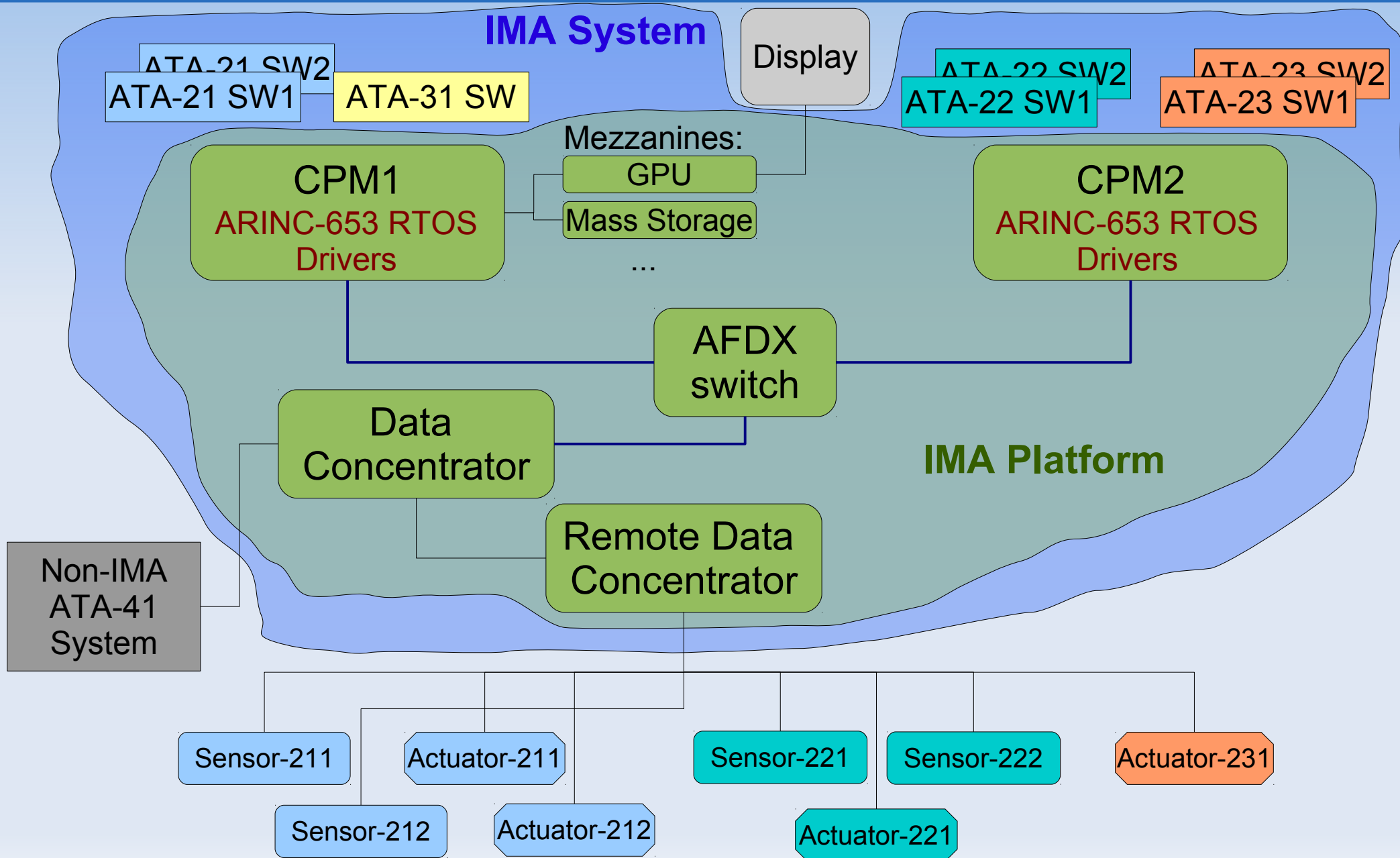


Development and Analysis of AADL Models with MASIW Framework *Tutorial*

Alexey Khoroshilov
khoroshilov@ispras.ru

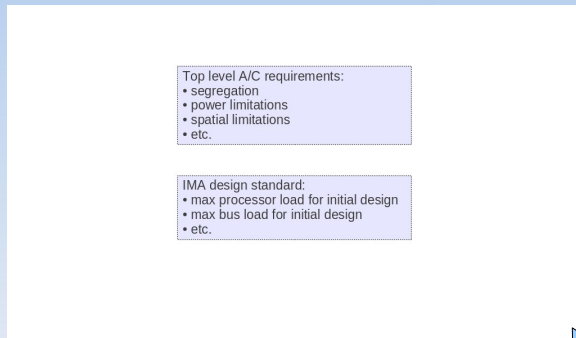


IMA System Design (1)

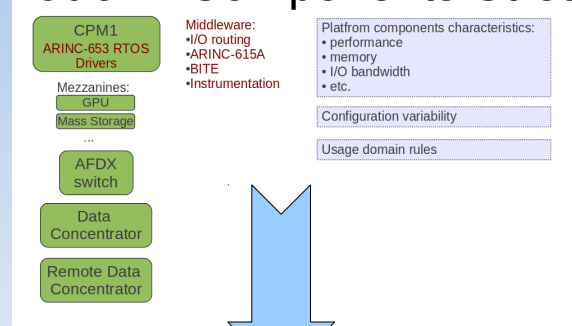


IMA System Design (2)

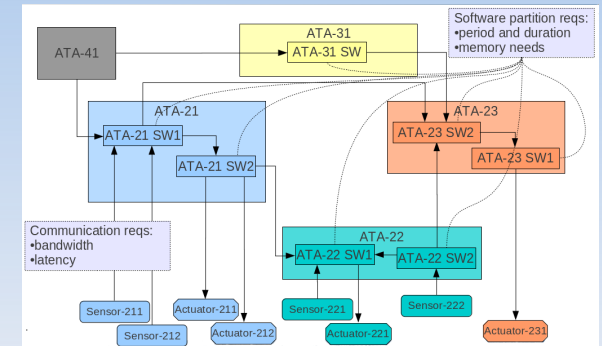
(1) A/C Stream



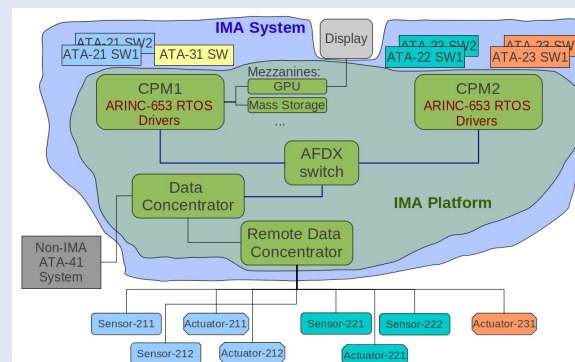
(2) Platform Components Stream



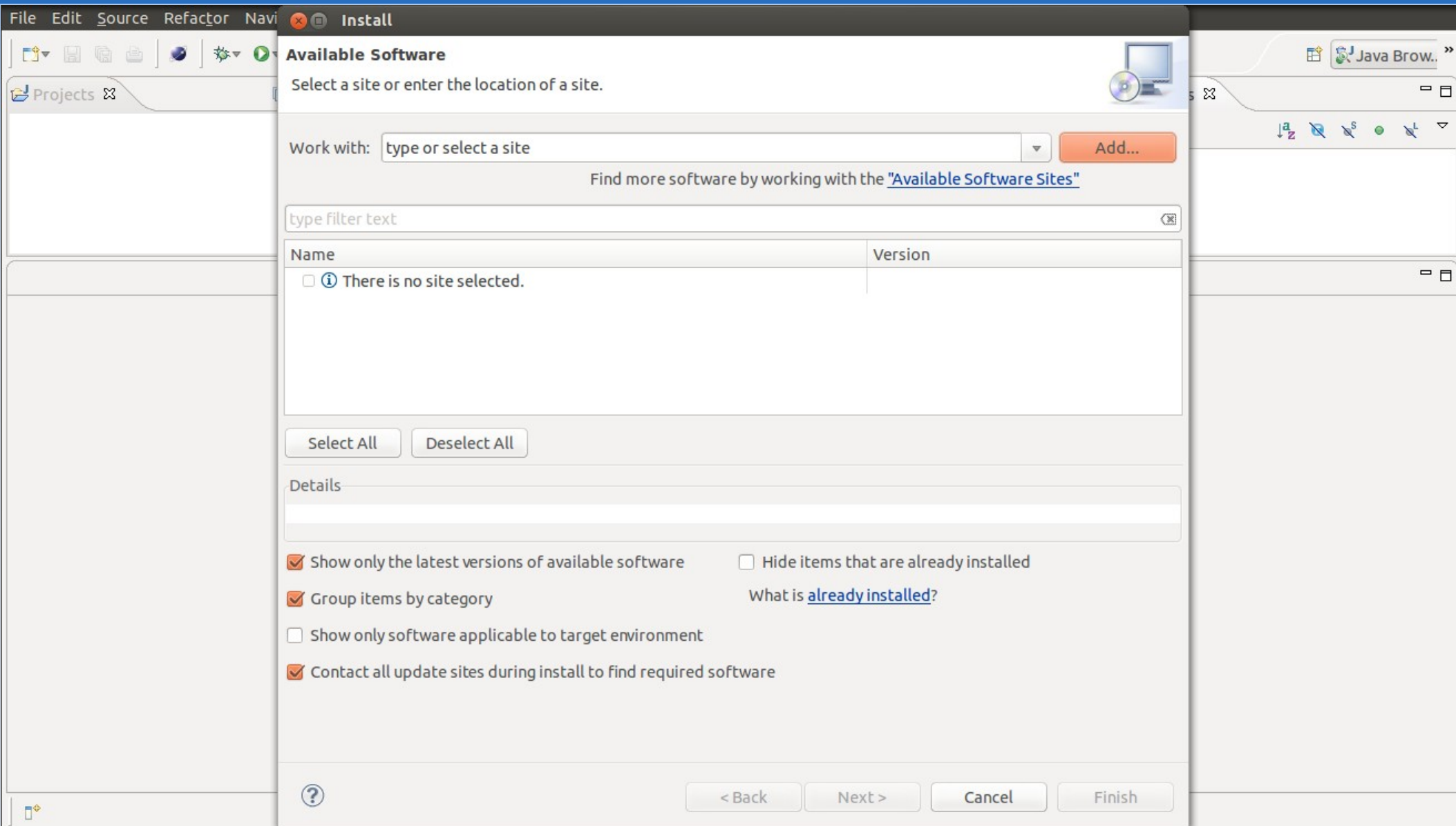
(3) ATA-XX Stream



IMA System Designer

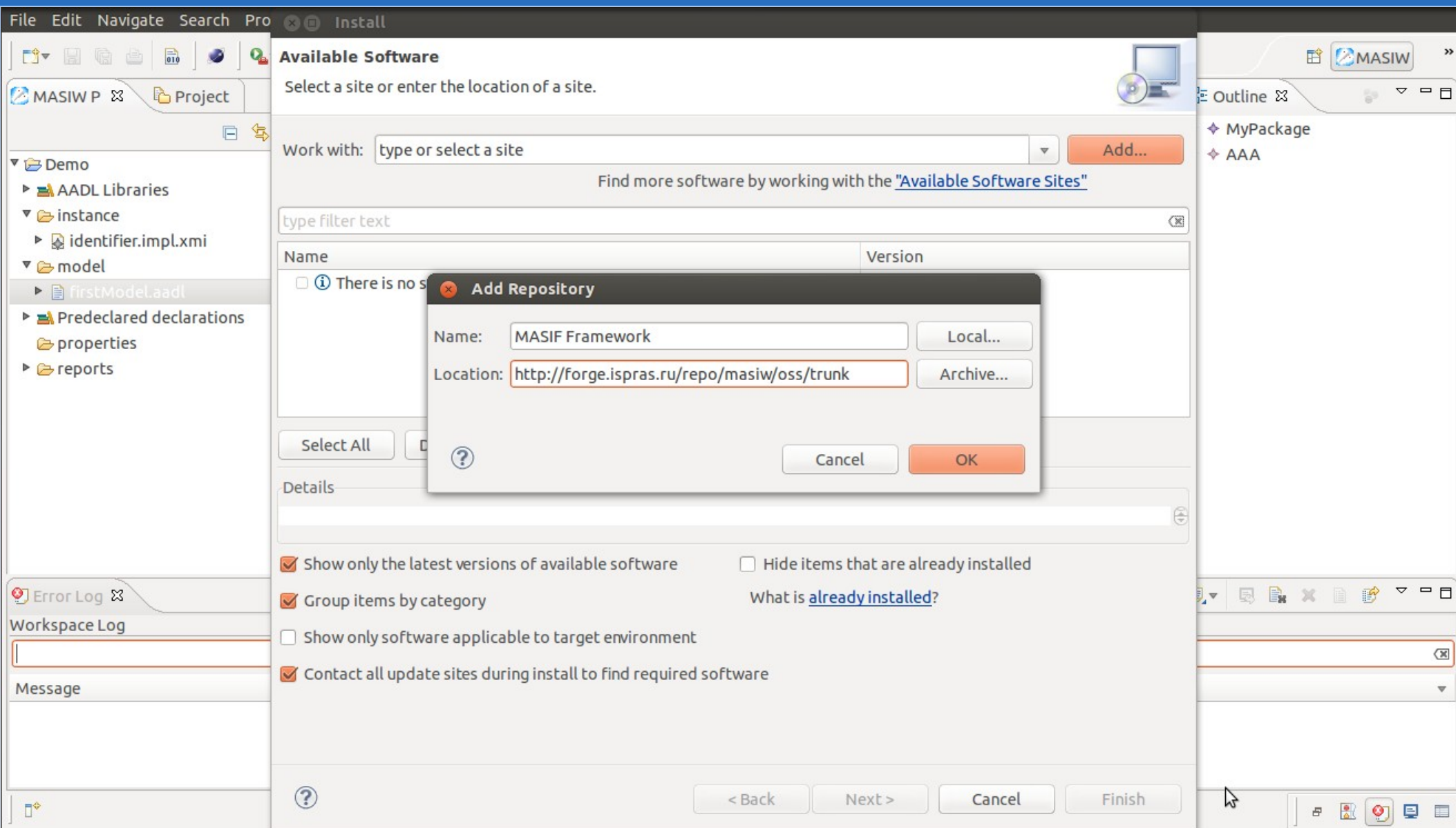


MASIW Installation (1)



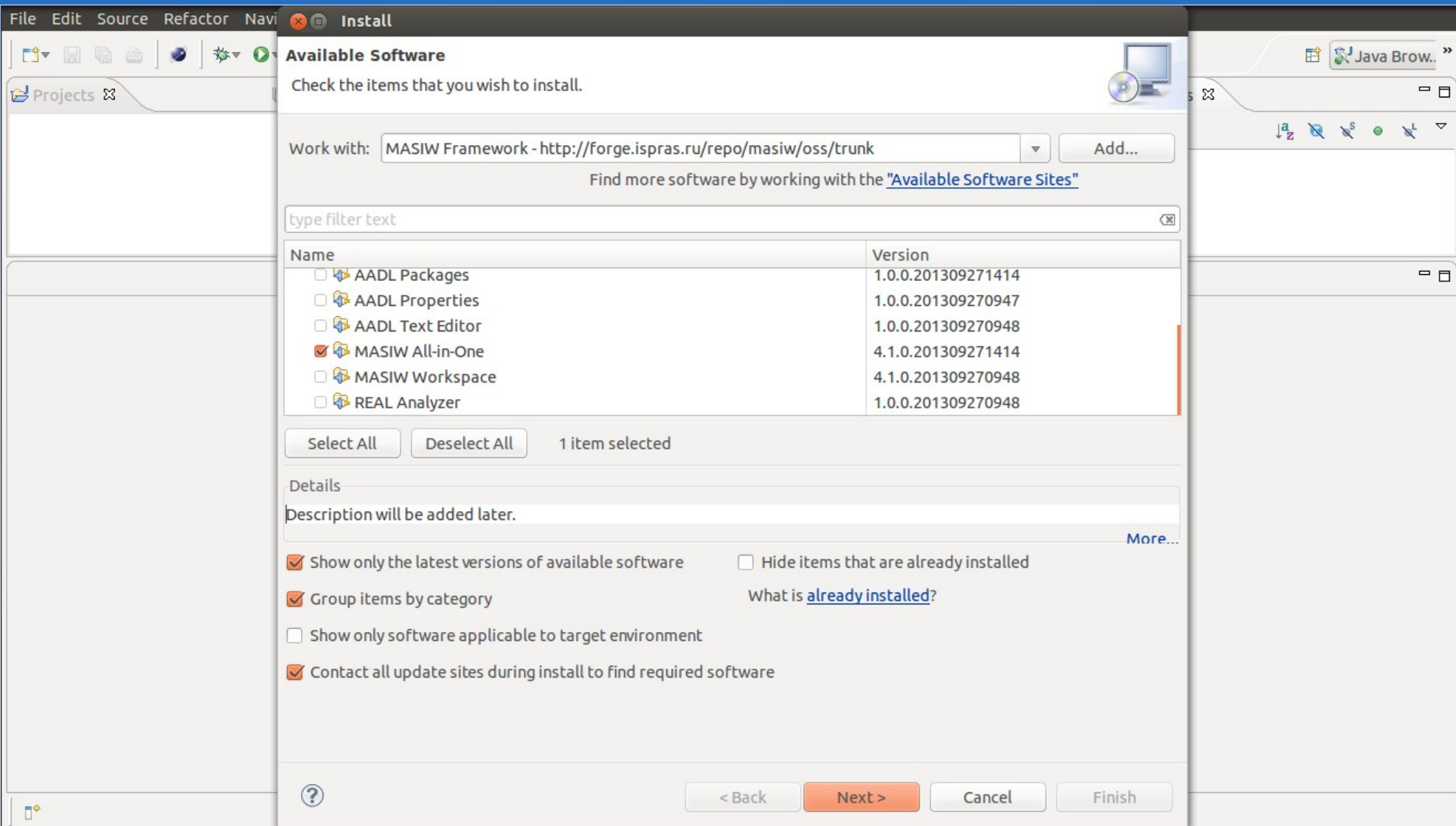
Help->Install new software...-->Add...

MASIW Installation (2)



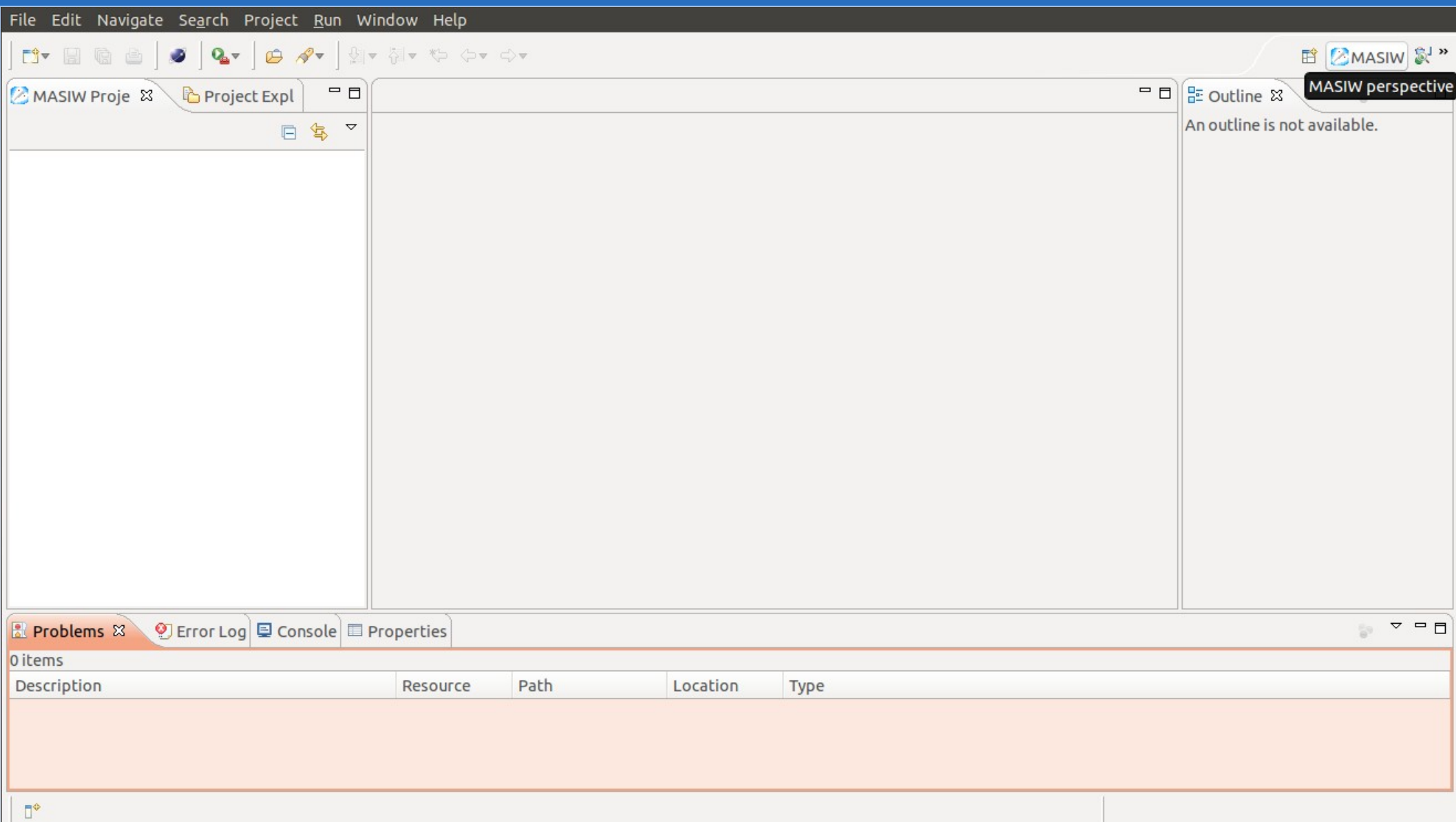
Name: MASIW Framework
Location: <http://forge.ispras.ru/repo/masiw/oss/trunk>

MASIW Installation (3)



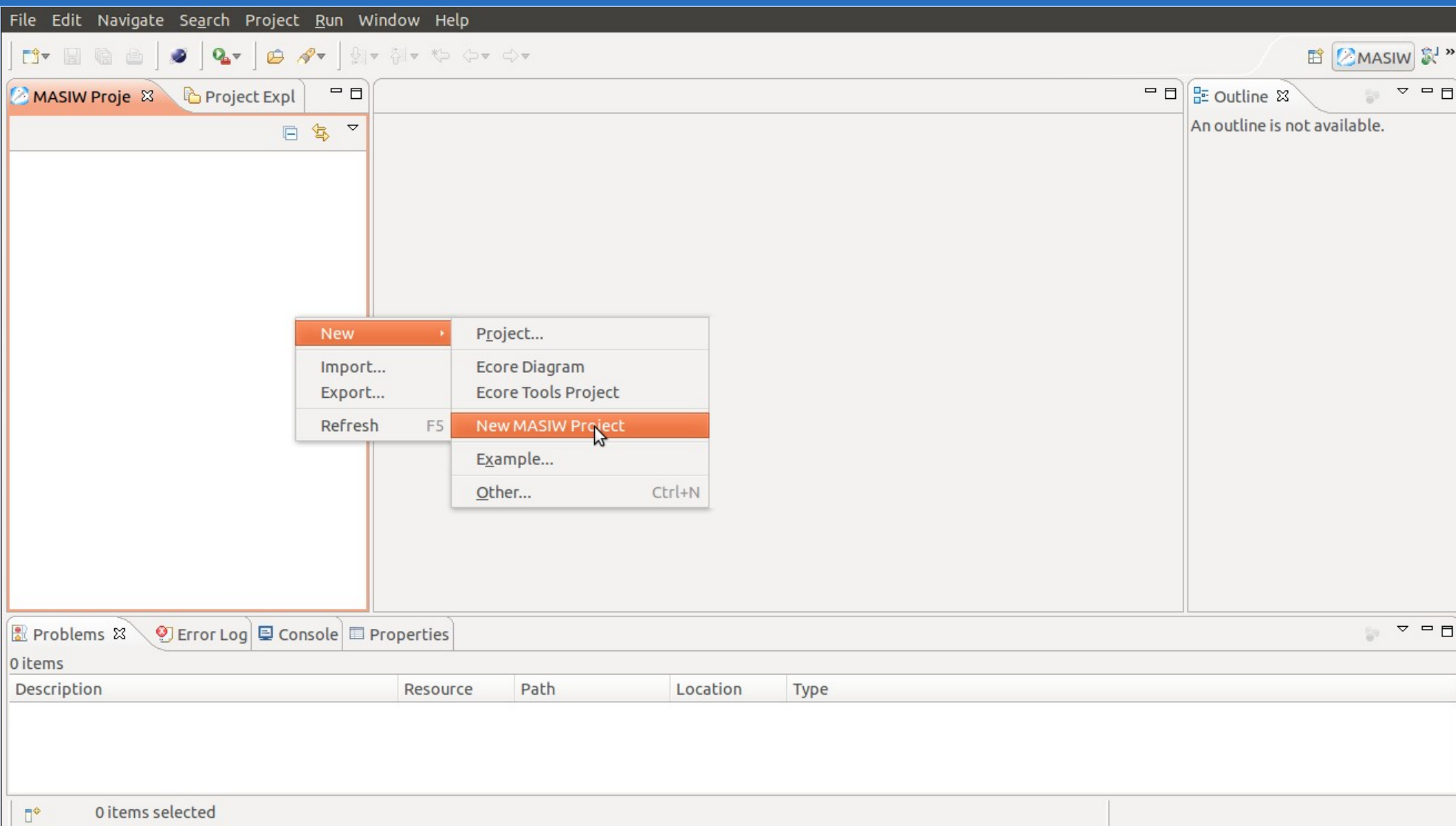
Choose the only component: MASIW All-in-One

MASIW Perspective



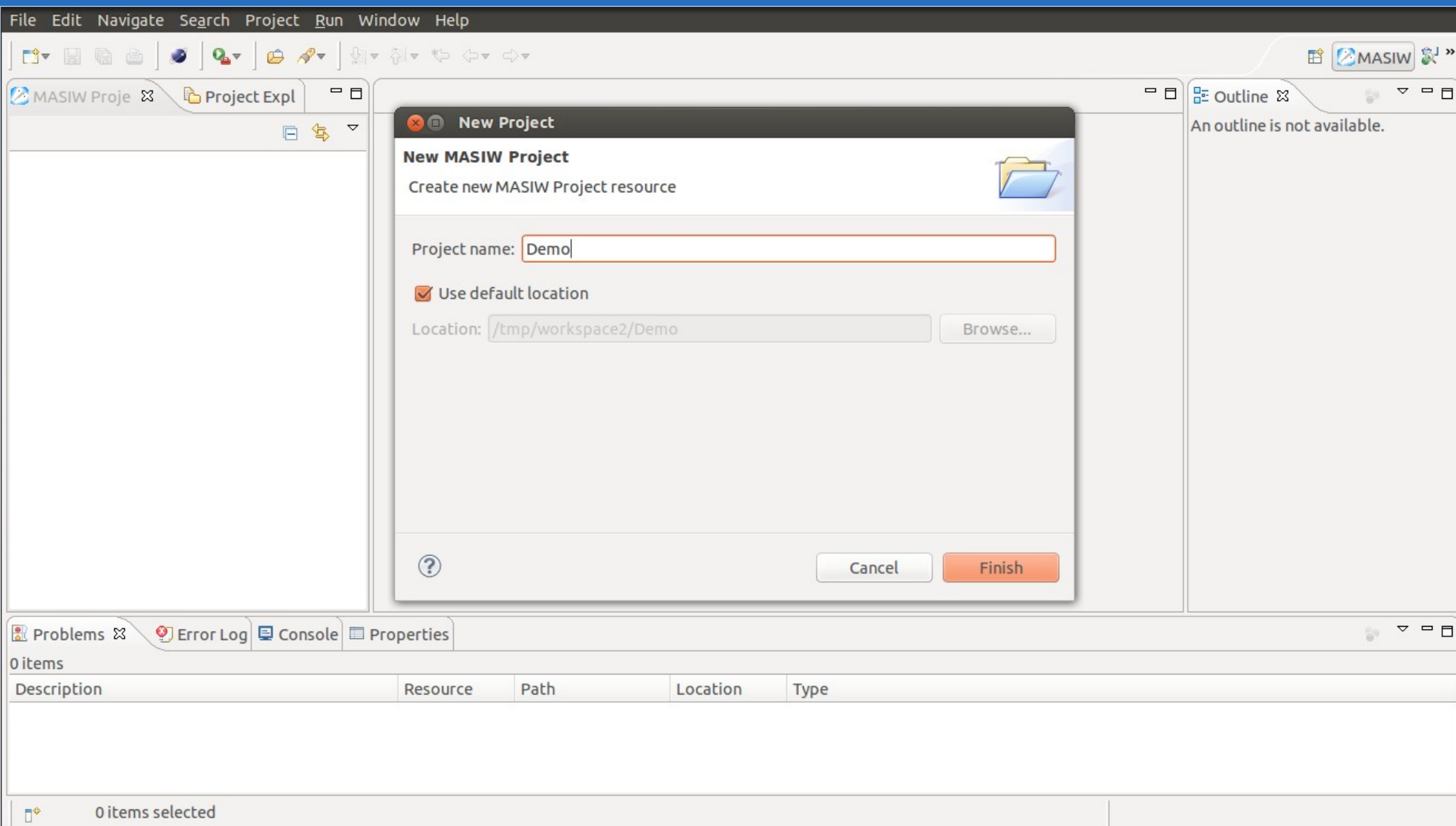
Open MASIW perspective

New MASIW Project (1)



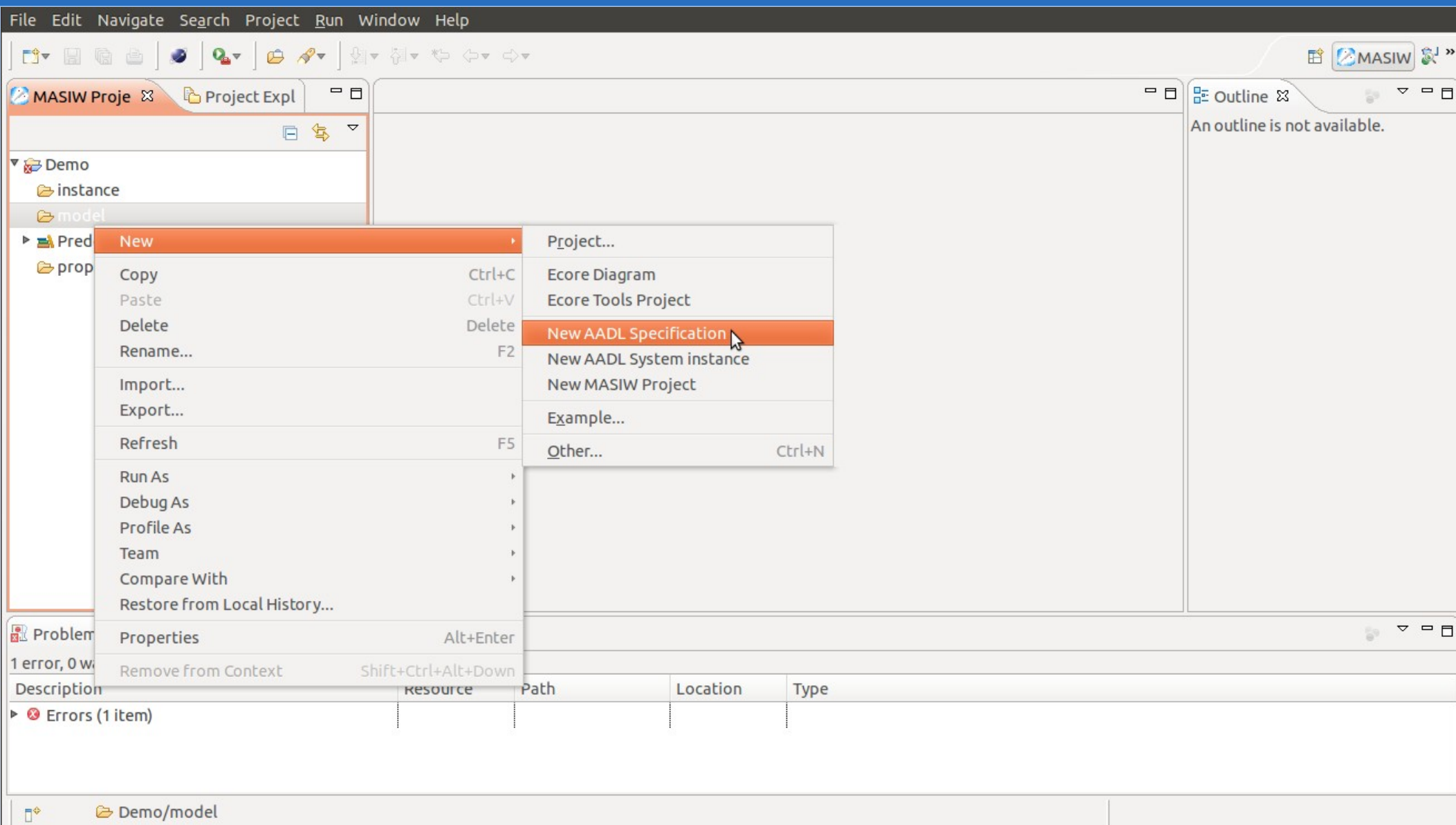
Context menu in MASIW Projects: New->New MASIW Project

New MASIW Project (2)



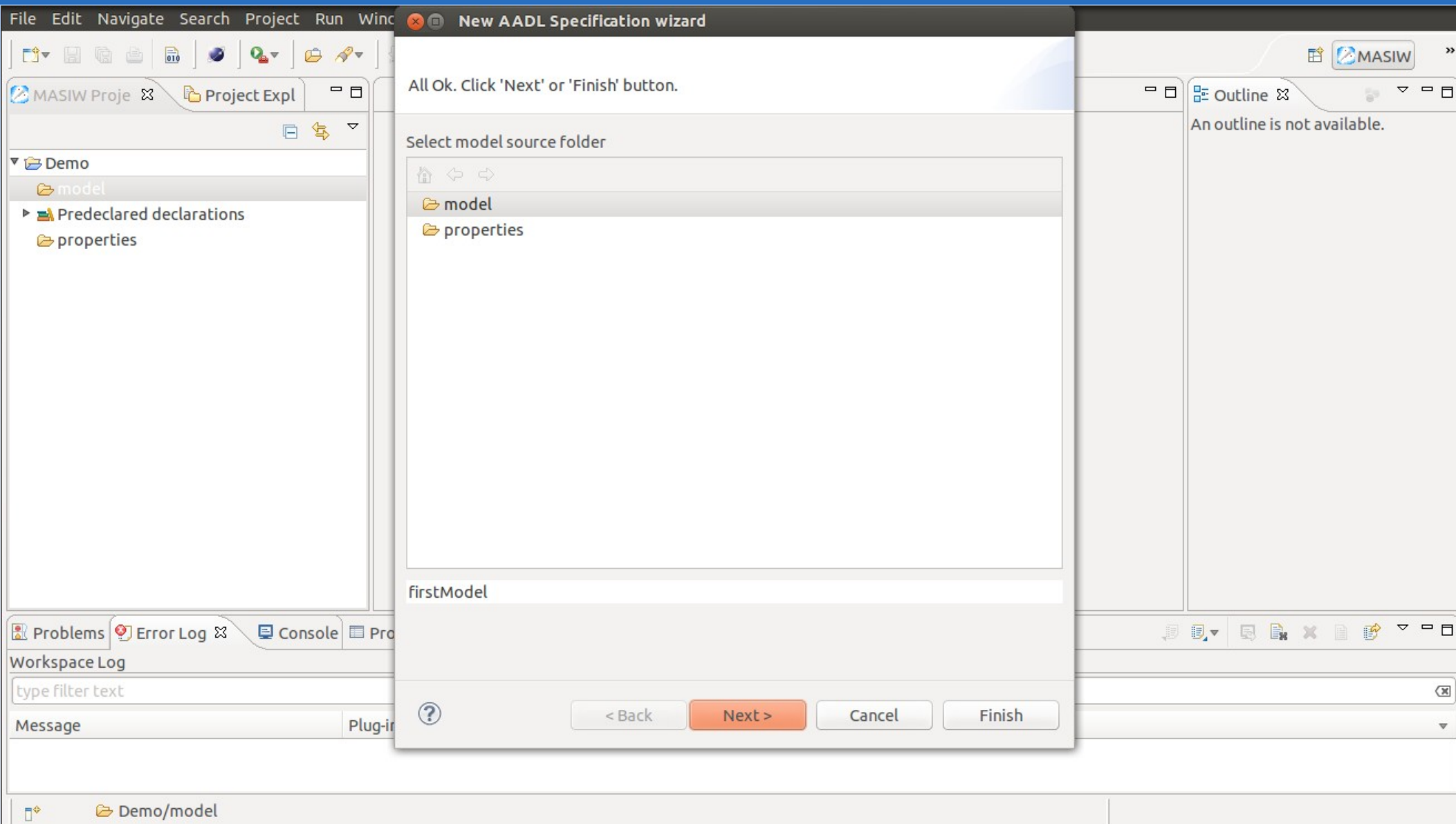
Define project name and click on Finish button

New MASIW Specification (1)



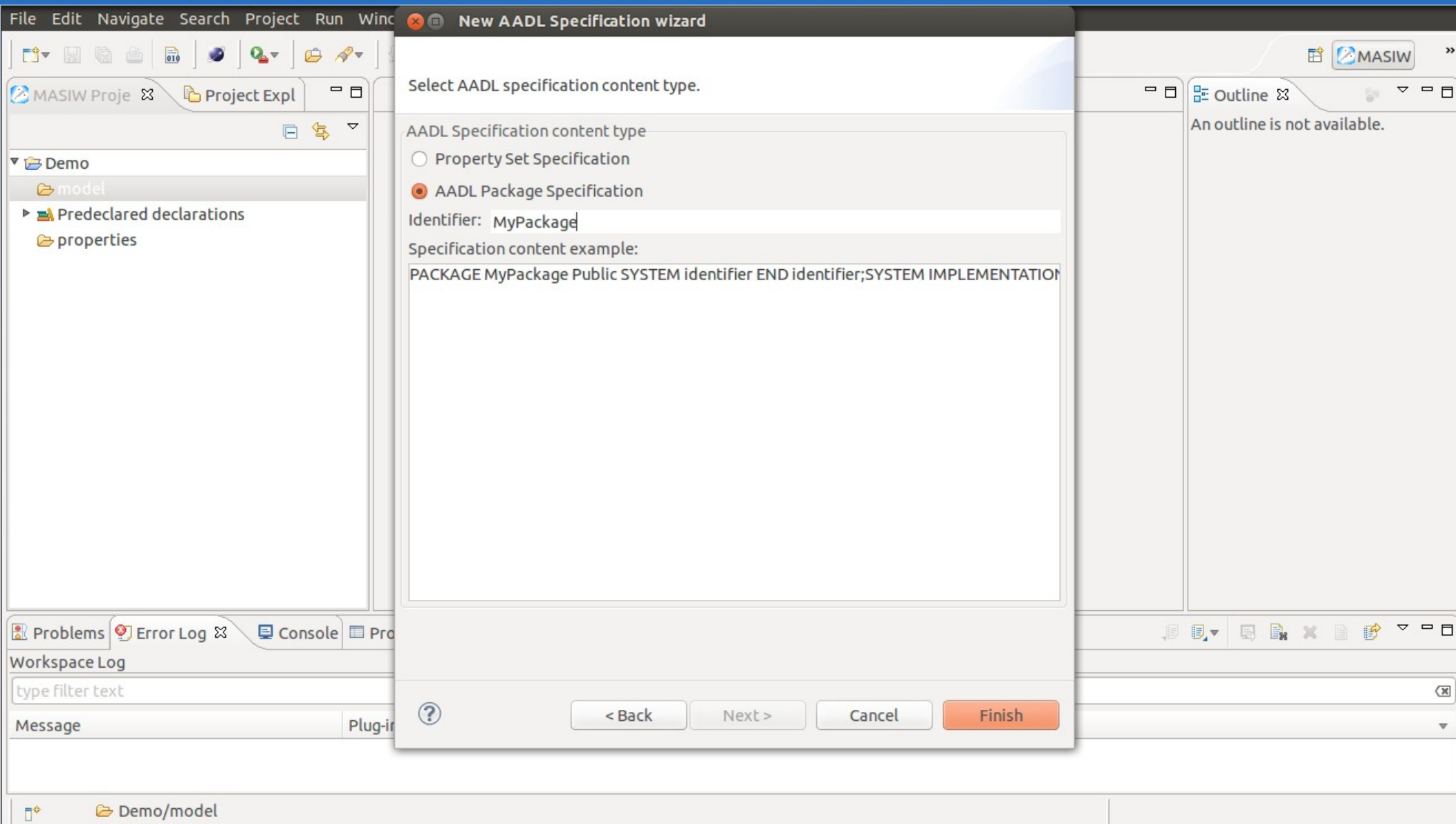
Context menu in MASIW Projects: New->New AADL Specification

New MASIW Specification (2)



Define name of a new aadl file and click on Next> button

New MASIW Specification (3)



Choose AADL Package Specification, define name of the package and click on Finish button

New MASIW Specification (4)

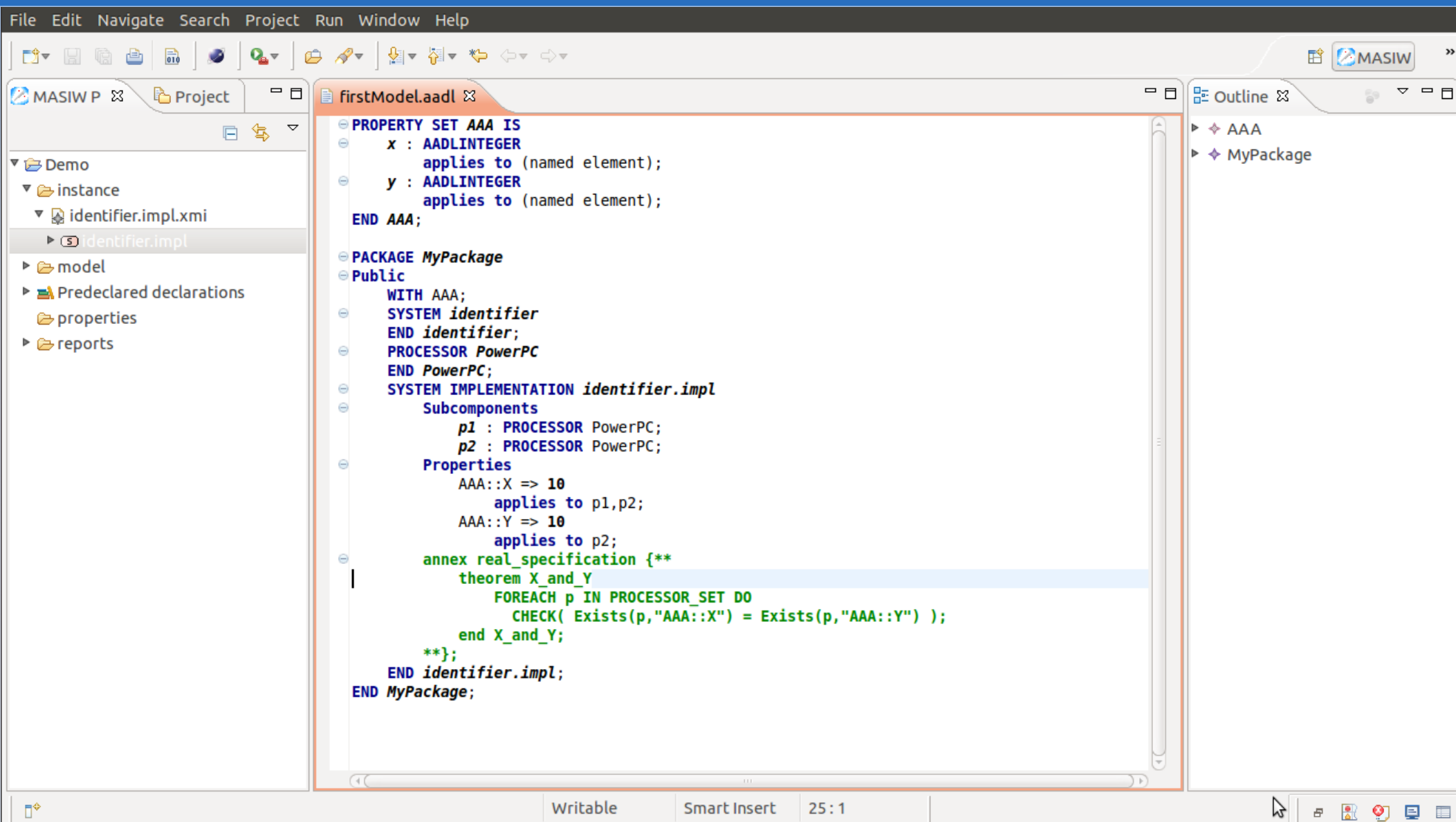
The screenshot displays the MASIW IDE interface. The main editor window, titled 'firstModel.aadl', contains the following AADL code:

```
PACKAGE MyPackage
Public
|
|  SYSTEM identifier
|  END identifier;
|  SYSTEM IMPLEMENTATION identifier.impl
|  END identifier.impl;
END MyPackage;
```

The code is formatted with indentation and line wrapping. The IDE also shows a Project Explorer on the left with a 'Demo' folder containing 'model', 'Predeclared declarations', and 'properties'. An Outline view on the right shows the 'MyPackage' structure. The bottom status bar indicates 'Writable', 'Smart Insert', and '3:1'.

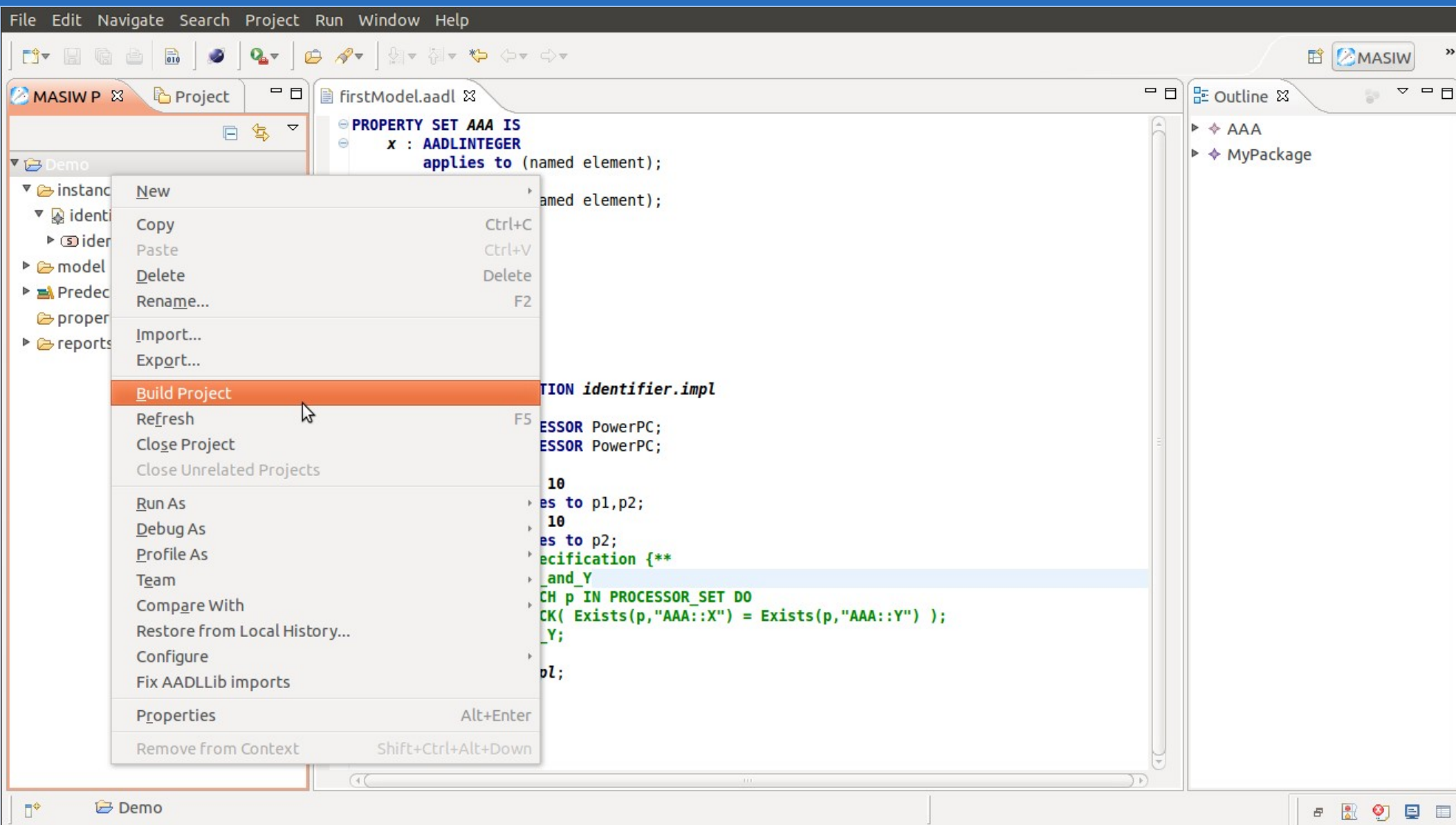
Use Ctrl-Shift-F to format aadl specification

MASIW Textual Editor



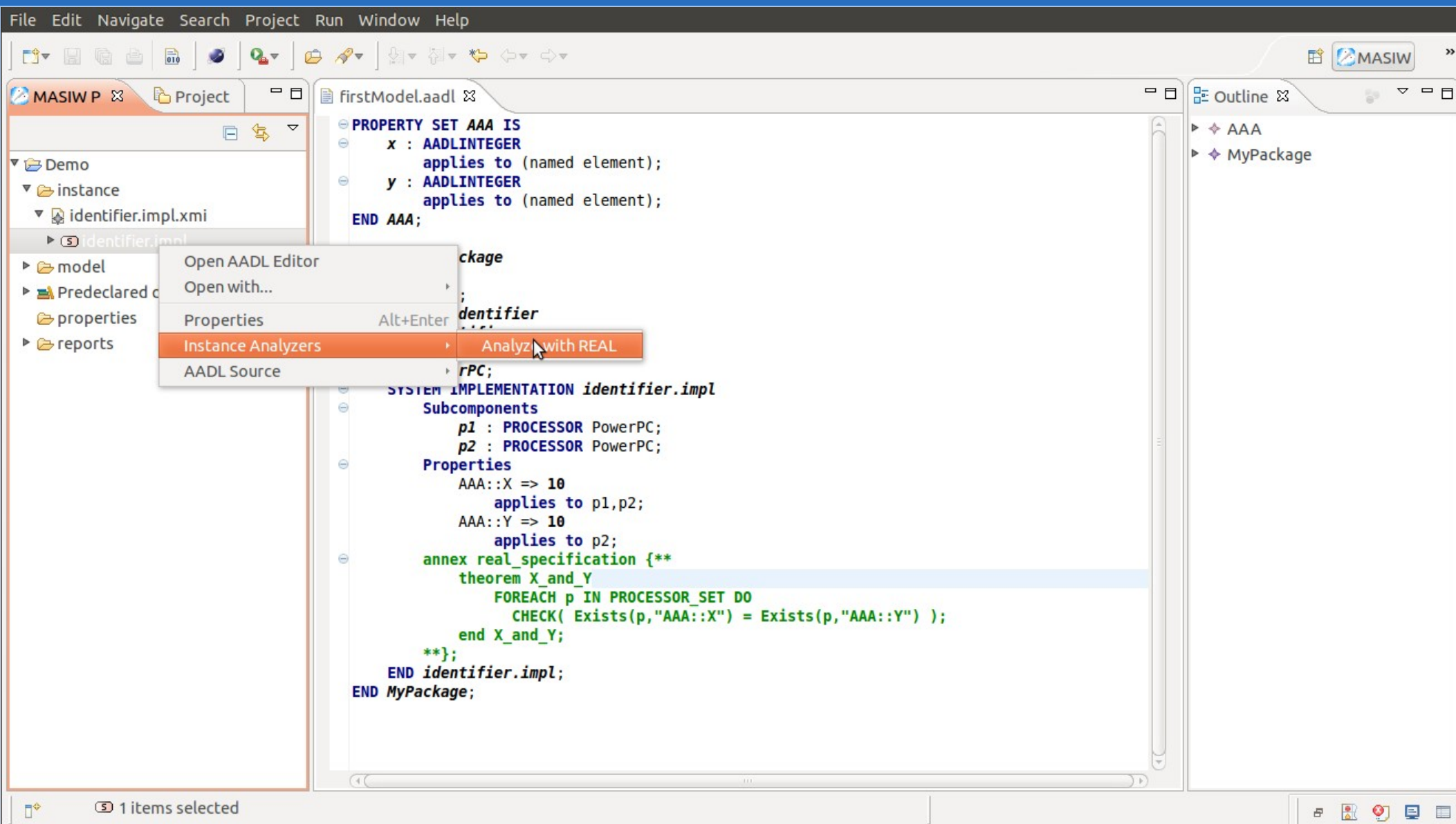
Define property set, package with a system implementation containing a couple of processors and a REAL theorem that check presence of both properties

Checking Theorems (1)



Context menu in MASIW Projects: Build Project

Checking Theorems (2)



Context menu in MASIW Projects: Instance Analyzers->Analyze with REAL

Checking Theorems (3)

The screenshot shows the MASIW IDE interface. The main editor displays the AADL code for `firstModel.aadl`. A dialog box is open in the center, titled "ERROR: Analyze with REAL results". The dialog contains a table with the following data:

Type	Instance	Name	Message
ERROR	identifier.impl	Annex REAL_SPECIFICATION for identifier.impl	FAILED

Below the table, there is a "Generate report" button (highlighted in orange), a "Cancel" button, and an "OK" button. The background code in the editor includes:

```
PROPERTY SET AAA IS
  x : AADLINTEGER
    applies to (named element);
  y : AADLINTEGER
    applies to (named element);
END AAA;

PACKAGE MyPackage
Public
  W
  S
  E
  P
  E
  S
  applies to p2;
  annex real_specification {**
    theorem X_and_Y
      FOREACH p IN PROCESSOR_SET DO
        CHECK( Exists(p,"AAA:X") = Exists(p,"AAA:Y") );
      end X_and_Y;
    **};
END identifier.impl;
END MyPackage;
```

Click on 'Generate report' button

Checking Theorems (4)

The screenshot displays the MASIW REAL Analyzer Report interface. The main window shows the report for the instance **identifier.impl**. The report title is **Annex REAL_SPECIFICATION for identifier.impl**. Below the title, it states *Check theorems from the annex* and **Analyzer status: FAILED**. A table lists the verification results for the theorems in the annex.

Instance	Name	Verification status
identifier.impl	Theorem MyPackage::X_and_Y	FAILED

The table has three columns: Instance, Name, and Verification status. The row for 'Theorem MyPackage::X_and_Y' is highlighted in red, indicating a failed verification. The 'Verification status' column contains the text 'FAILED'.

On the right side of the interface, there is an 'Outline' panel with the message 'An outline is not available.' The bottom status bar contains several icons for navigation and help.

Click on the FAILED row to see details

Checking Theorems (5)

The screenshot shows the MASIW REAL Analyzer Report interface. The main window displays the following information:

- Instance: **identifier.impl**
- Theorem: **MyPackage::X_and_Y**
- Analyzer status: **FAILED**

Instance	Name	Verification status
identifier.impl.p1	REAL Theorem: MyPackage::X_and_Y	Postcondition violation
identifier.impl.p2	REAL Theorem: MyPackage::X_and_Y	PASSED

The interface also includes a left sidebar with a project tree, a top menu bar (File, Edit, Navigate, Search, Project, Run, Window, Help), and a right sidebar with an Outline panel.

REAL theorem X_and_Y is failed for processor p1 of system identifier.impl, but it is not very helpful description for AADL model developer

Checking Theorems (6)

The screenshot shows the MASIW IDE interface. The main editor window displays the AADL code for 'firstModel.aadl'. The code defines a property set 'AAA' with variables 'x' and 'y' of type 'AADLINTEGER'. It then defines a package 'MyPackage' which includes the 'AAA' property set and a system 'identifier'. The system 'identifier' has two processors, 'p1' and 'p2', both of type 'PowerPC'. It also includes a system implementation 'identifier.impl' with subcomponents 'p1' and 'p2'. The implementation defines properties for 'AAA::X' and 'AAA::Y', both set to 10. A theorem 'X_and_Y' is defined within the implementation, stating that for each processor 'p', the existence of 'AAA::X' is equal to the existence of 'AAA::Y'. The theorem is currently highlighted in orange. A comment is being added to the theorem, starting with '--@', describing it as 'Each processor has to have either X and Y both or none of the theorem X_and_Y'. The IDE also shows a project tree on the left and an outline on the right.

```
PROPERTY SET AAA IS
  x : AADLINTEGER
    applies to (named element);
  y : AADLINTEGER
    applies to (named element);
END AAA;

PACKAGE MyPackage
Public
  WITH AAA;
  SYSTEM identifier
    END identifier;
  PROCESSOR PowerPC
    END PowerPC;
  SYSTEM IMPLEMENTATION identifier.impl
    Subcomponents
      p1 : PROCESSOR PowerPC;
      p2 : PROCESSOR PowerPC;
    Properties
      AAA::X => 10
        applies to p1,p2;
      AAA::Y => 10
        applies to p2;
    annex real_specification {**
      --@ Each processor has to have either X and Y both or none of the
      theorem X_and_Y
        FOREACH p IN PROCESSOR_SET DO
          CHECK( Exists(p,"AAA::X") = Exists(p,"AAA::Y") );
        end X_and_Y;
    **};
  END identifier.impl;
END MyPackage;
```

Let us add a description of the theorem in a comment starting with '--@'

Checking Theorems (7)

The screenshot shows the MASIW IDE interface. The main window displays a report for the theorem **MyPackage::X_and_Y**. The report title is **Instance: identifier.impl**. The theorem description is *Each processor has to have either X and Y both or none of the*. The analyzer status is **FAILED**.

Instance	Name	Verification status
identifier.impl.p2	REAL Theorem: MyPackage::X_and_Y <i>null</i>	PASSED
identifier.impl.p1	REAL Theorem: MyPackage::X_and_Y <i>Each processor has to have either X and Y both or none of the</i>	Postcondition violation

After 'Build Project', then 'Instance Analyzers->Analyze with REAL'
a new report contains the description helping user to understand the detected issue

Checking Theorems (8)

```
PROPERTY SET AAA IS
  x : AADLINTEGER
    applies to (named element);
  y : AADLINTEGER
    applies to (named element);
END AAA;

PACKAGE MyPackage
Public
  WITH AAA;
  SYSTEM identifier
  END identifier;
  PROCESSOR PowerPC
  END PowerPC;
  SYSTEM IMPLEMENTATION identifier.impl
  Subcomponents
    p1 : PROCESSOR PowerPC;
    p2 : PROCESSOR PowerPC;
  Properties
    AAA::X => 10
      applies to p1,p2;
    AAA::Y => 10
      applies to p2;
  annex real_specification {**
    --@ Each processor has to have either X and Y both or none of the
    theorem X_and_Y
      FOREACH p IN PROCESSOR SET DO
        var eprop := (IF Exists(p,"AAA::X") THEN "X" ELSE "Y");
        var nprop := (IF Exists(p,"AAA::Y") THEN "X" ELSE "Y");
        --@ $p has AAA::$eprop, but does not have AAA::$nprop
        CHECK( Exists(p,"AAA::X") = Exists(p,"AAA::Y") );
      end X_and_Y;
    **};
  END identifier.impl;
END MyPackage.
```

But in complex cases static description is often not enough.
Let us add a runtime generated message that describe specific of the issue.

Checking Theorems (9)

The screenshot shows the MASIW IDE interface. The main window displays a report for the instance `identifier.impl`. The theorem being checked is `MyPackage::X_and_Y`. The analyzer status is **FAILED**. The report includes a table with the following data:

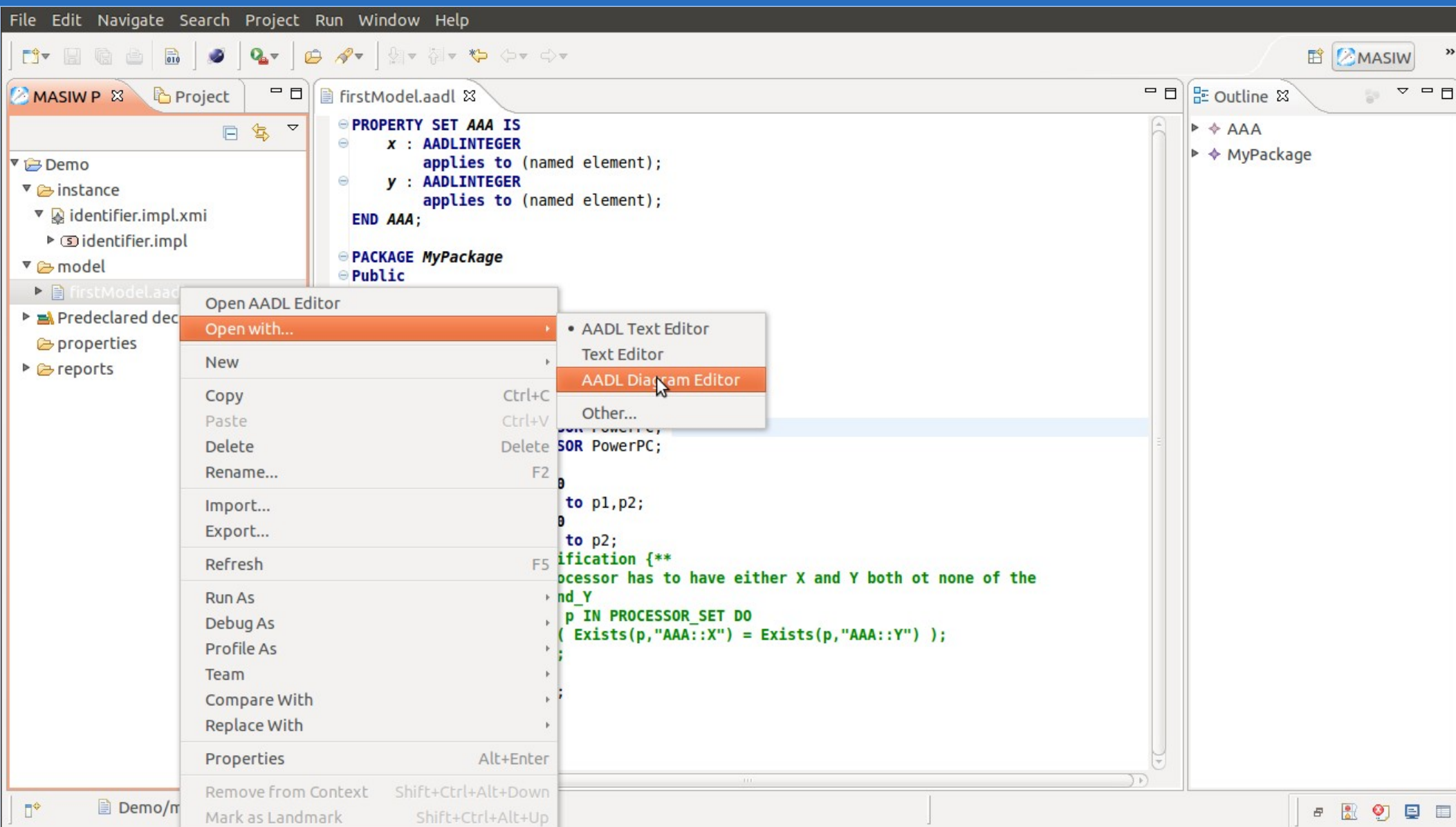
Instance	Name	Verification status
identifier.impl.p2	REAL Theorem: MyPackage::X_and_Y	PASSED
identifier.impl.p1	REAL Theorem: MyPackage::X_and_Y	identifier.impl.p1 has AAA::X, but does not have AAA::Y

The IDE interface also shows a project tree on the left with folders for `Demo`, `instance`, `model`, `Predeclared declarations`, `properties`, and `reports`. The right sidebar shows an `Outline` panel with the message "An outline is not available."

After 'Build Project', then 'Instance Analyzers->Analyze with REAL' a new report contains the specific description of this particular issue

MASIW Graphical Editor

MASIW Graphical Editor (1)

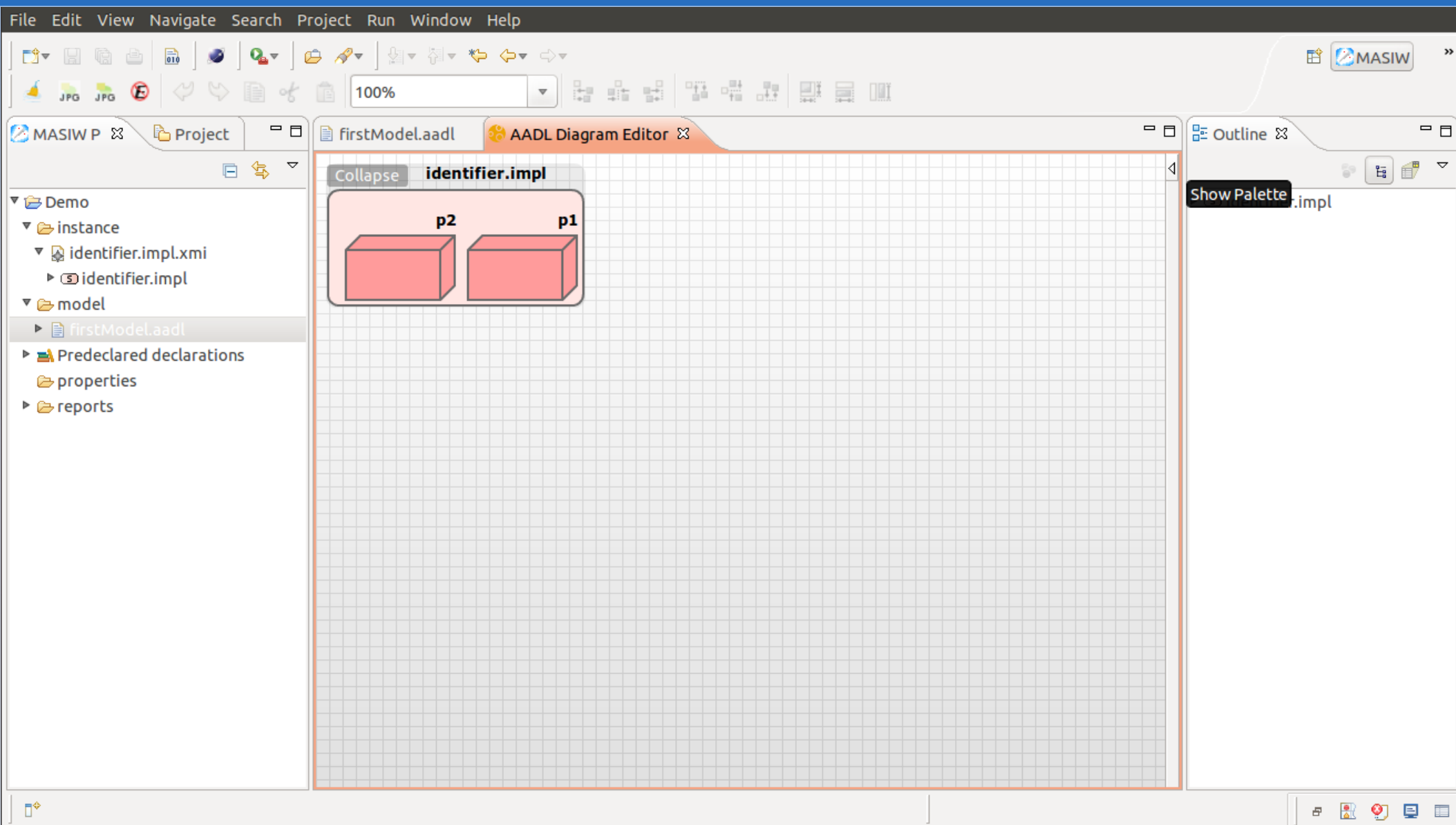


Context menu for aadl file: Open with...->AADL Diagram Editor

MASIW Graphical Editor (2)

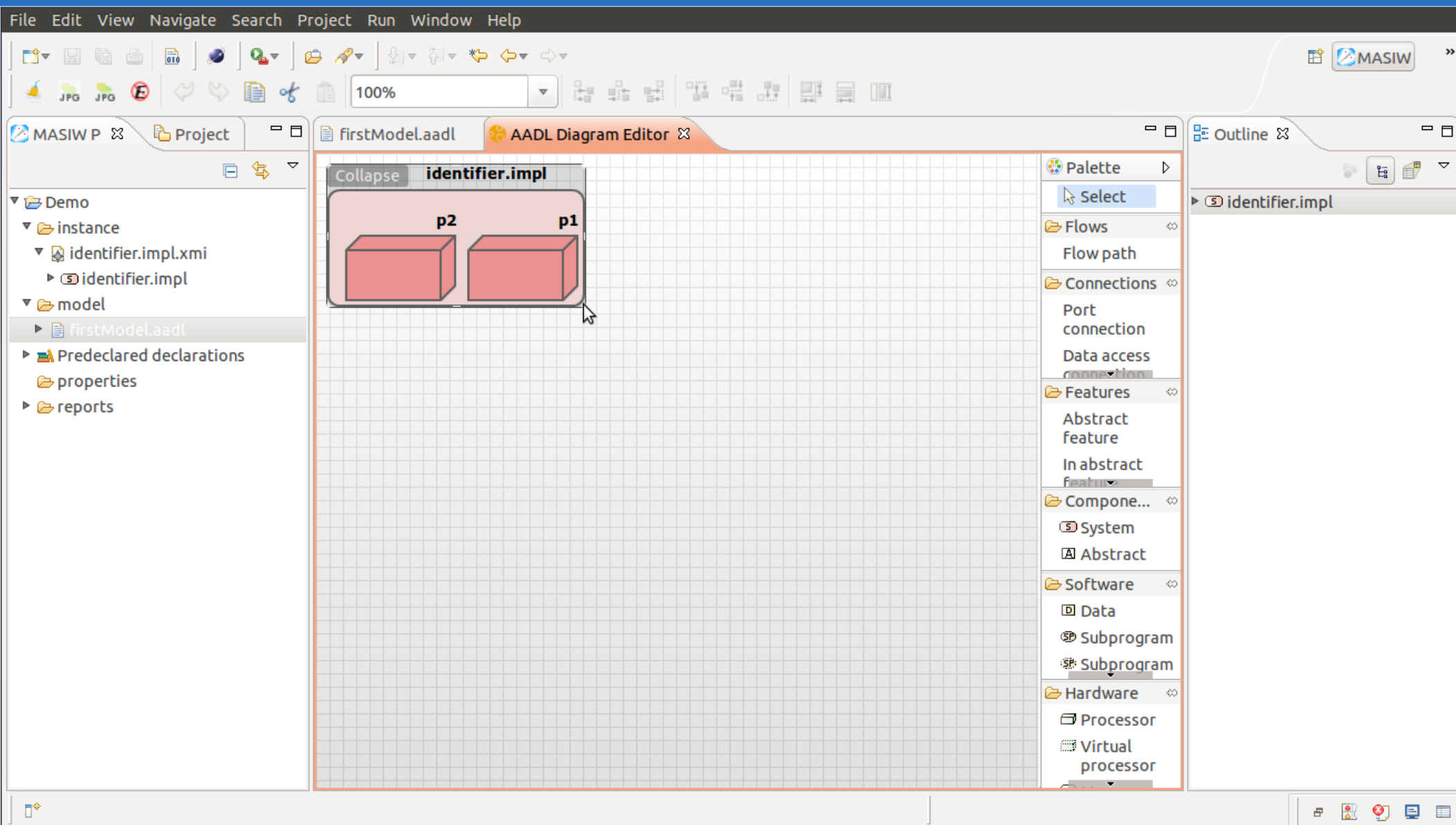
The screenshot displays the MASIW Graphical Editor interface. The main workspace shows an AADL diagram titled "identifier.impl" with a "Collapse" button. The diagram contains two process blocks, "p2" and "p1", represented as red 3D rectangular boxes. The interface includes a menu bar (File, Edit, View, Navigate, Search, Project, Run, Window, Help), a toolbar with various icons, and a status bar at the bottom. On the left, a Project Explorer shows a tree structure with folders like "Demo", "instance", "model", "Predeclared declarations", "properties", and "reports". On the right, an Outline pane shows the current diagram element "identifier.impl".

MASIW Graphical Editor (3)



Click on a triangle in the right top corner to open palette

MASIW Graphical Editor (4)

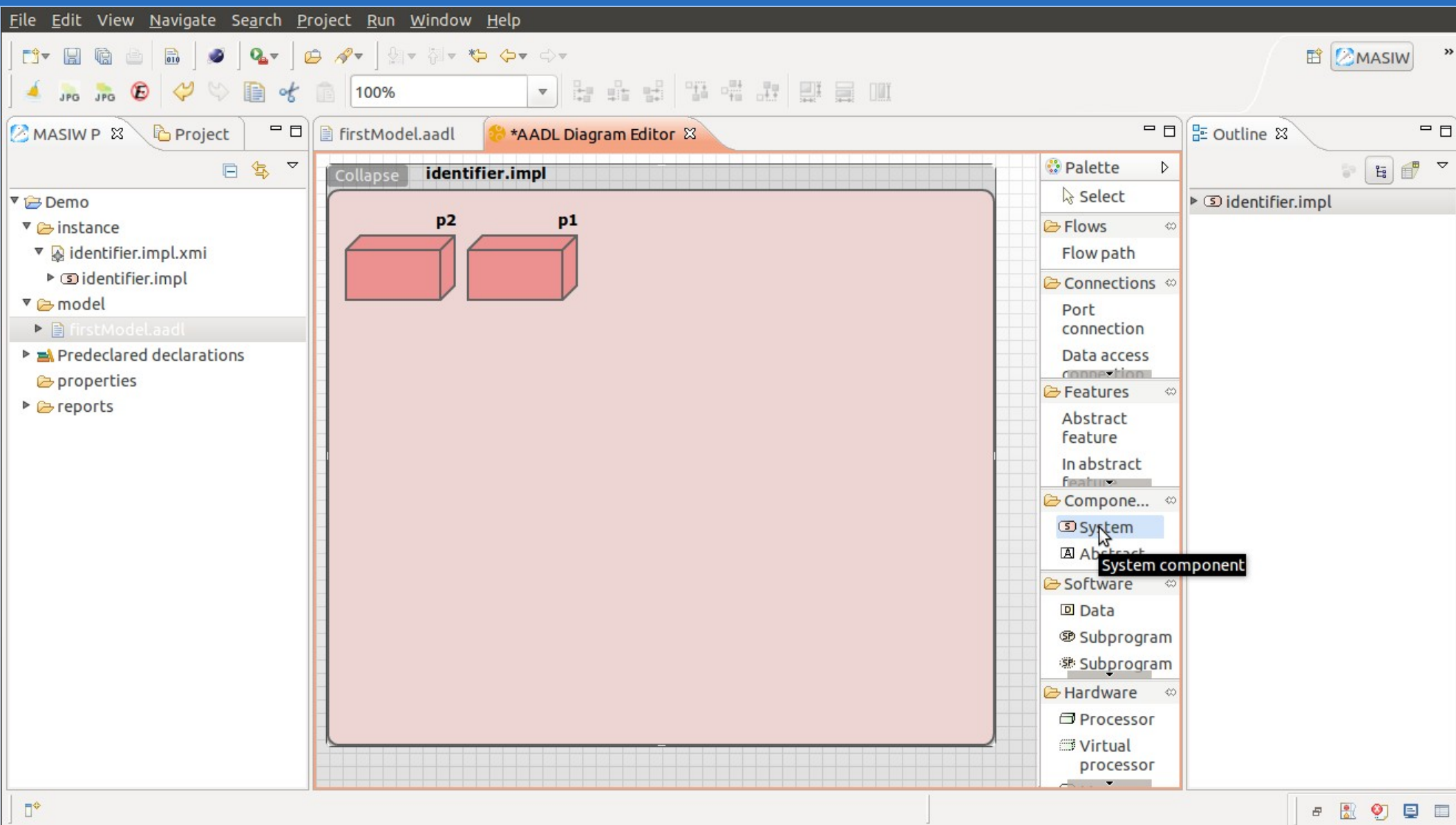


Select 'identifier.impl' system and use right bottom corner to resize it

MASIW Graphical Editor (5)

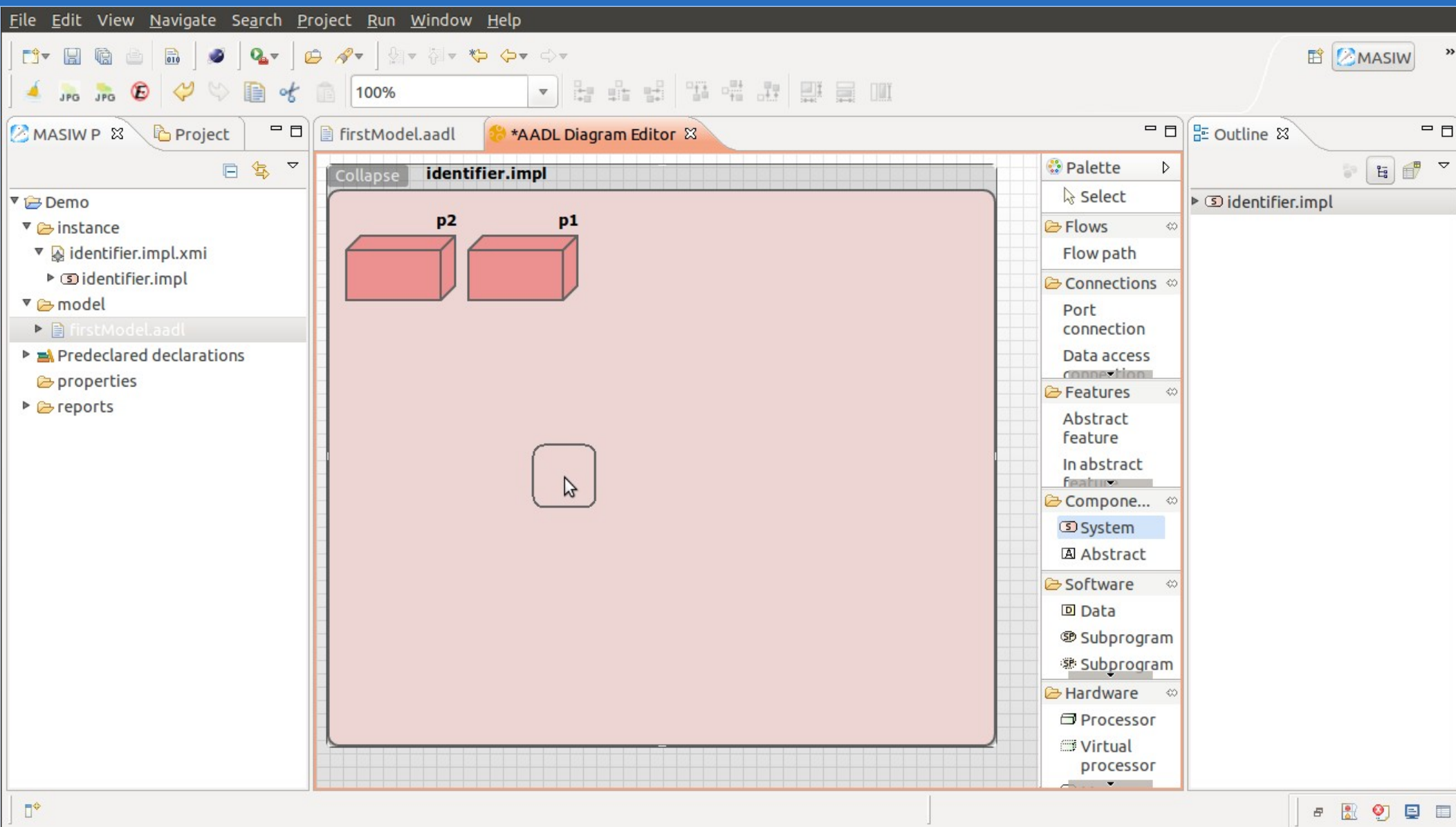
The screenshot displays the MASIW Graphical Editor interface. The main workspace shows an AADL diagram titled "identifier.impl" with a "Collapse" button. Inside the diagram, two red rectangular components are labeled "p2" and "p1". The interface includes a menu bar (File, Edit, View, Navigate, Search, Project, Run, Window, Help), a toolbar with various icons, and a status bar at the bottom. On the left, a Project Explorer shows a tree structure with folders like "Demo", "instance", "model", "Predeclared declarations", "properties", and "reports". On the right, a Palette contains various AADL elements such as "Select", "Flows", "Connections", "Features", "Component", "Software", and "Hardware". An Outline window on the far right shows the current diagram's structure.

MASIW Graphical Editor (6)



Click on system in palette to add a system

MASIW Graphical Editor (7)



Click on system in palette to add a system,
and then click on the system you want a new system is added to

MASIW Graphical Editor (8)

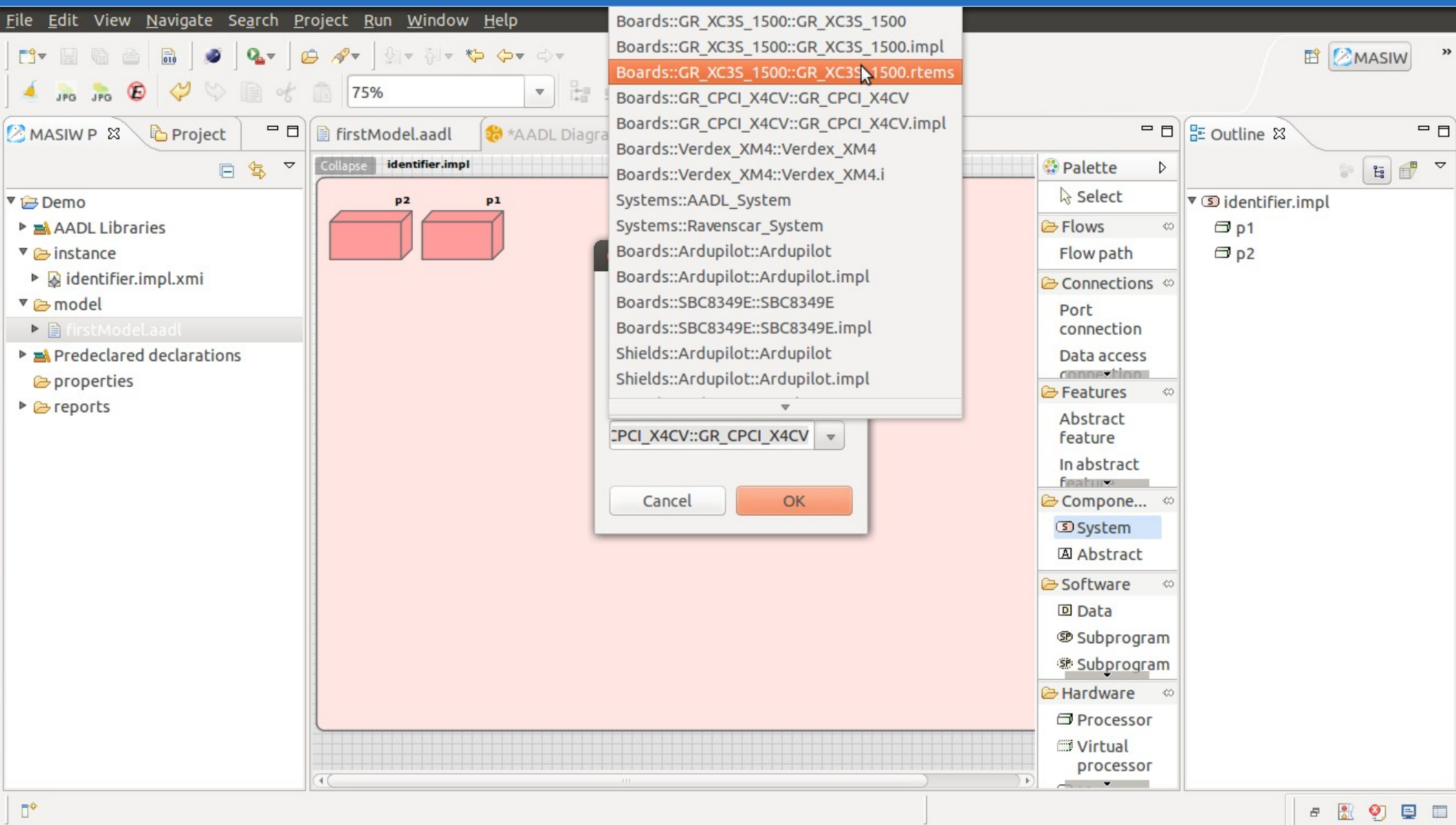
The screenshot displays the MASIW Graphical Editor interface. The main workspace shows a diagram with two red rectangular components labeled 'p2' and 'p1'. A dialog box is open in the center, titled 'Identifier' (partially obscured), with the following fields:

- Identificator:
- Component classifier:

The dialog box has 'Cancel' and 'OK' buttons at the bottom. The background diagram is dimmed. The interface includes a menu bar (File, Edit, View, Navigate, Search, Project, Run, Window, Help), a toolbar, a project browser on the left, a palette on the right, and an outline view on the far right.

Define an identifier for the system

MASIW Graphical Editor (9)



Define an identifier for the system,
and choose a classifier that a new system will extend

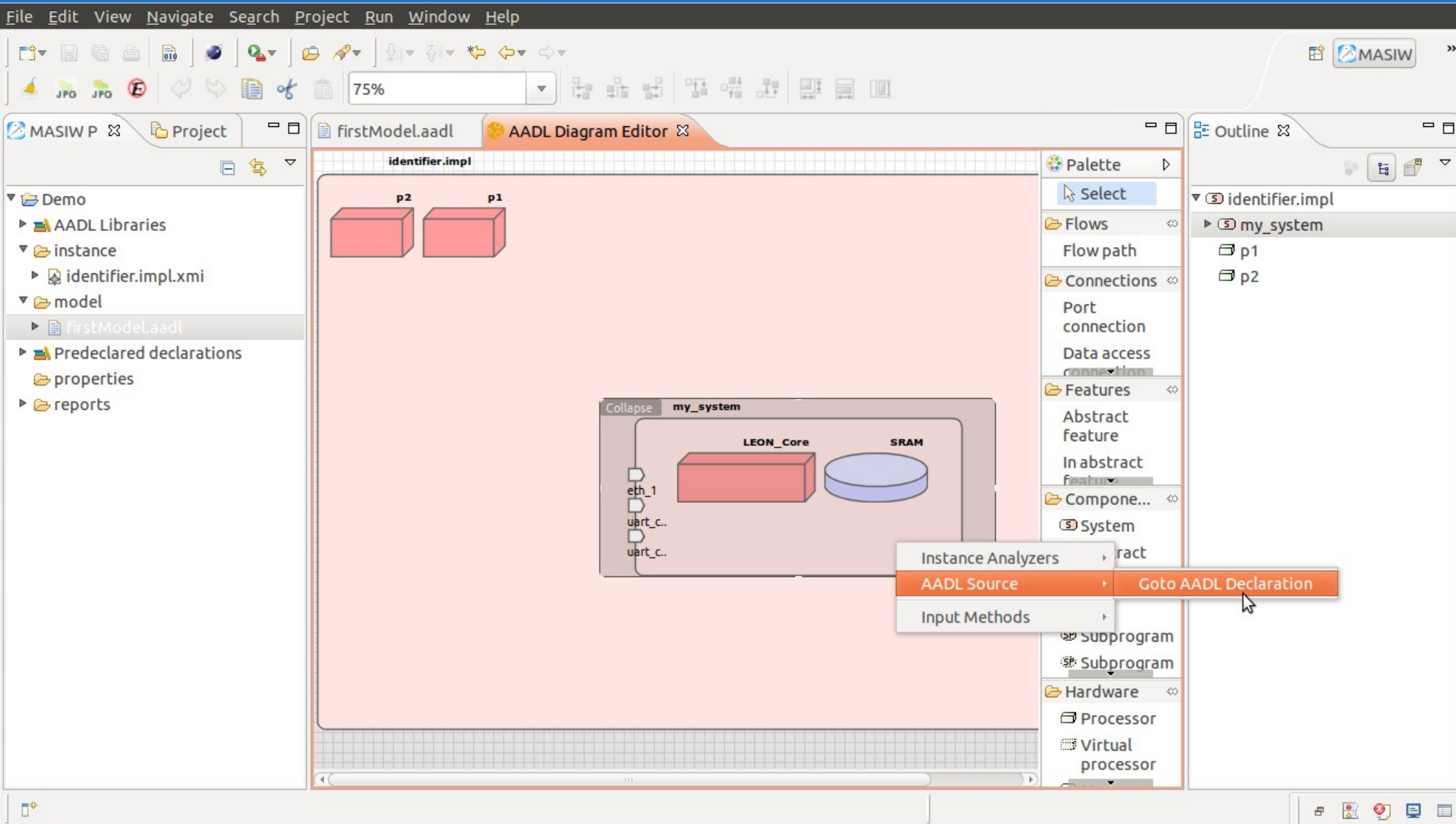
MASIW Graphical Editor (10)

The screenshot displays the MASIW Graphical Editor interface. The main workspace shows an AADL diagram titled "identifier.impl" with two red rectangular components labeled "p2" and "p1". A new system component, "my_system", is being added, represented by a grey-bordered box containing a red rectangular component labeled "LEON_Core" and a blue cylindrical component labeled "SRAM". The "my_system" component has three ports on its left side: "eth_1", "uart_c..", and "uart_c..".

The interface includes a menu bar (File, Edit, View, Navigate, Search, Project, Run, Window, Help), a toolbar with various icons, and a status bar at the bottom. The left sidebar shows a project tree with folders like "Demo", "AADL Libraries", "instance", "model", "Predeclared declarations", "properties", and "reports". The right sidebar shows an "Outline" view with a tree structure for "identifier.impl" containing "p1" and "p2". A "Palette" window is open on the right, listing various components such as "Select", "Flows", "Connections", "Features", "Software", and "Hardware".

A new system is added

MASIW Graphical Editor (11)



Context menu on figures in graphical editor allows to go to declaration of the components in textual form and to run analyzers

MASIW Graphical Editor (12)

The screenshot displays the MASIW Graphical Editor interface. The main window is titled "AADL Diagram Editor" and shows a code editor for the file "firstModel.aadl". The code defines a system implementation with subcomponents and properties. The line "my_system:SYSTEM my system 1.rtemsExt;" is highlighted in orange. The code is as follows:

```
WITH Boards::GR_XC3S_1500;  
WITH Boards::Verdex_XM4;  
WITH AAA;  
SYSTEM identifier  
END identifier;  
SYSTEM IMPLEMENTATION identifier.impl  
  Subcomponents  
    p1 : PROCESSOR PowerPC;  
    p2 : PROCESSOR PowerPC;  
    my_system:SYSTEM my system 1.rtemsExt;  
  Properties  
    AAA::X => 10  
      applies to p1,p2;  
    AAA::Y => 10  
      applies to p2;  
  annex real_specification {**  
    --@ Each processor has to have either X and Y both or none of the  
    theorem X_and_Y  
      FOREACH p IN PROCESSOR_SET DO  
        CHECK( Exists(p,"AAA::X") = Exists(p,"AAA::Y") );  
      end X_and_Y;  
    **};  
END identifier.impl;  
PROCESSOR PowerPC  
END PowerPC;  
SYSTEM my_system  
  EXTENDS Boards::Verdex_XM4::Verdex_XM4  
END my_system;  
SYSTEM my_system_1  
  EXTENDS Boards::GR_XC3S_1500::GR_XC3S_1500  
END my_system_1;  
SYSTEM IMPLEMENTATION my_system_1.rtemsExt  
  EXTENDS Boards::GR_XC3S_1500::GR_XC3S_1500.rtems  
END my_system_1.rtemsExt;  
END MyPackage;
```

The interface includes a menu bar (File, Edit, Navigate, Search, Project, Run, Window, Help), a toolbar, a project browser on the left showing a tree structure with "Demo", "AADL Libraries", "instance", "model", "Predeclared declarations", "properties", and "reports". The "model" folder is expanded to show "firstModel.aadl". An "Outline" view on the right shows a package structure with "MyPackage" and "AAA". The status bar at the bottom indicates "Writable", "Smart Insert", and "13:1".

Context menu on figures in graphical editor allows to go to declaration of the components in textual form and to run analyzers

Thank you!

Alexey Khoroshilov
khoroshilov@ispras.ru