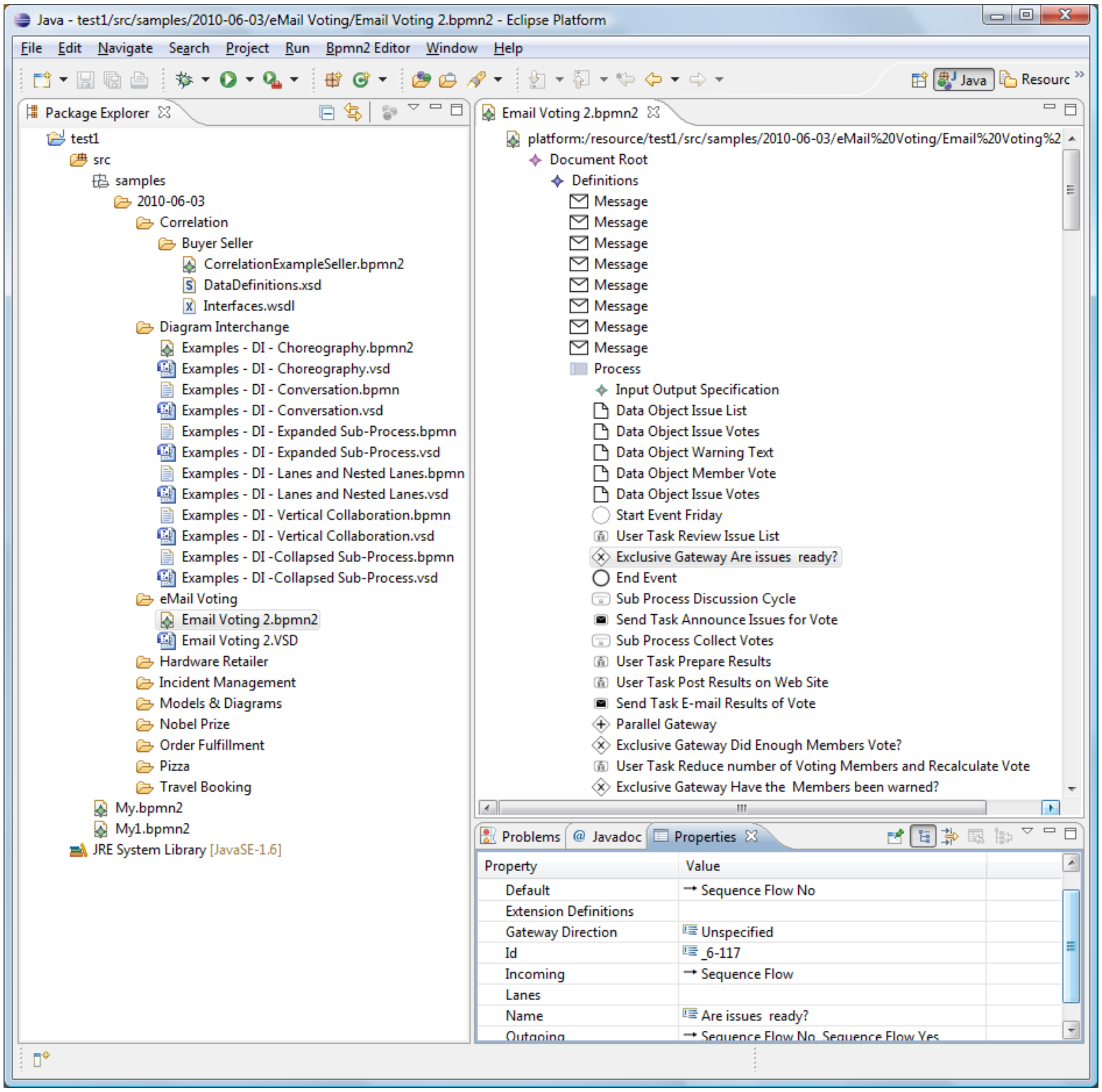
Property	Value
Extension Definitions	
Flow Node Refs	...
Id	_ZqQwoltdEd-AcashXLe-aw
Name	
Partition Element Ref	

Loading Examples

BPMN 2.0 comes with an Examples document. It contains several examples as diagrams and description to explain the BPMN features. The examples document also includes all samples as machine-readable files that comply with the official XML file format. You can easily load them into the EMF editor:

1. Download the ZIP.
2. Extract to a folder in your target IDE's workspace.
3. Rename the file extensions from ".bpmn" to ".bpmn2". (We don't register the ".bpmn" file extension in Eclipse for our editor, as it is frequently used for existing BPMN 1.x tool files that have incompatible file formats.)

Here is one of the most complex examples from the package – the "E-mail voting" – opened in the editor:


Java - test1/src/samples/2010-06-03/eMail Voting/Email Voting 2.bpmn2 - Eclipse Platform
File Edit Navigate Search Project Run Bpmn2 Editor Window Help
Java
Resourc
Package Explorer
Email Voting 2.bpmn2
Problems
Javadoc
Properties

```
test1
  src
    samples
      2010-06-03
        Correlation
          Buyer Seller
            CorrelationExampleSeller.bpmn2
            DataDefinitions.xsd
            Interfaces.wsdl
        Diagram Interchange
          Examples - DI - Choreography.bpmn2
          Examples - DI - Choreography.vsd
          Examples - DI - Conversation.bpmn
          Examples - DI - Conversation.vsd
          Examples - DI - Expanded Sub-Process.bpmn
          Examples - DI - Expanded Sub-Process.vsd
          Examples - DI - Lanes and Nested Lanes.bpmn
          Examples - DI - Lanes and Nested Lanes.vsd
          Examples - DI - Vertical Collaboration.bpmn
          Examples - DI - Vertical Collaboration.vsd
          Examples - DI -Collapsed Sub-Process.bpmn
          Examples - DI -Collapsed Sub-Process.vsd
        eMail Voting
          Email Voting 2.bpmn2
          Email Voting 2.VSD
        Hardware Retailer
        Incident Management
        Models & Diagrams
        Nobel Prize
        Order Fulfillment
        Pizza
        Travel Booking
    My.bpmn2
    My1.bpmn2
  JRE System Library [JavaSE-1.6]
```

```
platform:/resource/test1/src/samples/2010-06-03/eMail%20Voting/Email%20Voting%2
  Document Root
    Definitions
      Message
      Message
      Message
      Message
      Message
      Message
      Message
      Message
      Process
        Input Output Specification
        Data Object Issue List
        Data Object Issue Votes
        Data Object Warning Text
        Data Object Member Vote
        Data Object Issue Votes
        Start Event Friday
        User Task Review Issue List
        Exclusive Gateway Are issues  ready?
        End Event
        Sub Process Discussion Cycle
        Send Task Announce Issues for Vote
        Sub Process Collect Votes
        User Task Prepare Results
        User Task Post Results on Web Site
        Send Task E-mail Results of Vote
        Parallel Gateway
        Exclusive Gateway Did Enough Members Vote?
        User Task Reduce number of Voting Members and Recalculate Vote
        Exclusive Gateway Have the  Members been warned?
```

Property	Value
Default	Sequence Flow No
Extension Definitions	
Gateway Direction	Unspecified
Id	_6-117
Incoming	Sequence Flow
Lanes	
Name	Are issues ready?
Outgoing	Sequence Flow No, Sequence Flow Yes

Code on it

You already have the playground to write code that works with BPMN 2.0 models. You could e.g. write an application that creates BPMN processes automatically, transforms them into another format and much more.

To give you a glance, I have a very small example that creates a process with a Start Event and End Event and saves it as BPMN file. Note that this code does not create a complete, however definitely a correct BPMN file.

The method createAndInitResource() is a little helper that creates a BpmnResource and initializes it with the mandatory DocumentRoot and Definitions objects.

```
public static void main(String[] args) throws IOException {
    BpmnFactory factory = BpmnFactory.eINSTANCE;
    Definitions definitions = new
BpmnResourceFactoryImpl().createAndInitResource(URI.createFileURI("c:/demo.bpmn2"));
    Process simpleProcess = factory.createProcess();
    definitions.getRootElements().add(simpleProcess);
    simpleProcess.getFlowElements().add(factory.createStartEvent());
    simpleProcess.getFlowElements().add(factory.createEndEvent());

    ResourceSet resourceSet = new ResourceSetImpl();
    Resource resource = definitions.eResource();
    resourceSet.getResources().add(resource);
    resource.save(null);
}
```

What's next?

In the next article I will describe how I created the EMF metamodel from two OMG sources using a little merge program. This was necessary, as OMG specifies the metamodel in CMOF (Common Meta Object Facility) and the XML file format in an XML Schema. In the third article I will explain how the EMF BPMN 2.0 Metamodel was used to generate the XSLT scripts that convert between the two official BPMN 2.0 file formats.

For the MDT BPMN2 Eclipse project there is also a lot work left, e.g.

- Complete the set of icons (both as bitmaps and SVG)
- Support for multi-file BPMN models (using the <import> element)
- BPMN 2.0 Model validation

If you have questions, feedback, ideas, or even want to contribute something to the MDT/BPMN2 project, contact me, send a mail to the MDTBPMN2 mailing list, join the MDT news group, or create a bugzilla ticket with product set to “MDT” and component to “BPMN2”.

Related Contents

For more information, visit the Business Process Modeling homepage

Copyright