

RSM 7.5.3 and IFI UML Total for INF5150

v1.2

Contents

- Changes in document from 1.1. to 1.2..... 3
- Changes in document from 1.0 to 1.1..... 3
- Introduction..... 4
 - Important note:..... 4
 - Contact 4
- Installing RSM 5
 - Installation from zipped archives 5
 - Installation via Web download and Installation Manager 5
 - Installing 5
- Installing IFI UML Total..... 5
- Updating IFI UML Total..... 6
- RSM Configuration 6
 - Model Validation settings: 6
 - Appearance settings..... 6
- Preparing your Workspace for PATS and JavaFrame 7
 - SMSPorts 7
 - Resource Jars..... 8
 - Create a new Modeling Project and Import ICU0 8
 - JavaFrame Transformation Configuration..... 9
 - A successfully configured workspace 11
- Transforming, Executing and Testing 12
 - JavaFrame Transformation..... 12
 - Executing 13
- Modeling with RSM 15
 - Creating a new Model Project..... 15
 - Import required libraries into the model 17
 - Create the necessary packages 18

Methodology	18
In the Usecases package.....	18
In the Context package.....	18
In the System package.....	19
General tips	19
Modeling with the JavaFrame Profile.....	19
Root <<Composite>> Class	19
SmsInput and SmsOutput Mediators	21
Tips, Tricks and Work-arounds for modeling in RSM	22
Important information about Composite structures and Connectors.....	22
Visual tips	22
Adding Parts to Composite Structures	22
Adding Activities as Effects in Transitions	23
Adding triggers to Transitions	23
Adding Attributes to Statemachines	24
Multiplicity in Attributes	24

Changes in document from 1.1. to 1.2

- Added a small introduction with an important note: Introduction
- Adding ICU0 importing to: Preparing your Workspace for PATS and JavaFrame
- Updated Modeling with RSM. Added import of model libraries like SMSPorts.
- Added section: Methodology. Step-by-step which elements are wise to create first and in what order.
- Updated: Modeling with the JavaFrame Profile. Complete guide for setting up <<Composite>> classes and the SmsInput and Output Mediators.
- Added section: Tips, Tricks and Work-arounds for modeling in RSM
 - contains answers to questions and feedback/tips from the students.

Changes in document from 1.0 to 1.1

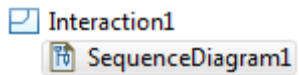
- Added update instructions to IFI UML Total: Updating IFI UML Total
- Modified execution of FakePATS if .jar is not associated with a java binary on the operating system: Executing

Introduction

This guide will take you through the process of installing, updating and using IBM Rational Software Modeler with the IFI UML Total package.

Important note:

Do NOT use RSM's internal sequence diagram editor, ever! Open Sequence Diagrams by right-clicking a sequence diagram element like:



and select **Open Diagram with SeDi**

Contact

For any questions/clarifications about this document or if you find any errors send an e-mail to:

raynerv@ifi.uio.no

with a subject-field starting with **INF5150**:

Installing RSM

Installation from zipped archives

Extract/unzip the downloaded archives

- Run RSM_SETUP/launchpad.exe
- During setup you will be prompted with a warning stating that you must supply one or more disks. These are situated in the zip files supplied.

Installation via Web download and Installation Manager

- Run IBMIM_win32.exe

Installing

- Install IBM® Rational® Software Modeler V7.5
- The installation will first install IBM® Installation Manager which will in turn install RSM.
- Upon restart select Install
- Install the update for Installation Manager
- Again upon restart select Install
- **Click “Check for Other Versions and Extensions” <--- Important**
- Install IBM® Rational® Software Modeler
 - Version **7.5.3**
- When prompted to choose which packages to install keep all selections default
- After the install you will need to convert your Trial license to an Academic permanent license follow this guide:

<http://www-01.ibm.com/support/docview.wss?uid=swg21263473>

The JAR-file that contains the license will be distributed by the course teachers

Installing IFI UML Total

- In RSM select Help → Software Updates... → Available Software
- Click Add Site...
- Add the site:
<http://www.uio.no/studier/emner/matnat/ifi/INF5150/h09/undervisningsmateriale/ifi-uml-total/updatesite/>
- Select all 3 features and Install

Updating IFI UML Total

- In RSM select Help → Software Updates... → Available Software
- Expand the IFI-UML-Update site
 - The update-site have a strange name due to a bug in RSM, e.g. "file:/C:...../IFI UML Update site"
- If "Include items that have already been installed" is not checked you will not see any features in the update-site if no updates are available.
- If a feature is visible then an update is available.
- Select the feature and Install

RSM Configuration

Model Validation settings:

Window → Preferences → Model Validation → Constraints → UML 2.1 → UML 2.1 Specification

- Turn off "Messages cannot cross combined fragment boundaries"
 - Interactions Package → Messages

Appearance settings

Window → Preferences → Modeling → Appearance

- Deselect "Show Gradient"

Preparing your Workspace for PATS and JavaFrame

Download the contents of the Resource folder at:

<http://www.uio.no/studier/emner/matnat/ifi/INF5150/h09/undervisningsmateriale/ifi-uml-total/resources/>

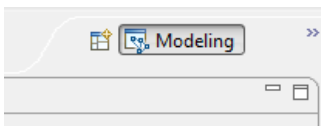
You will need:

- fakepats.jar (**might be updated later**)
- JFDebug.jar (**will be updated later**)
- SMSMediators-XXXXXX.jar
- SMSPorts.emx
- ICU0.emx

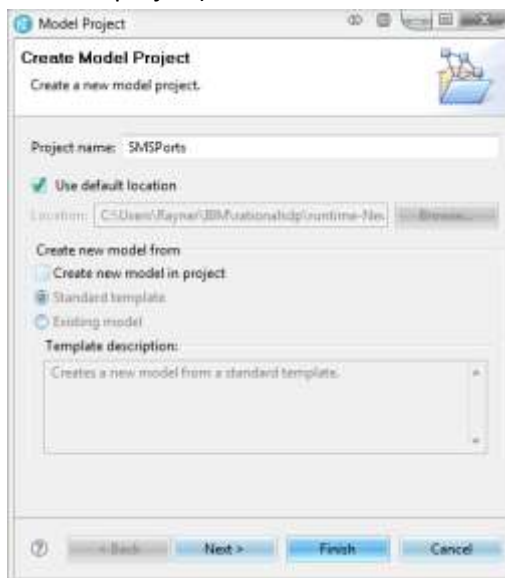
SMSPorts

SMSPorts.emx is a Model Library that we reference in the course modeling projects. It is itself a modeling project and contains specialized classes and signals that are PATS specific.

- Enable the **Modeling** perspective:



- Go File → New → Model Project
- Name the project **SMSPorts** (correct case and spelling is vital) and Deselect “Create new model in project)



- Click Finish
- In the Project Explorer right-click the project SMSPorts and select **Import**
- Select **File System** as import source
- Import the resource file **SMSPorts.emx** downloaded previously

Resource Jars

Several resource jars are required for transformation, compilation, execution and testing of your modeled systems.

FakePATS is a simulated version of Telenors PATS service. Within FakePATS we may create simulated actors that have simulated geographic positions and which can send and receive SMS-messages.

SMSMediators (JAR, not EMX) contains the concrete implementation of the signals and classes defined in the modeling library of SMSMediators (EMX).

JFDebug is both the runtime and a useful debugging tool of JavaFrame systems.

- Go File → New → Project
- Expand the General folder and select **Project**
- Name the project **JavaJars** and click Finish
- In the Project Explorer right-click the project JavaJars and select **Import**
- Select **File System** as import source
- Import the resource files:
 - fakepats.jar
 - JFDebug.jar
 - SMSMediators-XXXXXX.jar

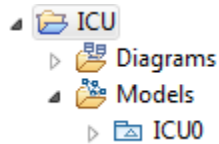
Create a new Modeling Project and Import ICU0

- Enable the **Modeling** perspective
- Go File → New → Model Project
- Name the project **ICU** and deselect "Create new model in project"
- Click Finish
- In the Project Explorer right-click the project ICU and select **Import**
- Select **File System** as import source
- Import the resource file **ICU0.emx** downloaded previously

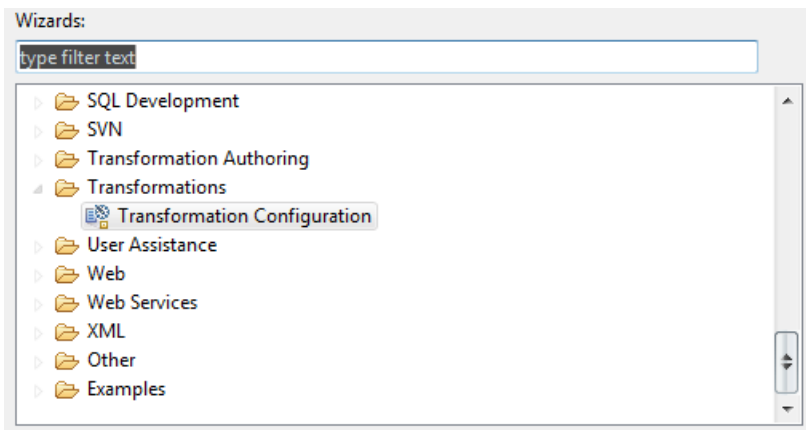
JavaFrame Transformation Configuration

To be able to run a JavaFrame transformation on a model we first need create a **Transformation Configuration** in the project which the model resides.

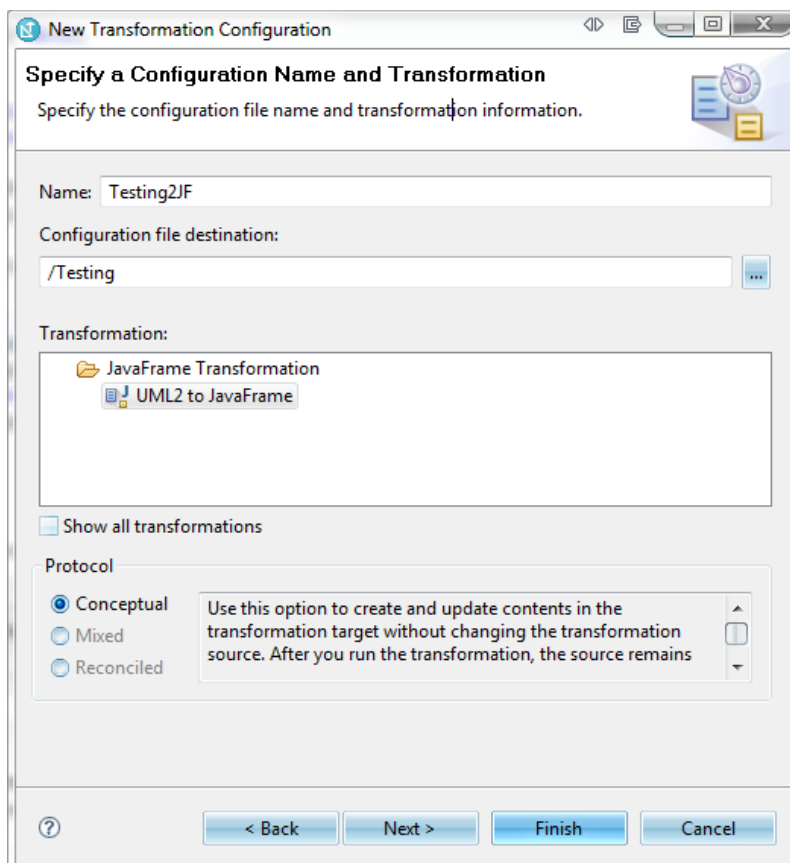
- In the Project Explorer right click the Model Project which you want to create a Transformation Configuration for e.g.:



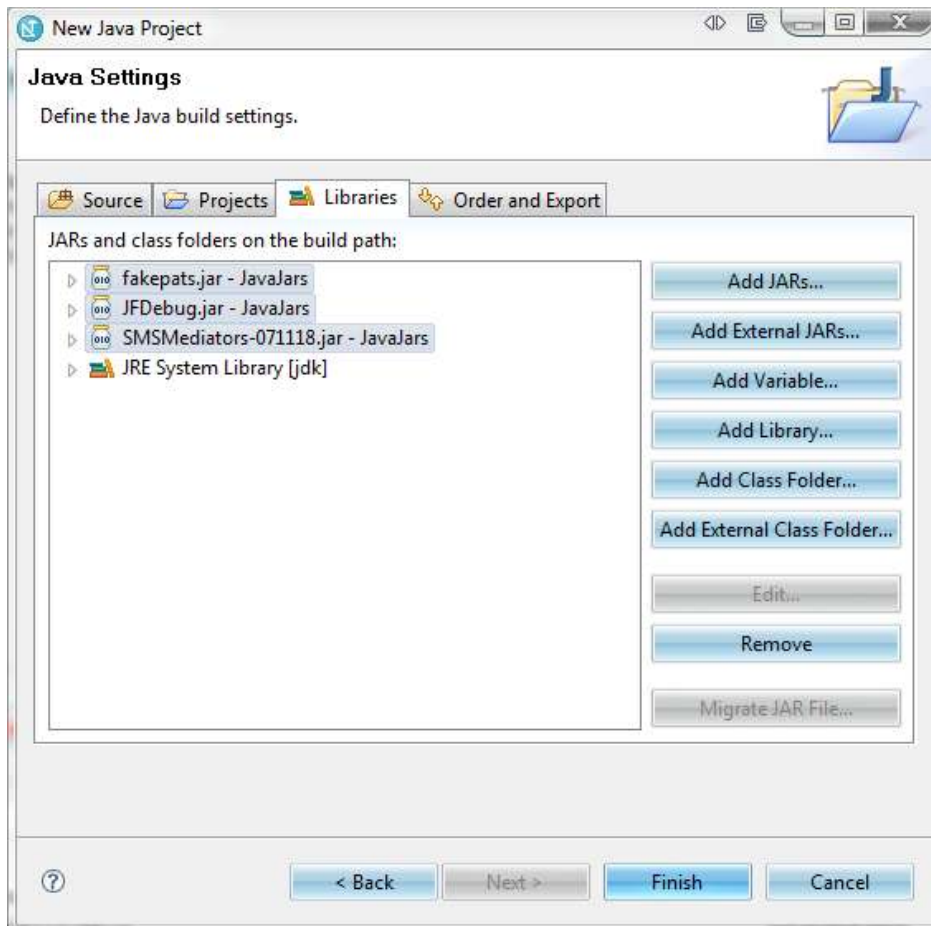
- Click New → Other and select **Transformation Configuration** from the **Transformations** folder:



- Give the configuration a suitable name and select **UML2 to JavaFrame** as the transformation to use:



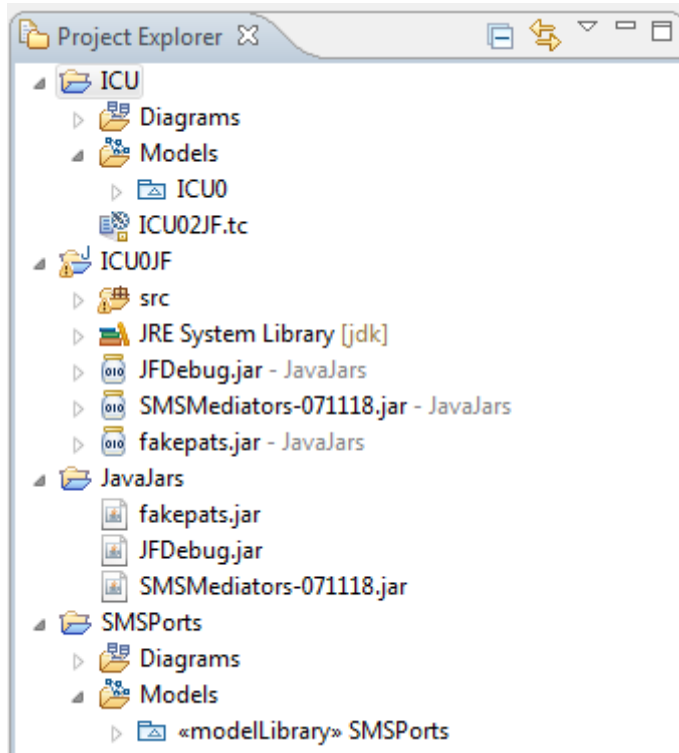
- Click Finish
- The transformation configuration is however not complete. We still need to select a Source and Target for the transformation.
- Create a new Java Project: File → New → Other → Java → Java Project
- Give it a reasonable name and click Next
- You will need to add the following resource jars imported previously into JavaJars to the Build Path of the project: JFDebug.jar, SMSMediators-XXXXXX.jar, fakepats.jar.
- Click the Libraries tab and **Add JARS...** to add the jars



- Click Finish
- Open the Transformation Configuration you create previously inside your Modeling project. The file name ends with **.tc**
- Select the Source and Target tab
- As **Target** select the Java project you have just created.
- You do not need to select a Source at this point.

A successfully configured workspace

In the modeling perspective a successfully configured workspace looks like this:



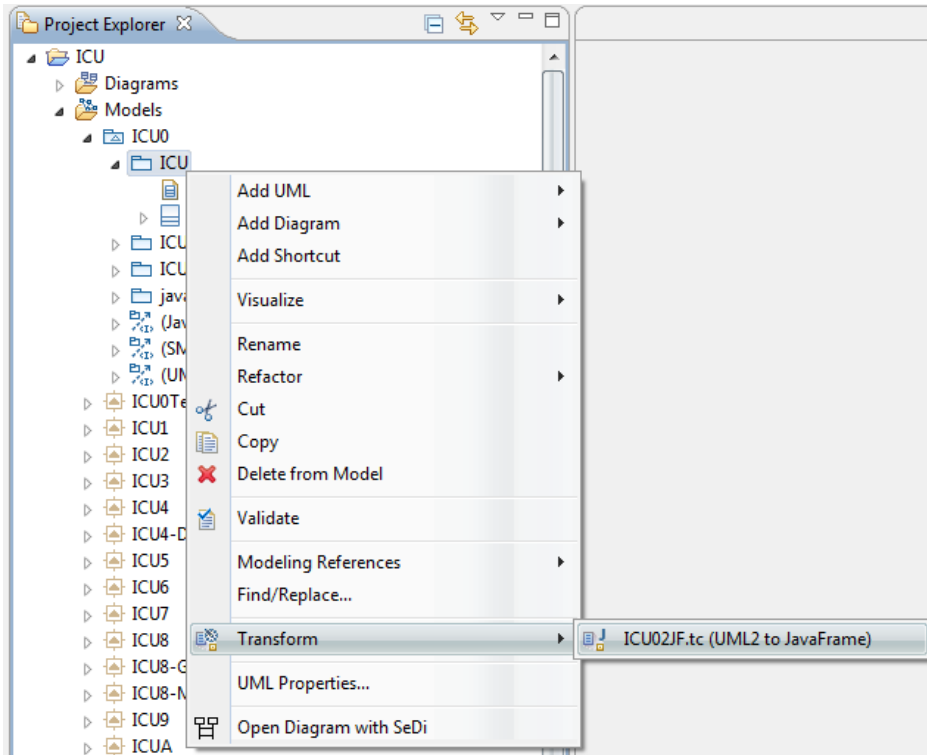
Importantly: Only SMSPorts and JavaJars need those exact names. You may name your own projects as you wish.

Transforming, Executing and Testing

JavaFrame Transformation

How to transform an UML model to JavaFrame:

- Right-click the package that contains your system implementation. Select Transform and click on the transformation configuration you created previously.



A successful transformation completes without warnings. This implies that the model being transformed is well-formed.

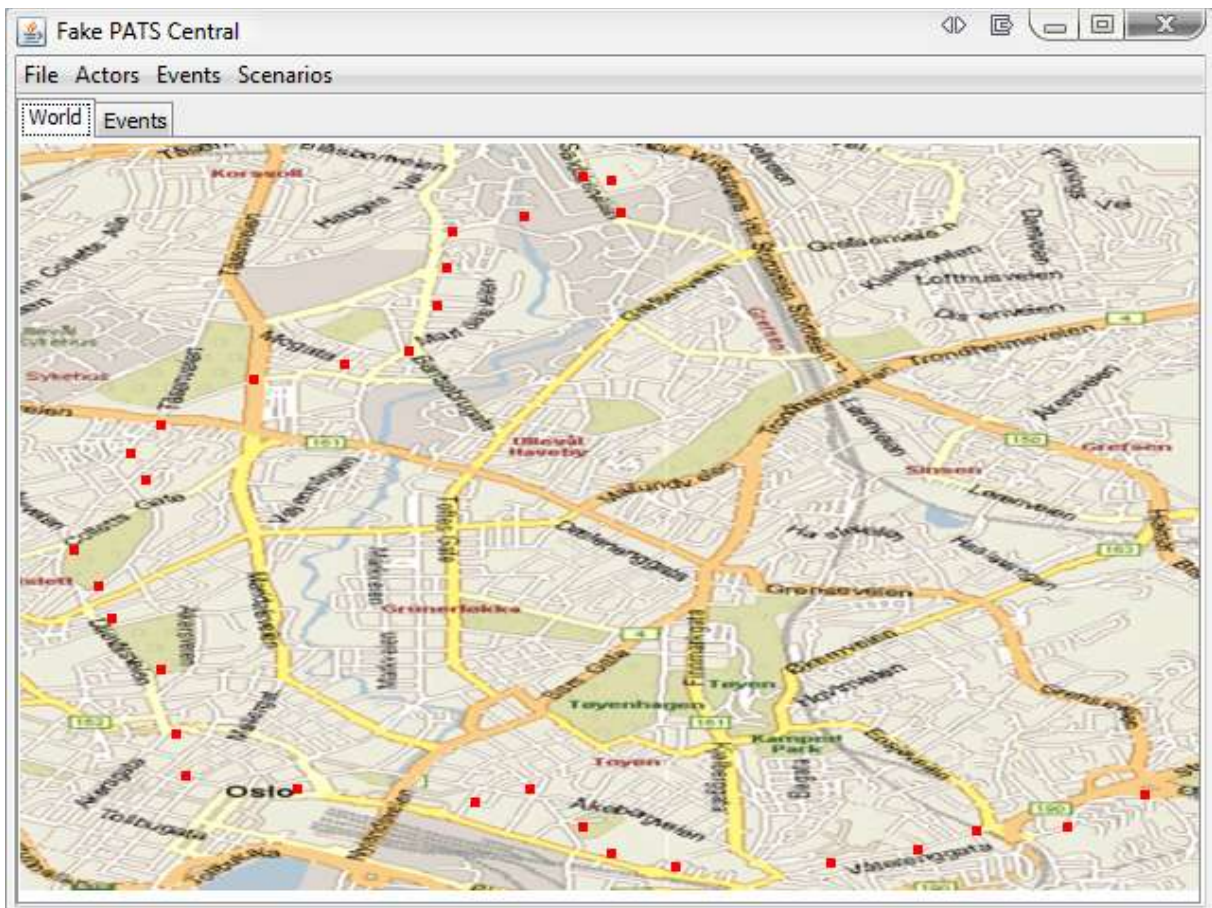
However you may still have errors within the code that has been manually inserted into the diagrams (e.g. Java statements inside OpaqueExpressions):

- Switch to the Java perspective to verify the code that has been generated.

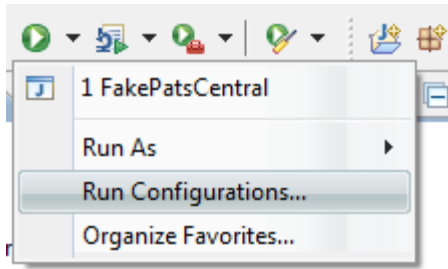
Executing

You will first need to start the FakePATS service:

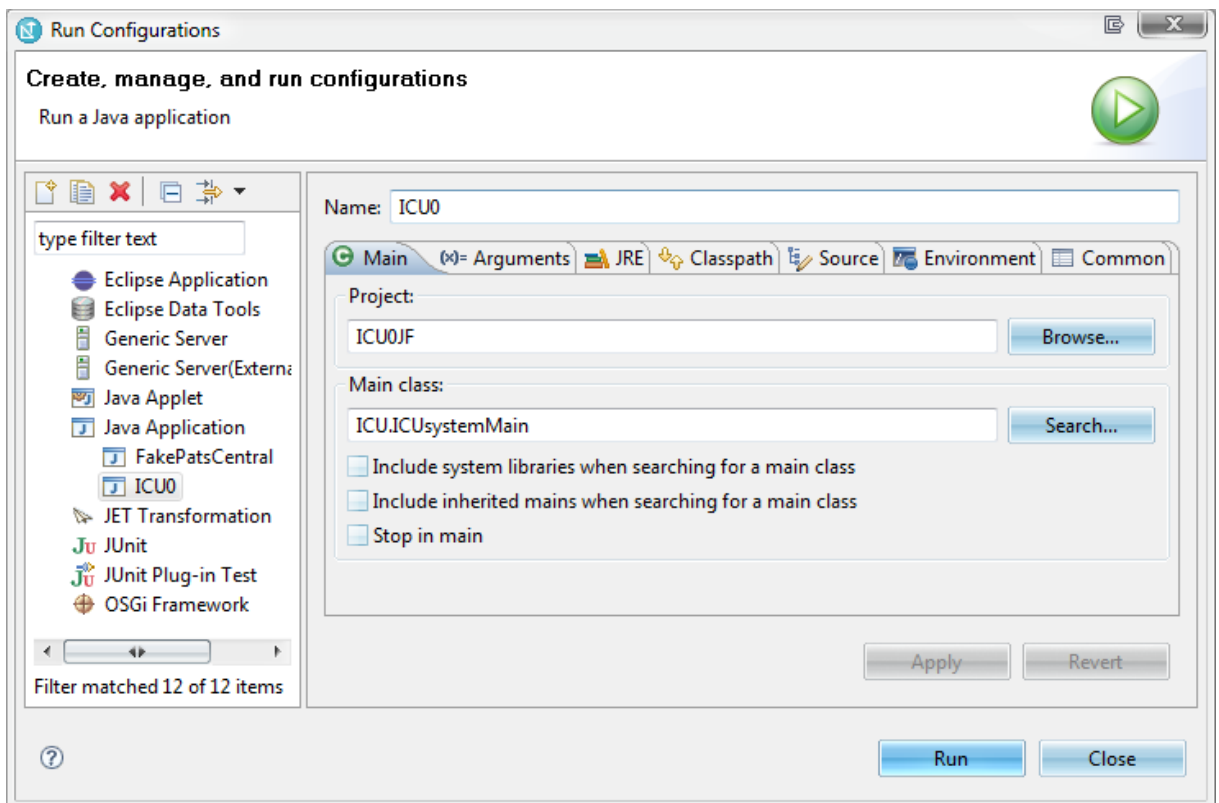
- Right-click the fakepats.jar in your JavaJars folder.
- Select Run as... → Java Application
- FakePATS should now be running:
 - If it does not you need to associate the .jar extension to the Java Platform binary in your operating system.
 - Or you may launch FakePATS as an external tool in eclipse.
 - Under Run → External Tools → External tools configuration... select **Program** and click **New**
 - Name it **FakePATS**, in **Working Directory** click **Browse Workspace** and choose the **JavaJars** project.
 - In **Location** navigate your File System to find javaw.exe. You may find an instance in the installation folder of RSM, e.g. C:\Program Files (x86)\IBM\SDP\jdk\bin\javaw.exe.
 - In **Arguments** type: -jar fakepats.jar
 - Click **Apply** and Run to run FakePATS.



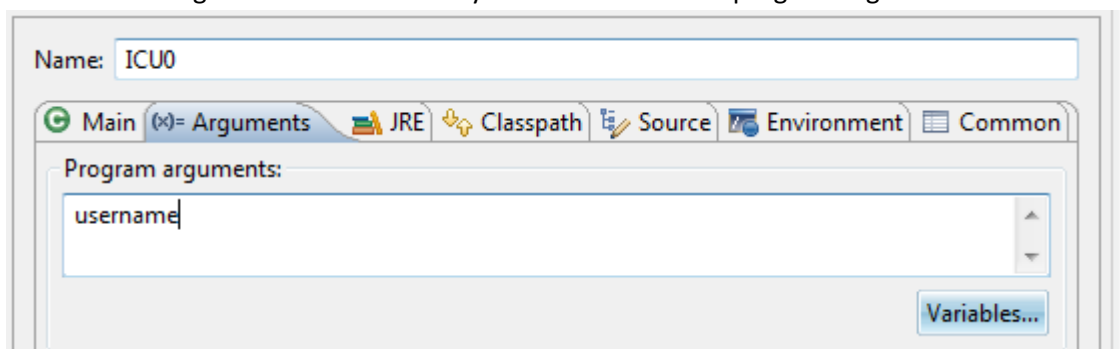
- Switch to the Java perspective
- Click the **Run** button in the menu bar



- Select **Run Configurations...**
- Select **Java Application** and press the **New** button
- Name your **Run Configuration**
- Browse and find your Java project
- Click the **Search** button and select the appropriate **Main class** for your configuration



- Switch to the Arguments tab and write your username in the program arguments:



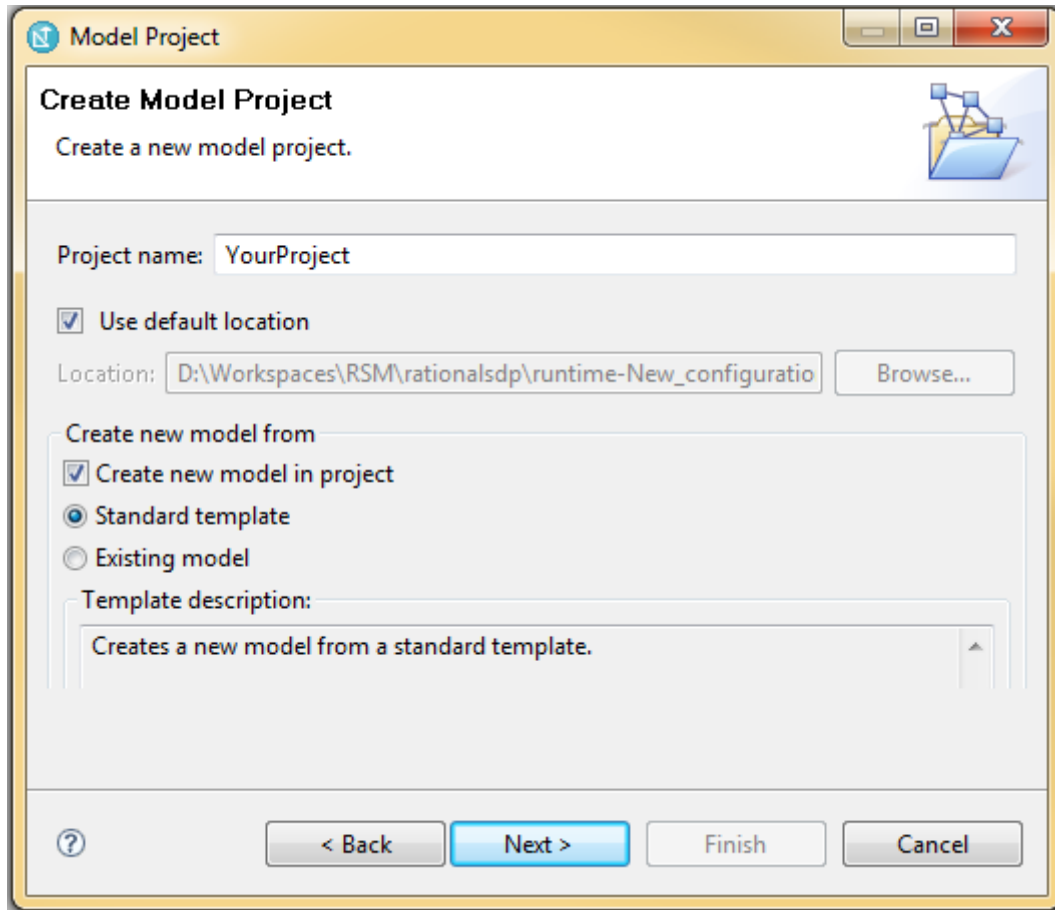
- Click Apply

You may now run your system. Make sure FakePATS is running in the background. For information regarding FakePATS and JFDebug consult their documentation.

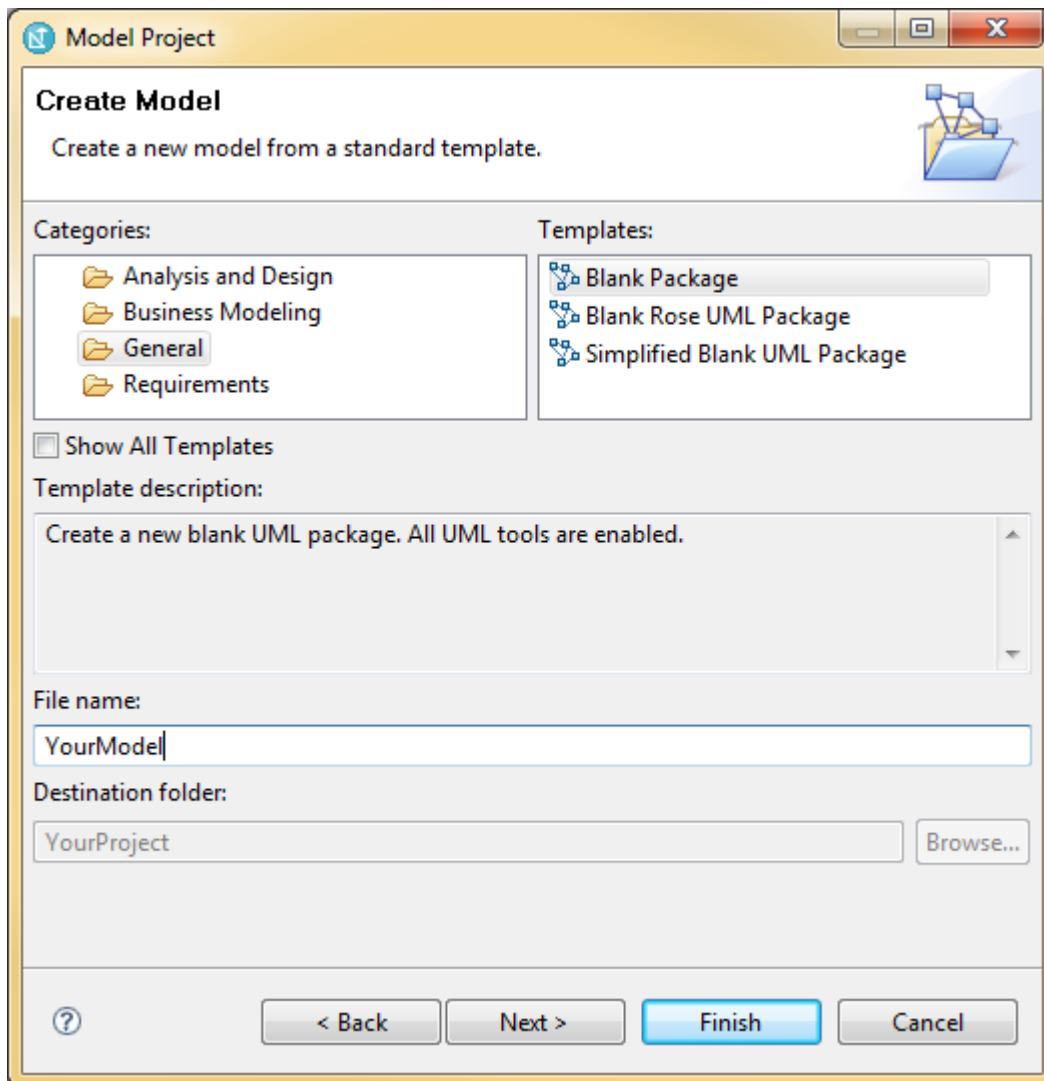
Modeling with RSM

This section describes what you need to do to create a new model project in RSM. JavaFrame specific actions are detailed under: Modeling with the JavaFrame Profile

Creating a new Model Project

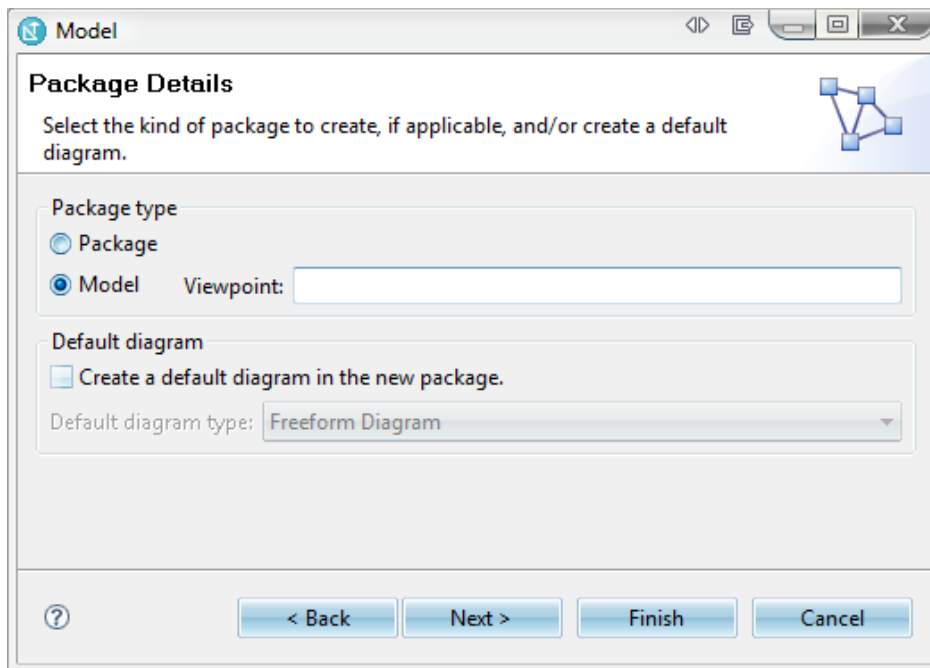


Make sure "Create new Model in project is selected". This will create a fresh model (.emx) inside the Modeling project.



Choose **General**, **Blank Package** as Category and Template.

Name your project in **File name**.



When creating a new Model make sure that you select **Model** as the Package type in the Model creation wizard and not **Package**.

Deselect create a default diagram.

Click **Finish**.

Import required libraries into the model

You will need the same Model libraries as ICU0

- JavaPrimitiveTypes
 - SMSPorts
 - UMLPrimitiveTypes
- ▶ ICU0
 - ▶ ICU
 - ▶ ICUcontext
 - ▶ ICUusecases
 - ▶ java
 - ▶ (JavaPrimitiveTypes)
 - ▶ (SMSPorts)
 - ▶ (UMLPrimitiveTypes)

By comparison your new model only contains **UMLPrimitiveTypes**

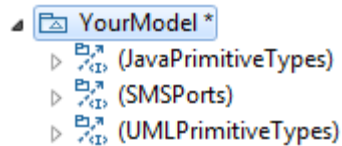
- ▶ YourModel
 - ▶ (UMLPrimitiveTypes)

Right-click **YourModel** and select **Import Model Library**

In **Deployed Library** select **JavaPrimitiveTypes**.

Repeat the process again for SMSPorts but now select **Library in Workspace** and browse to the **SMSPorts.emx**.

Your project should now look like this:



Create the necessary packages

We will be using the same package structure as in ICU. So you will need to create the following (**you may use your own project's names**) using the naming convention of *Your*.

1. A package called **Your** that will contain all your statemachines, internal signals, internal class definitions, and all your sequence diagrams that are not *top-level*. Importantly it will also contain your **root-composite class**, more on this in: Modeling with the JavaFrame Profile
 - a. Create a Class Diagram in this package.
2. A package called **YourContext** that will contain your Collaboration diagram that represents your system's **context** in the real world, and your **top-level** sequence diagrams.
 - a. Create a Class Diagram in this package.
3. A package called **YourUsecases**, that contains your Usecase diagram and the **Actors used in your collaboration diagram above**.
 - a. Create a Usecase Diagram in this package.

Methodology

There are a couple of useful hints to how you should proceed with modeling your project. Most of these are related to the **order of creation** of elements so that you get current **typing** and **references** quickly.

The steps below are by no means obligatory, but they may help you get things right **the first time**.

In the Usecases package

1. Create your Usecase diagram under **YourUsecases**
 - a. Create **Actors** and a **subsystem** *theYourSystem* here. The subsystem **does not refer to your concrete implementation composite** (e.g. <<Composite>> ICUSystem in the ICU package). Create usecases as needed.

In the Context package

2. Once you have all your Actors in place in the usecases you may create your **Collaboration diagram**.
 - a. Having your Actors already prepared allows you to use those Actors as **type** for parts in the Collaboration diagram (e.g. Actor::**Mobile** in ICU on the part **ICUenvironment::Mobile**)
 - b. However, we have not yet created a **Composite class** for your system in the **Your** package. You may create a part and select **Create A Class**.
 - i. This class is created in the package **YourContext**. Move the class to the **Your package** by dragging and dropping it in the Project Explorer.
 - c. You may create Ports on the Part representing your system in the Collaboration diagram. Remember to **type them** with the necessary Mediators from **SMSPorts**.

Also remember to follow the steps later in the document (Modeling with the JavaFrame Profile) to configure your Composite class and ports correctly.

3. You may now also start creating **Sequence Diagrams** inside your **Collaboration**
 - a. Create new sequence diagrams **inside your collaboration**
 - b. This will automatically create an **Interaction** within which your diagram resides.
 - c. Expand the Interaction and right-click the **SequenceDiagram**
 - i. Select **Open Diagram With SeDI**. Do NOT use RSMs internal editor to edit sequence diagram.

In the System package

4. Once you have created your top-level sequence diagrams you may begin creating your system, and you may also begin decomposing your YourSystem lifelines that you have created in the sequence diagrams in the **Context** package.
 - a. Decomposing a Lifeline will Automatically create an Interaction in the proper place and with the name given. E.g. decomposing the lifeline **icuserver : ICUSystem** in **KMLFile** will create an Interaction in the **ICUSystem class**.
 - b. You will need to manually add a Sequence Diagram to the interaction that has been created.

General tips

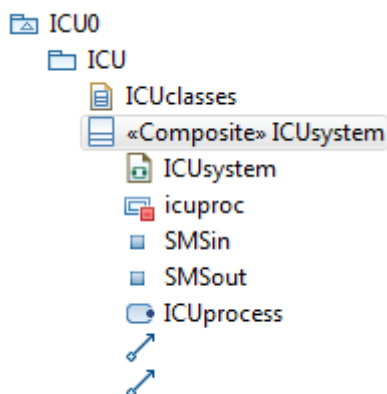
Draw sequence diagrams first and *then* statemachines. Refactoring and changing sequence diagrams is easy, changing statemachines and system structure is hard.

Modeling with the JavaFrame Profile

There are some initial basic steps that need to be done to your model before we are able to transform it to JavaFrame.

Root <<Composite>> Class

- To your top-level element (**Model**) add the Profile **JavaFrame** in the properties view.
- The systems root class needs JavaFrame's *Composite* stereotype applied with property **main** set to true



Properties Tasks Console Bookmarks Error Log Problems

<Class> «Composite» ICU0::ICU::ICUsystem

General Keywords:

Attributes Applied Stereotypes:

Stereotype	Profile	Required
Composite	JavaFrameProfile	False

Operations

Stereotypes

Documentation

Constraints

Relationships

Advanced

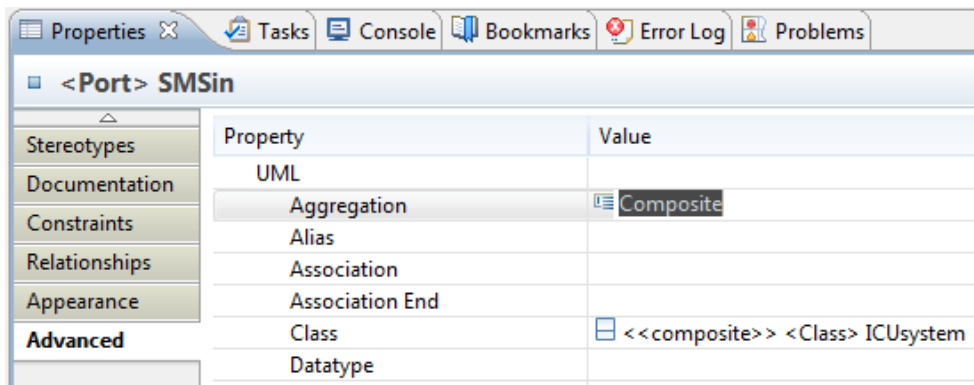
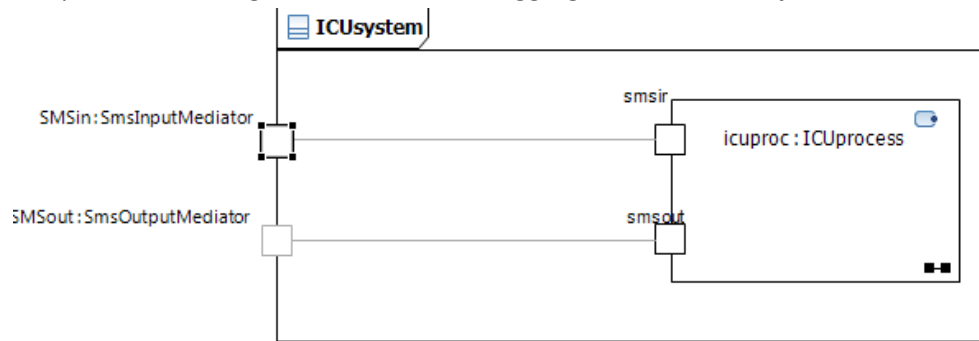
Apply Stereotypes... Unapply Stereotypes

Stereotype Properties:

Property	Value
Composite	
main	True

SmsInput and SmsOutput Mediators

The ports connecting to PATS need their Aggregation set to **Composite**



- o Do this for both SmsInputMediator and SmsOutputMediator

You also need to set the ports **default value** as pictured, both for Input and Output.

Name:	SMSin
Type:	<<mediator>> <Class> SmsInputMediator Select type ...
Default Value:	ARGS[0],ARGS[0] ...
Multiplicity:	1
Service:	<input checked="" type="checkbox"/> Service
Behavior:	<input type="checkbox"/> Behavior

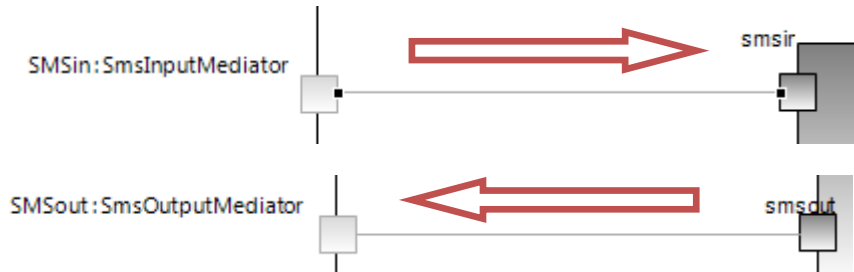
These values take the **Username** the you use as an argument when running the system and initializes the port with it.

Tips, Tricks and Work-arounds for modeling in RSM

Important information about Composite structures and Connectors

When drawing Connectors keep in mind that they are **directional!**

The line itself does not have any arrow-head, but they are "there" just not visible.

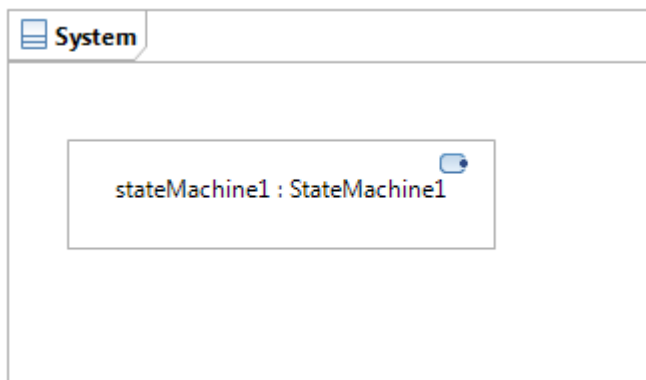


Visual tips

If for some reason you cannot **see** a compartment or some other visual element that "should" be in the diagram try **right-clicking the element** and select **Filters**.

Adding Parts to Composite Structures

You may **drag and drop** elements from the Project Explorer into the composite structure. RSM will then automatically generate an attribute name (remember to double-check it).

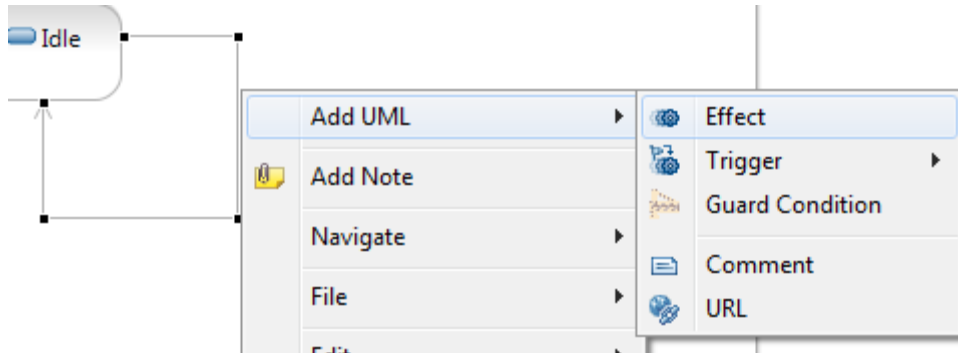


When adding a **Part** to a **Composite structure** selecting a StateMachine under "Select an existing element" is not possible. There are two ways to do this:

1. Drag the statemachine from the **Project Explorer** into the Composite structure.
2. Create a Part as **Unspecified** and then select the StateMachine

Adding Activities as Effects in Transitions

To add an Activity you need to right-click the transition and select **Add UML, Effect**. You will then be given the choice of **OpaqueBehavior, StateMachine or Activity**. We will **not** use StateMachines in transitions.



Adding triggers to Transitions

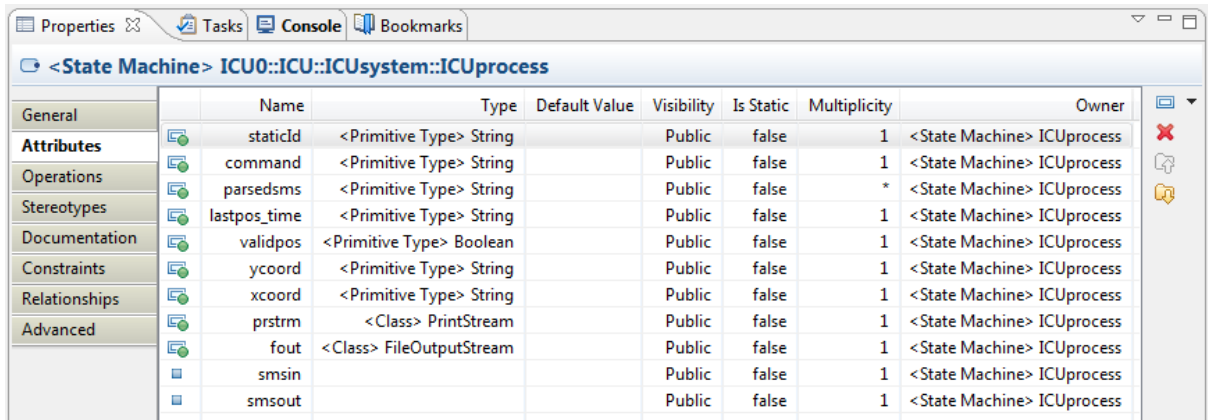
You may add triggers by either:

- Typing the name of the signal as the name of the transition (be precise!)
- Adding it as a Trigger via the **Properties view** or the **context-menu**.
 - If you use the context menu use **Signal Events**, and select the proper **Signal Type** from the list. E.g. for an **Sms signal event** choose **Sms** from **SMSPorts**.


Adding Attributes to Statemachines

Adding attributes to Statemachines is essentially the same as adding Global variables to a Class (where the Statemachine is the class, and the transitions are methods). This allows you to store and retrieve data from transition to transition.

Adding attributes is done via the **Properties view** when a Statemachine is selected.



General	Name	Type	Default Value	Visibility	Is Static	Multiplicity	Owner
Attributes	staticId	<Primitive Type> String		Public	false	1	<State Machine> ICUprocess
Operations	command	<Primitive Type> String		Public	false	1	<State Machine> ICUprocess
	parsedsms	<Primitive Type> String		Public	false	*	<State Machine> ICUprocess
Stereotypes	lastpos_time	<Primitive Type> String		Public	false	1	<State Machine> ICUprocess
Documentation	validpos	<Primitive Type> Boolean		Public	false	1	<State Machine> ICUprocess
Constraints	ycoord	<Primitive Type> String		Public	false	1	<State Machine> ICUprocess
Relationships	xcoord	<Primitive Type> String		Public	false	1	<State Machine> ICUprocess
Advanced	prstrm	<Class> PrintStream		Public	false	1	<State Machine> ICUprocess
	fout	<Class> FileOutputStream		Public	false	1	<State Machine> ICUprocess
	msin			Public	false	1	<State Machine> ICUprocess
	msout			Public	false	1	<State Machine> ICUprocess

Add attributes by clicking the top-button on the right .

As you can see all the attributes in **ICU0** are **typed**, either as Primitives or as we can see from **prstrm** and **fout** as **Classes**.

Attributes that are typed to Classes refer to concrete Classes in the model. As in ICU0 you may create a **java** package at the root level. Within this package you may create other packages, and then finally the Class itself that you want as a type.

Creating a Package structure like **java/util** with a Class **LinkedList** works in much the same way as an **import** statement in Java, e.g. **import java.util.LinkedList**. The JavaFrame transformation engine employs this information to set the correct imports in the generated code.

Multiplicity in Attributes

In ICU0 the attribute **parsedsms** has a ***** multiplicity. This will when transforming the model with JavaFrame result in **parsedsms** becoming an Array of type String. You may use any ***** typed attributes as arrays in your in-model code.