Hazard Relation Diagrams

Definition and Evaluation

Bastian Tenbergen

Chapter 1 – Installation of the Tools

In Chapter 7 of the dissertation, the tools to support the manual modeling and creation of Hazard Relation Diagrams was explained. In the following subsections, installation of these tools are illustrated.

C1.1 Installation of the Enterprise Architect Profile

C1.1.1 Technical Prerequisites

The following technical prerequisites must be met:

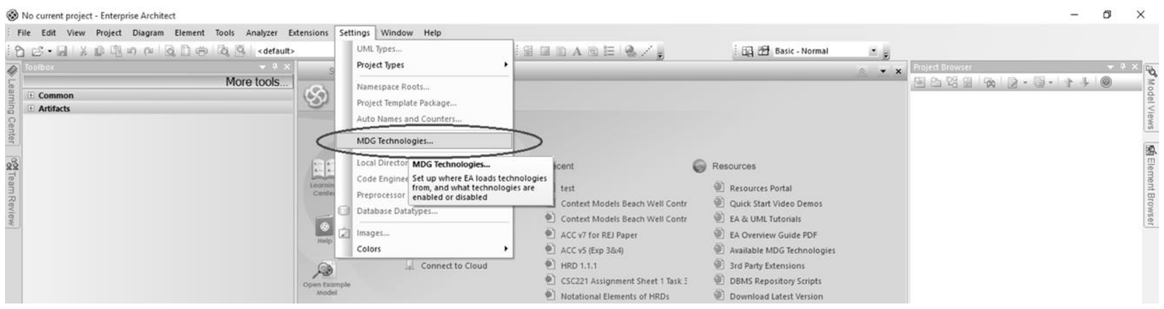
- Operating System: Microsoft Windows 10
- Enterprise Architect version 11.1
- Write access to at least one directory
- The Hazard Relation Diagram profile MDG Technology file for deployment or the profile EAP file for modification and deployment.

Available at: http://goo.gl/Bsaf4B

C1.1.2 Installing the Pre-Generated MDG Technology File

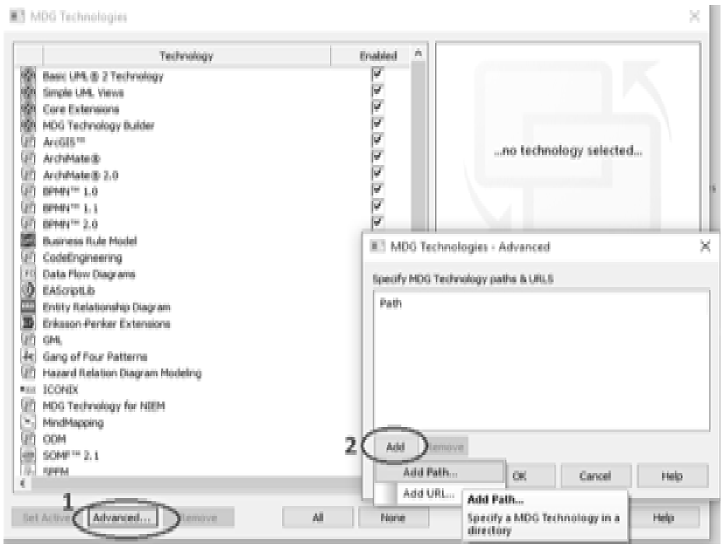
To deploy the pre-generated MDG Technology XML file containing the Hazard Relation Diagram profile from Section 7.2.2 of the dissertation, adhere to the following steps:

1. Obtain the profile archive and extract the file "Hazard Relation Diagram v.1.3.xml" from the archive. Save the XML file to a local directory.
2. Open Enterprise Architect. Select "Cancel" on the "Open Project…" dialog should it appear automatically on startup.
3. From the Menu Bar, select "Settings," then "MDG Technologies…"

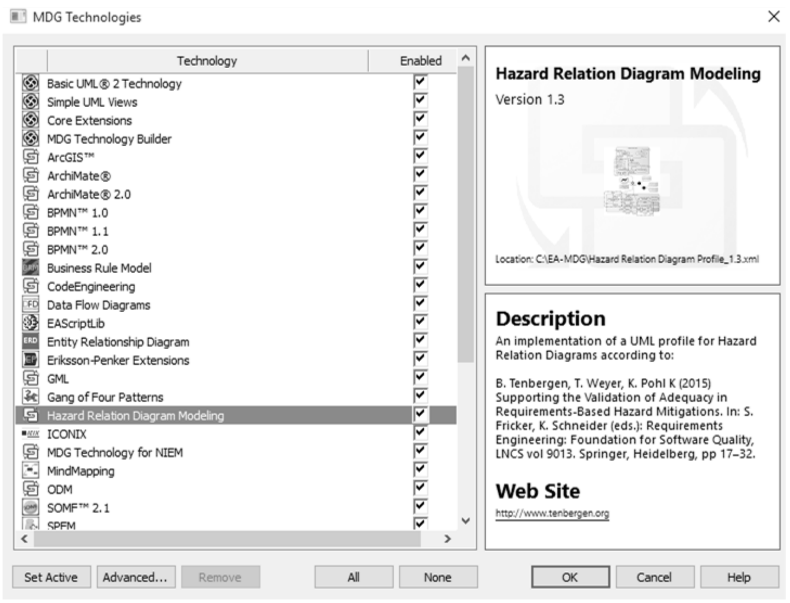


MDG Technologies...
Set up where EA loads technologies from, and what technologies are enabled or disabled

4. A window opens showing all currently installed MDG Technologies. First, select "Advanced…" and second, in the opening windows, select "Add" and "Add Path…"



5. Navigate to the directory where the "Hazard Relation Diagram v.1.3.xml" file was saved. Add this directory path and select "OK." Afterwards, the MDG Technologies windows reload and the profile is available for use (see Chapter 2, Section C2.1). Restarting Enterprise Architect might be necessary for all profile features to become available.



Technology	Enabled
Basic UML® 2 Technology	✓
Simple UML Views	✓
Core Extensions	✓
MDG Technology Builder	✓
ArcGIS™	✓
ArchiMate®	✓
ArchiMate® 2.0	✓
BPMN™ 1.0	✓
BPMN™ 1.1	✓
BPMN™ 2.0	✓
Business Rule Model	✓
CodeEngineering	✓
Data Flow Diagrams	✓
EAScriptLib	✓
Entity Relationship Diagram	✓
Eriksson-Penker Extensions	✓
GML	✓
Gang of Four Patterns	✓
Hazard Relation Diagram Modeling	✓
ICONIX	✓
MDG Technology for NIEM	✓
MindMapping	✓
ODM	✓
SOMF™ 2.1	✓
SPEM	✓

Hazard Relation Diagram Modeling
Version 1.3

Location: C:\EA-MDG\Hazard Relation Diagram Profile_1.3.xml

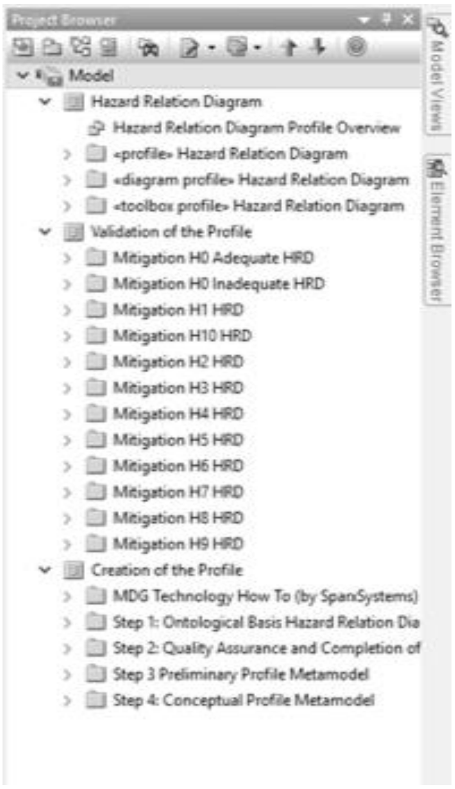
Description
An implementation of a UML profile for Hazard Relation Diagrams according to:

B. Tenbergen, T. Weyer, K. Pohl K (2015) Supporting the Validation of Adequacy in Requirements-Based Hazard Mitigations. In: S. Fricker, K. Schneider (eds.): Requirements Engineering: Foundation for Software Quality, LNCS vol 9013. Springer, Heidelberg, pp 17–32.

Web Site
http://www.tenbergen.org

C1.1.3 Modifying the Profile and Re-Generating the MDG Technology (Optional)

Anyone is encouraged to build upon this research and extend Hazard Relation Diagrams for different applications or with additional features. To do so, the source files of implemented profile from Section 7.2.2 of the dissertation is made available in the link given in Section C2.1.1. Obtain the archive, extract the file "Hazard Relation Diagram profile v1.3.eap," and open it using Enterprise Architect. After opening the file, the project explorer shows the following contents:

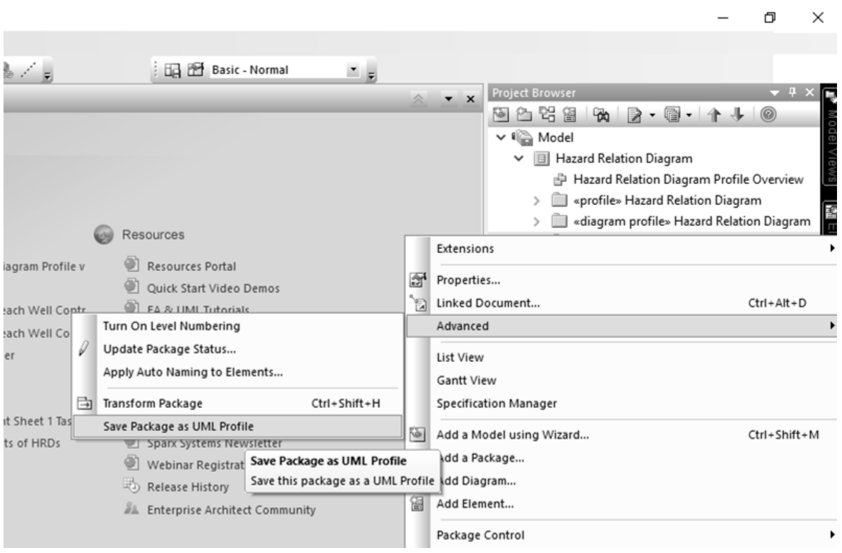


```
Model
  Hazard Relation Diagram
      Hazard Relation Diagram Profile Overview
    «profile» Hazard Relation Diagram
    «diagram profile» Hazard Relation Diagram
    «toolbox profile» Hazard Relation Diagram
  Validation of the Profile
    Mitigation H0 Adequate HRD
    Mitigation H0 Inadequate HRD
    Mitigation H1 HRD
    Mitigation H10 HRD
    Mitigation H2 HRD
    Mitigation H3 HRD
    Mitigation H4 HRD
    Mitigation H5 HRD
    Mitigation H6 HRD
    Mitigation H7 HRD
    Mitigation H8 HRD
    Mitigation H9 HRD
  Creation of the Profile
    MDG Technology How To (by SparxSystems)
    Step 1: Ontological Basis Hazard Relation Dia
    Step 2: Quality Assurance and Completion of
    Step 3 Preliminary Profile Metamodel
    Step 4: Conceptual Profile Metamodel
```

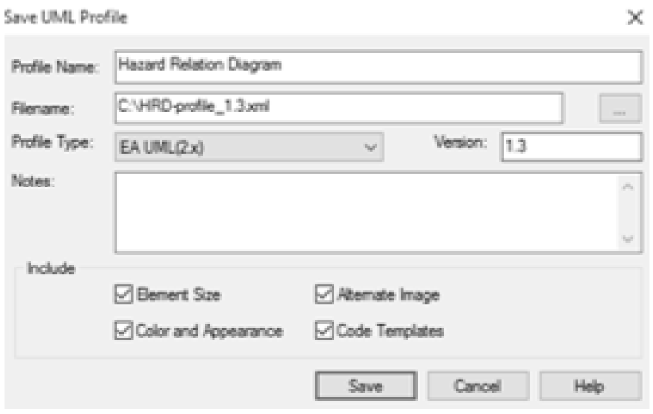
- The package **“Hazard Relation Diagram”** contains the implemented profile from Section 7.2.2 of the dissertation. To modify the profile implementation, consult reference [SparxSystems 2014] in the dissertation.
- The package **“Validation of the Profile”** contains various test diagrams, specifically the experimental stimuli used during the experimental evaluation from Part III of the dissertation.
- The package **“Creation of the Profile”** contains some help files on how to create and deploy MDGF Technologies provided by SparxSystems (subpackage “MDG Technology How To (by SparxSystems)” as well as one package for each step of the approach used to create the conceptual profile underlying the implementation (see Section 7.2 of the dissertation).

To create a new MDG Technology file for the Hazard Relation Diagram profile, adhere the following steps:

1. Save the <<profile>> package (cf. Section 7.2.2 of the dissertation) as a UML profile by right-clicking on the package, selecting “Advanced” and “Save Package as UML Profile.” Be sure to repeat this step for all packages


Basic - Normal
Project Browser
Model
Hazard Relation Diagram
Hazard Relation Diagram Profile Overview
«profile» Hazard Relation Diagram
«diagram profile» Hazard Relation Diagram
Resources
Resources Portal
Quick Start Video Demos
EA & UML Tutorials
Sparx Systems Newsletter
Webinar Registrat
Release History
Enterprise Architect Community
Turn On Level Numbering
Update Package Status...
Apply Auto Naming to Elements...
Transform Package Ctrl+Shift+H
Save Package as UML Profile
Save Package as UML Profile
Save this package as a UML Profile
Extensions
Properties...
Linked Document... Ctrl+Alt+D
Advanced
List View
Gantt View
Specification Manager
Add a Model using Wizard... Ctrl+Shift+M
Add a Package...
Add Diagram...
Add Element...
Package Control

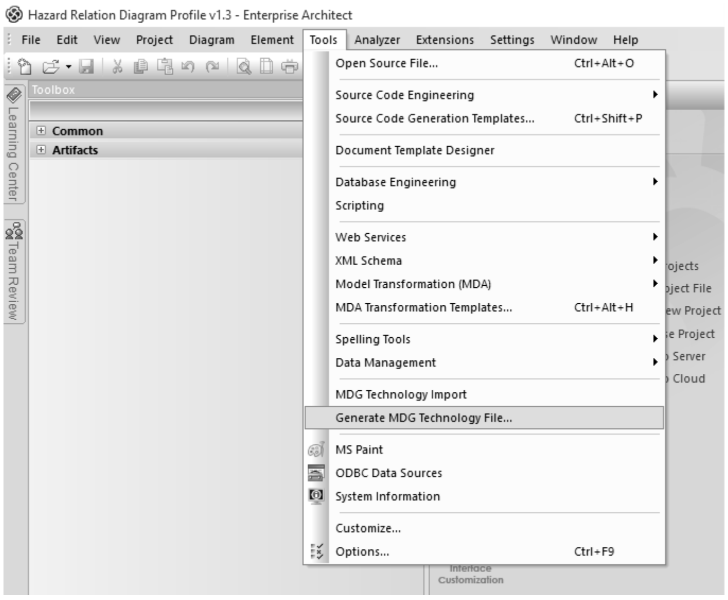
2. In the following dialog windows, use “Hazard Relation Diagram” as the profile name and store it to the local hard drive using some file name. Once finished, click “Save.”


Save UML Profile

Profile Name: Hazard Relation Diagram
Filename: C:\HRD-profile_1.3.xml
Profile Type: EA UML(2.x)
Version: 1.3
Notes:

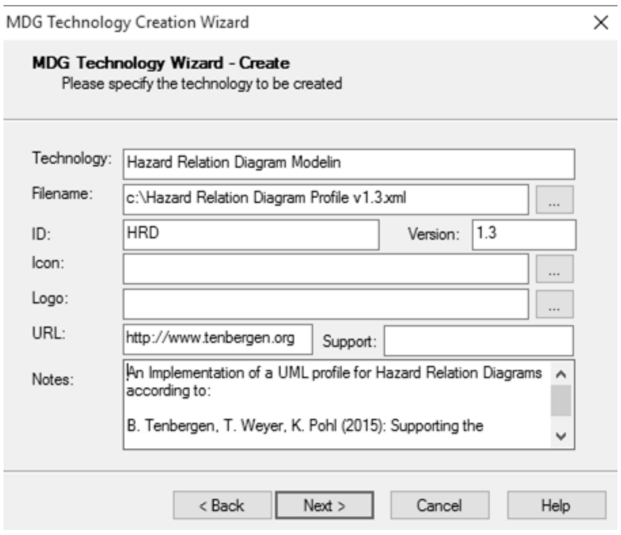
Include
- [x] Element Size
- [x] Alternate Image
- [x] Color and Appearance
- [x] Code Templates

3. Repeat Step 1 and Step 2 for the <<diagram profile>> and <<toolbox profile>> packages. Be sure to save each package to a different file name.
4. Once all three packages were saved as XML files, select “Tools” in the Menu Bar and select “Generate MDG Technology File…”. If this menu option is unavailable, open some EAP project file first.



Hazard Relation Diagram Profile v1.3 - Enterprise Architect
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Toolbox
Common
Artifacts
Learning Center
Team Review
Open Source File... Ctrl+Alt+O
Source Code Engineering
Source Code Generation Templates... Ctrl+Shift+P
Document Template Designer
Database Engineering
Scripting
Web Services
XML Schema
Model Transformation (MDA)
MDA Transformation Templates... Ctrl+Alt+H
Spelling Tools
Data Management
MDG Technology Import
Generate MDG Technology File...
MS Paint
ODBC Data Sources
System Information
Customize...
Options... Ctrl+F9

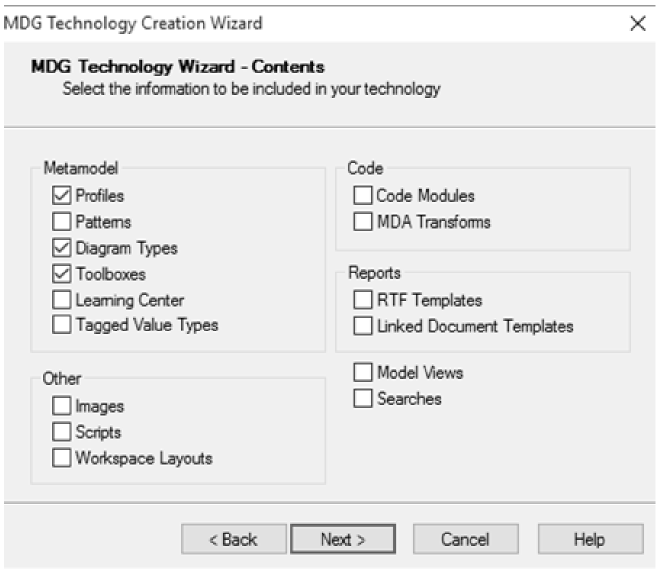
5. A MDG Technology generation wizard opens. Click "Next" on the first screen. On the following screen, select "Do not use MTS file for this technology" and click "Next." On the third screen, enter "Hazard Relation Diagram Modeling" in the field "Technology" and enter a path to a file under which the profile shall be saved in the field "Filename." Enter "HRD" in the field "ID" and chose a version number, and URL, and some release notes. Click "Next" when done.


MDG Technology Creation Wizard

MDG Technology Wizard - Create
Please specify the technology to be created

Technology:	Hazard Relation Diagram Modelin
Filename:	c:\Hazard Relation Diagram Profile v1.3.xml
ID:	HRD
Version:	1.3
Icon:	
Logo:	
URL:	http://www.tenbergen.org
Support:	
Notes:	An Implementation of a UML profile for Hazard Relation Diagrams according to: B. Tenbergen, T. Weyer, K. Pohl (2015): Supporting the

6. On the next screen, in the subsection "Metamodel," select "Profiles," "Diagram Types," and "Toolboxes." These are the technical Enterprise Architect profiles that were exported in Steps 1-3 above and indicate the same packages described in Section 7.2.2 of the dissertation. When done, click "Next."


MDG Technology Creation Wizard

MDG Technology Wizard - Contents
Select the information to be included in your technology

Metamodel
- ☑ Profiles
- ☐ Patterns
- ☑ Diagram Types
- ☑ Toolboxes
- ☐ Learning Center
- ☐ Tagged Value Types

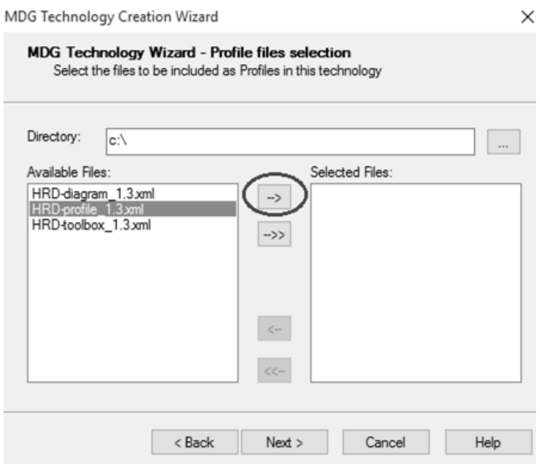
Code
- ☐ Code Modules
- ☐ MDA Transforms

Reports
- ☐ RTF Templates
- ☐ Linked Document Templates

- ☐ Model Views
- ☐ Searches

Other
- ☐ Images
- ☐ Scripts
- ☐ Workspace Layouts

7. On the next screen, the profiles to be added to the MDG Technology is added. Navigate to the directory indicated in Step 2. All exported XML files are shown in the list under "Available Files:" Select the profile file name indicated under Step 2 and click the arrow button "->". The file is moved into the "Selected Files:" compartment. Click "Next".

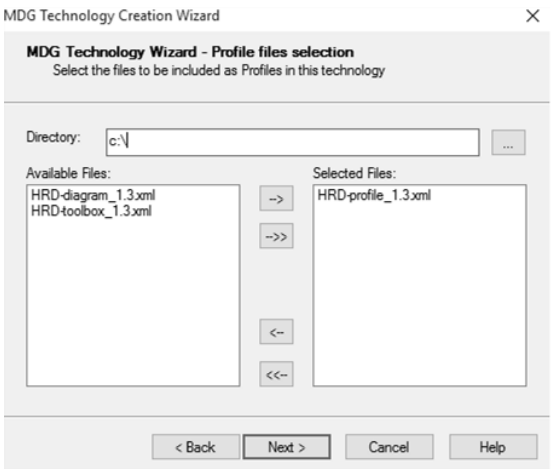

MDG Technology Creation Wizard

MDG Technology Wizard - Profile files selection
Select the files to be included as Profiles in this technology

Directory: c:\

Available Files:
HRD-diagram_1.3.xml
HRD-profile_1.3.xml
HRD-toolbox_1.3.xml

Selected Files:


MDG Technology Creation Wizard

MDG Technology Wizard - Profile files selection
Select the files to be included as Profiles in this technology

Directory: c:\

Available Files:
HRD-diagram_1.3.xml
HRD-toolbox_1.3.xml

Selected Files:
HRD-profile_1.3.xml

8. Repeat Step 7 for the diagram XML file and the toolbox XML file, respectively. Be sure to select the right one in the respective wizard screen.
9. On the final screen, a summary is shown. Click "Finish" and the MDG Technology is created. Deploy the new MDG Technology as outlined in Section C2.1.2.

C1.2 Installation of the Tool Prototype for the Automatic Generation of Hazard Relation Diagrams

C1.2.1 Technical Prerequisites

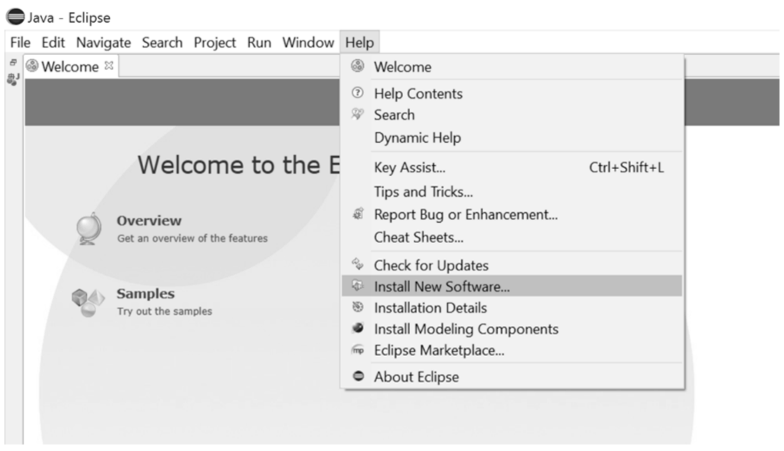
The following technical prerequisites must be met:

- Operating System:
  - Microsoft Windows XP or higher. Windows 10 recommended.
  - MacOS X v.10.11 El Capitan
  - Linux with Kernel version 4.4.1
- Eclipse Luna Modeling Package SR2., Available at: http://goo.gl/dePCw6
- Write access to at least one directory
- The Tool Prototype Archive. Available at: http://goo.gl/NqKLwu

C1.2.2 Installing Eclipse

To install the Eclipse distribution in preparation for the Tool Prototype as presented in Section 7.2 of the dissertation, adhere to the following steps:

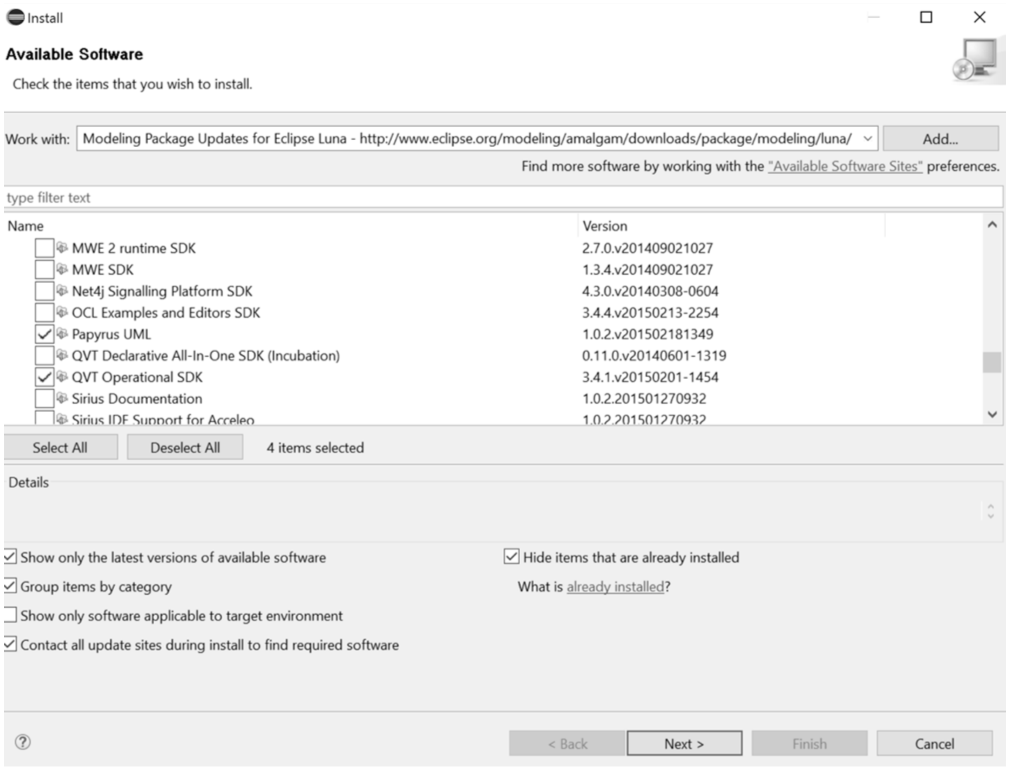
1. Obtain the Eclipse Luna Modeling Package and extract the archive contents to a local directory.
2. Start Eclipse and select some directory for Eclipse's workspace.
3. From the Eclipse menu bar, select "Help" and then "Install New Software



4. Under "Work with:," select "Modeling Package Updates for Eclipse Luna" and wait for the available package updates to load. Then, open the "Modeling" branch and select:

- EMF – Eclipse Modeling Framework Xcore SDK
- Graphical Modeling Framework (GMF) Tooling SDK
- Papyrus UML

- QVT Operational SDK



Install

Available Software

Check the items that you wish to install.

Work with: Modeling Package Updates for Eclipse Luna - http://www.eclipse.org/modeling/amalgam/downloads/package/modeling/luna/ Add...

Find more software by working with the "Available Software Sites" preferences.

type filter text

	Name	Version
☐	MWE 2 runtime SDK	2.7.0.v201409021027
☐	MWE SDK	1.3.4.v201409021027
☐	Net4j Signalling Platform SDK	4.3.0.v20140308-0604
☐	OCL Examples and Editors SDK	3.4.4.v20150213-2254
✓	Papyrus UML	1.0.2.v201502181349
☐	QVT Declarative All-In-One SDK (Incubation)	0.11.0.v20140601-1319
✓	QVT Operational SDK	3.4.1.v20150201-1454
☐	Sirius Documentation	1.0.2.201501270932
☐	Sirius IDE Support for Acceleo	1.0.2.201501270932

Select All Deselect All 4 items selected

Details

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

< Back Next > Finish Cancel

5. Select "Next >" and advance to the next screen. Select "Next >" again and accept the Eclipse user agreement on the final screen. When done, select "Finish". Eclipse installs all components, which might take some time and might require Eclipse to restart.

C1.2.3 Installing the Eclipse Plugins for the Tool Prototype

Once the Eclipse Modeling distribution is installed, the Eclipse Plugins for the Tool Prototype must be installed. To do so, adhere to the following steps:

1. Quit Eclipse, if it is currently running.
2. Obtain the Tool Prototype and extract the archive contents to a local directory.
3. From the "plugins" directory of the extracted archive contents, move all .jar files into the "plugins" directory of your local Eclipse installation directory.
4. Restart Eclipse. The Tool Prototype is now ready to be used.

C1.2.4 Modifying the Tool Prototype (Optional)

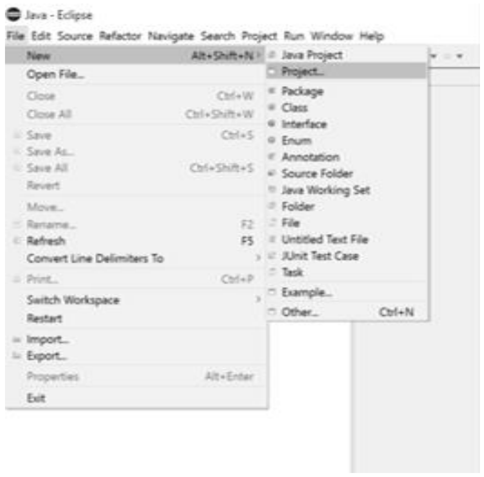
Anyone is encouraged to build upon this research and extend the tool prototype to create Hazard Relation Diagrams. This entails modifying the Ecore artifact type implementations as well as the QVTo transformation scripts (see Chapter 3). However, modifying the Ecore artifact type implementations is an involved process requiring a combination of Eclipse Ecore development as well as Eclipse Plugin development. Detailing how this can be achieved are beyond the scope of this dissertation. Resources on how to accomplish this, however, can be found at the Eclipse Ecore project website[1] as well as the Eclipse Plugin Development Environment project website[2]. Nevertheless, in order to be able to modify the artifact type implementations, it is necessary to first obtain Ecore source of the artifacts from the previously downloaded tool archive and importing the source files into a new Eclipse project, analogous to the way outlined in Section C2.2.5.

To modify the QVTo scripts, all that needs to be done is copy the scripts from Appendix D into Eclipse and modify the source code. Alternatively, the QVTo scripts can also be found in the ACC example project (see Section C2.2.5 as well as Section C3.2.1).

C1.2.5 Importing the ACC Example Project

Once the Eclipse Plugins for the Tool Prototype are installed, an example project can be imported. This example project features the Adaptive Cruise Control example from Section 2.1 of the dissertation that was used throughout this dissertation. To import the ACC example project, adhere to the following steps:

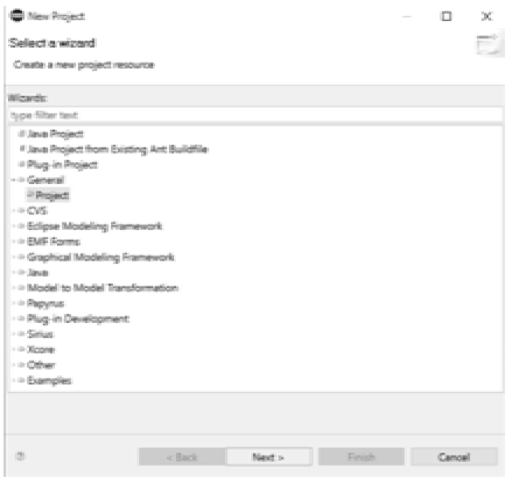
1. Obtain and store in a local directory the ACC example project, Available at: http://goo.gl/NqKLwu
2. Open Eclipse and create a new project by selecting "File" and then "New" and "Project…" from the menu bar.



3. On the next dialog, select "General" and "Project" under "Wizards" and click "Next". In the subsequent window, type in some project name and select "Finish".

[1] http://www.eclipse.org/ecoretools/

[2] http://www.eclipse.org/pde/

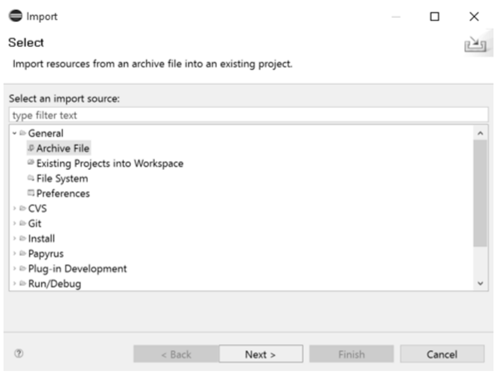

New Project

Select a wizard

Create a new project resource

Wizards:

4. Next, select “File” and then “Import…” from the menu bar. In the dialog window open the “General” tree in the field “Select an input source:” and chose “Archive File” and select “Next”.

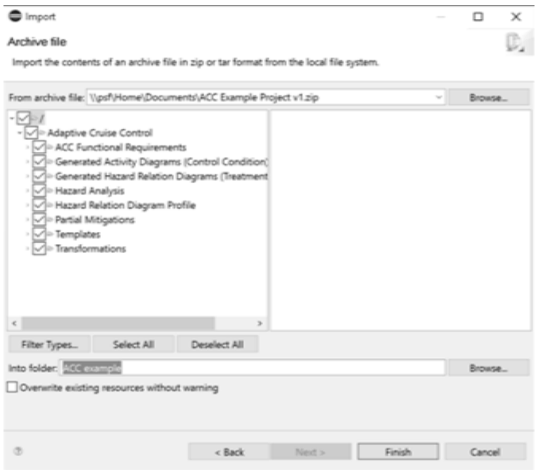

Import

Select

Import resources from an archive file into an existing project.

Select an import source:

5. On the next dialog, select the file stored to the local directory from Step 1 and select the project root. If desired, open the project root tree to verify that all components are there. Under “Into folder:,” select the project name indicated under Step 3.


Import

Archive file

Import the contents of an archive file in zip or tar format from the local file system.

From archive file: \\psf\Home\Documents\ACC Example Project v1.zip

Into folder: ACC example

Overwrite existing resources without warning

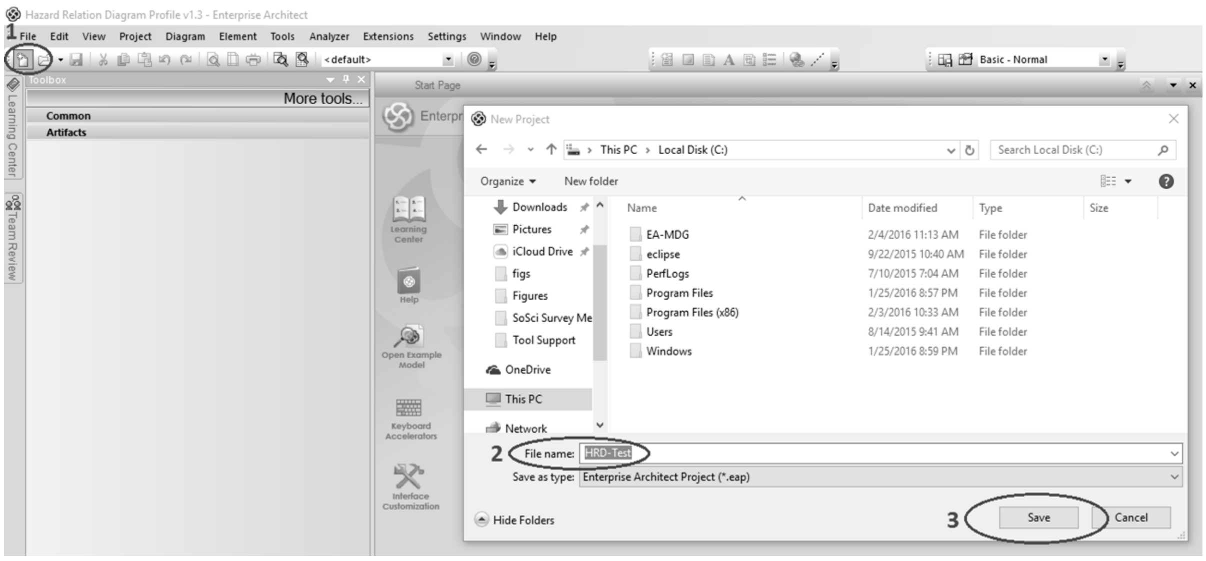
Chapter 2 – Using the Tools

In Chapter 7 of the dissertation, the tools to support the manual modeling and automatic creation of Hazard Relation Diagrams was explained. In the following subsections, it is explained, how these tools can be used.

C2.1 Using the Enterprise Architect Profile

In order to start manually creating Hazard Relation Diagrams using the Enterprise Architect Profile, it is necessary to first deploy the profile, as outlined in Section C2.1.2. Afterwards, start Enterprise Architect and adhere to the following steps:

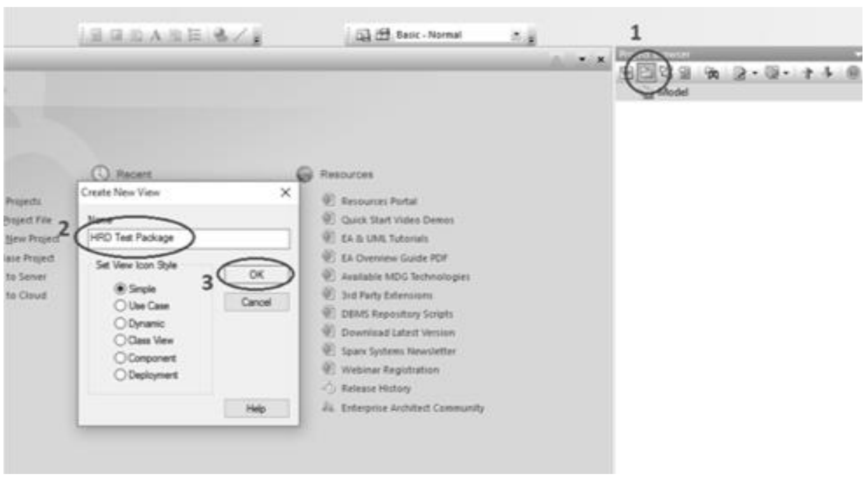
1. Start a new Project by clicking the "New Project" icon in upper left corner in the Menu Bar. Select a local directory and some file name. Click "Save."


Hazard Relation Diagram Profile v1.3 - Enterprise Architect
1
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Toolbox
More tools...
Common
Artifacts
Start Page
New Project
This PC > Local Disk (C:)
Organize
New folder
2
3
Save
Cancel
Hide Folders

File name: HRD-Test

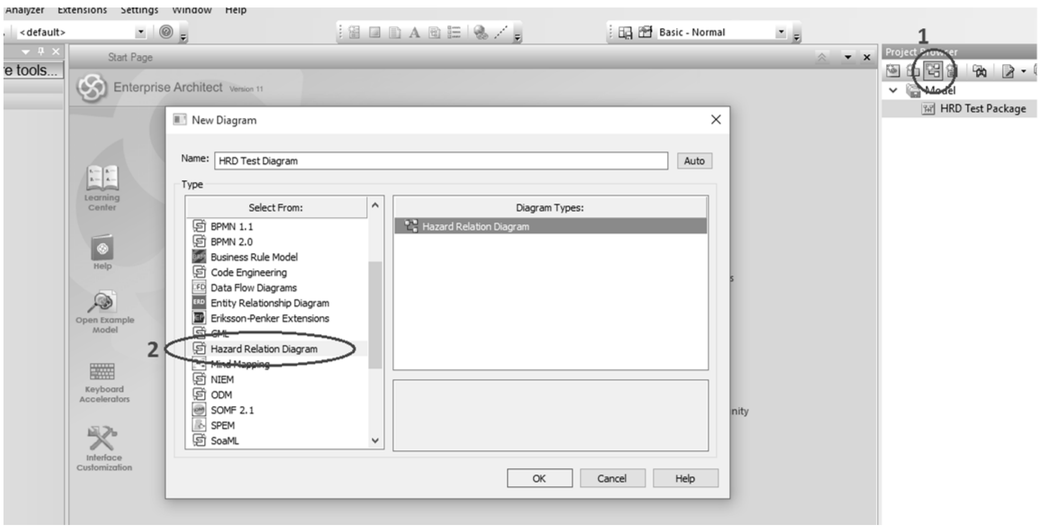
Save as type: Enterprise Architect Project (*.eap)

2. A new EAP file for the project is created. If the "Model Wizard" opens, click "Cancel." In the project explorer, create a new package inside the model using the "New Package" button. Select some name for the package and click "OK."



Name: HRD Test Package

3. A new package inside the model is created. Select the newly created package and create a new diagram using the “New Diagram” button to the right of the “New Package” button. Under “Select From,” scroll down to the “Hazard Relation Diagram” and select “Hazard Relation Diagram” in the field “Diagram Types:”. Add some custom name and click “OK.”



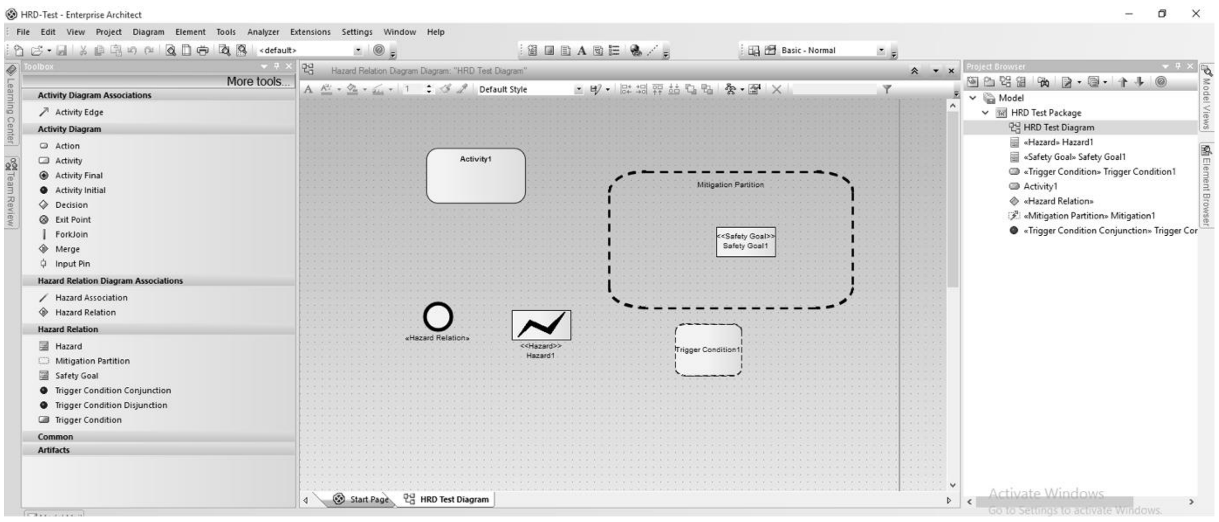
New Diagram

Name: HRD Test Diagram

Select From: Hazard Relation Diagram

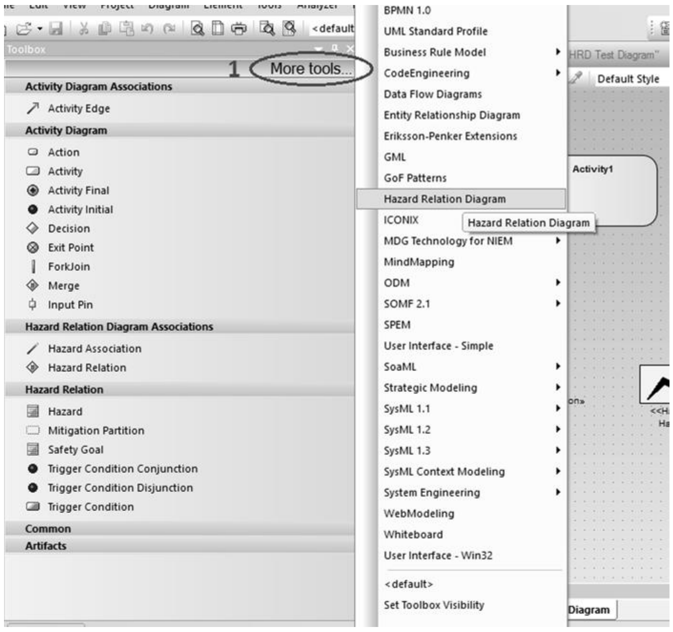
Diagram Types: Hazard Relation Diagram

4. A new Hazard Relation Diagram is created and opened. The appropriate toolbox will automatically open as well. Now modeling can begin by dragging the desired modeling elements from the toolbox on the left into the diagram area in the middle.



HRD-Test - Enterprise Architect
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Toolbox
More tools...
Activity Diagram Associations
Activity Edge
Activity Diagram
Action
Activity
Activity Final
Activity Initial
Decision
Exit Point
Fork/Join
Merge
Input Pin
Hazard Relation Diagram Associations
Hazard Association
Hazard Relation
Hazard Relation
Hazard
Mitigation Partition
Safety Goal
Trigger Condition Conjunction
Trigger Condition Disjunction
Trigger Condition
Common
Artifacts
Hazard Relation Diagram Diagram: "HRD Test Diagram"
Activity1
Mitigation Partition
<<Safety Goal>> Safety Goal1
«Hazard Relation»
<<Hazard>> Hazard1
Trigger Condition1
Start Page
HRD Test Diagram
Project Browser
Model
HRD Test Package
HRD Test Diagram
«Hazard» Hazard1
«Safety Goal» Safety Goal1
«Trigger Condition» Trigger Condition1
Activity1
«Hazard Relation»
«Mitigation Partition» Mitigation1
«Trigger Condition Conjunction» Trigger Cor

5. The toolbox can also be manually opened by clicking on “More tools…” and then selecting “Hazard Relation Diagram” from the list. This is useful, if the toolbox does not open automatically, or modeling elements from Hazard Relation Diagrams shall be added to other diagram types.



C2.2 Using the Tool Prototype for the Automatic Generation of Hazard Relation Diagrams

C2.2.1 Overview over the ACC Example Project

The steps from Section C2.2.3 will import the ACC example project into the local project created in the Eclipse workspace. The ACC example project consists of the following components:

- The directory **"ACC Functional Requirements,"** which contains the unaltered excerpt of the functional requirements specification of the Adaptive Cruise Control from Figure 2-1 of the dissertation as a UML activity diagram.
- The directory **"Generated Activity Diagrams (Control Condition),"** which contains the hazard-mitigating requirements used in the control conditions in the experiments (see Section 10.3 of the dissertation).
- The directory **"Generated Hazard Relation Diagrams (Treatment Condition),"** which contains the Hazard Relation Diagrams used in the control conditions in the experiments (see Section 10.3 of the dissertation).
- The directory **"Hazard Analysis,"** which contains the output of a Functional Hazard Analysis conducted on the ACC functional requirements in the file "Adaptive Cruise Control.functionalhazardanalysis". The file represents the artifact type "Hazard Analysis Result" and was used to extend the hazard-mitigating requirements from the directory "Generated Activity Diagrams (Control Condition)" into Hazard Relation

Diagrams in the directory “Generated Hazard Relation Diagrams (Treatment Condition)”.

- The directory **“Hazard Relation Diagram profile,”** which contains the Ecore implementation from of the conceptual UML profile from Section 7.2 of the dissertation and defines the stereotypes needed to extend hazard-mitigating requirements into Hazard Relation Diagrams.
- The directory **“Partial Mitigations,”** which contain the mitigation templates for each hazard in the file Adaptive Cruise Control.functionalhazardanalysis. The partial mitigations represent the artifact type “Partial Mitigation” and was used to generate the hazard-mitigating requirements in the directory “Generated Activity Diagrams (Control Condition)” from the unaltered ACC functional requirements in the directory “ACC Functional Requirements.“

The directory **“Templates,”** which contain empty templates for both the artifact type “Partial Mitigation” as well as the artifact type “Hazard Analysis Result.”

The directory **“Transformations,”** which contain the QVTo script implementations discussed in Section 7.3.1 of the dissertation and which can be found in Chapter 3. These can be used to execute the approach outlined throughout this dissertation, either to repeat the results or to apply on novel examples.

C2.2.2 Documenting Hazard Analysis Results

As outlined in Section C2.2.1, the file “empty.functionalhazardanalysis” represents an implementation of the Ecore artifact type “Hazard Analysis Result.” Using this template, the results of a hazard analysis can be documented as follows:

1. Double-click on the file “empty.functionalhazardanalysis” in the directory “Templates” from the ACC example project.
2. A “Resource Set” in the editor frame of Eclipse is shown. Open the “platform” tree to reveal the “FHA” root node.
3. Right-click on the “FHA” root node to create a “New Child.“ Two types of children can be added: hazard-inducing requirements and hazards. Children of type “Hazard-inducing Requirement” are the requirements from the unaltered functional requirements specification, i.e. an activity from some activity diagram that cause some hazard. Children of type “Hazard” represent the hazards identified by the Hazard Analysis for some hazard-inducing requirement in question.

4. For each child, some unique ID (this could be a universally unique ID or some freely choosable human-readable unique identifier) as well as some human-readable designation (e.g., a description of the hazard or the name of the hazard-inducing activity) must be identified. To do so, right-click the child, and select “Show Properties View”. Eclipse will then open the properties pane for Ecore objects (and unless closed by the user, keep it open for the next Ecore object).
5. For children of type “Hazard” the property field “Relates to” must be specified. This can be any previously specified hazard-inducing requirement (i.e. some sibling of that hazard that is under the same FHA root node and not also a hazard).
6. Furthermore, for each child of type “Hazard” sub-children can be added. This specifically entails the subchild “Safety Goal” and “Trigger Conditions”. Safety goals can be added by simply adding a new child of type “Safety Goal” to some hazard node and specifying an ID and a description in the property view. In the current implementation of the tool prototype, only one safety goal may be added per hazard (see Section C2.2.5).
7. Repeat Step 6 for a set of trigger conditions. Children of type “Trigger Conditions” may contain further children, specifically atomic children of type “Trigger Condition,” trigger condition conjunctions (type “andnode”) and trigger condition disjunctions (type “ornode”). It is to note that the IDs given to the atomic trigger conditions must be referenced by the trigger condition conjunctions and disjunctions in order to create a binary trigger condition tree (see Section C2.2.5).
8. Repeat Steps 3 through 7 for each hazard-inducing requirement and each hazard. Then finished, the result is a completely specified Hazard Analysis Result that can be used to create Hazard Relation Diagrams (see Section C2.2.4), similar to the file “Adaptive Cruise Control.functionalhazardanalysis” in the directory “Hazard Analysis” of the ACC example project.

C2.2.3 Documenting Partial Mitigations

As outlined in Section C2.2.1, the file “empty.hazardmitigation” represents an implementation of the Ecore artifact type “Partial Mitigation”. Using this template, the transformation steps to create UML activity diagrams containing hazard-mitigating requirements from UML activity diagrams containing hazard-inducing requirements can be specified as follows:

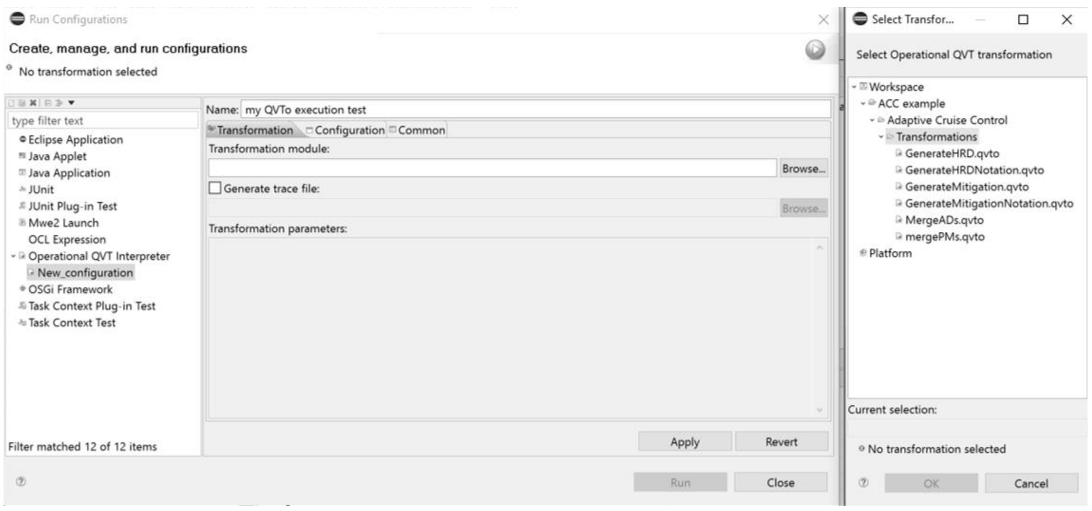
1. Double-click on the file “empty.hazardmitigation” in the directory “Templates” from the ACC example project.

2. A “Resource Set” in the editor frame of Eclipse is shown. Open the “platform” tree to reveal the “Mitigation List” root node. This represents the root node containing all transformation steps.
3. Right-click on the “Mitigation List” root node and select “Show Properties View”. Eclipse will then open the properties pane for Ecore objects (and unless closed by the user, keep it open for the next Ecore object). Under “Activity Diagram name” indicate the human-readable designation of the activity diagram containing hazard-inducing requirements. Under “Hazard ID,” select the ID of the hazard to be mitigated. This ID should correspond to an ID from some specified Hazard Analysis result using the respective Ecore artifact type, as outlined above. For the file “UML model file,” select the .uml file for the activity diagram containing hazard-inducing requirements.
4. Right-click on the “Mitigation List” root node to create a “New Child.“ Each child represents a specific transformation step, i.e. insertion, removal, or substitution of some activity diagram element. The order to added children specifies the order in which the transformation steps are executed. Be sure to document the correct IDs as specified in the activity diagram containing hazard-inducing requirements.
5. Repeat Step 4 for each necessary transformation step. When finished, the result is a completely specified partial mitigation that can be used to create a UML activity diagram containing hazard-mitigation requirements (see Section C2.2.4), similar to the partial mitigations in the in the directory “Partial Mitigations” of the ACC example project.

C2.2.4 Executing Transformations using Imported Run Configurations

Section 7.3.1 of the dissertation detailed what QVTo transformation scripts have been implemented. In this section, it is discussed how these can be executed. This is done by means of Eclipse’s runtime configurations, which specify the input parameters needed to execute the QVTo scripts and output artifacts created by them. To create a new run configuration, follow these steps:

1. From the Eclipse menu bar, select “Run” and “Run Configurations…” to open the dialog to create a new QVTo run configuration.
2. On the left side, select “Operational QVT Interpreter,” right-click and select “New.“ This will create a new run configuration for QVT Operational Mappings. Then, on the right side, assign a human-readable name under “Name”.
3. Under “Transformation Module” click on “Browse…” and find the desired QVTo script in the directory “Transformations” of the ACC example project.

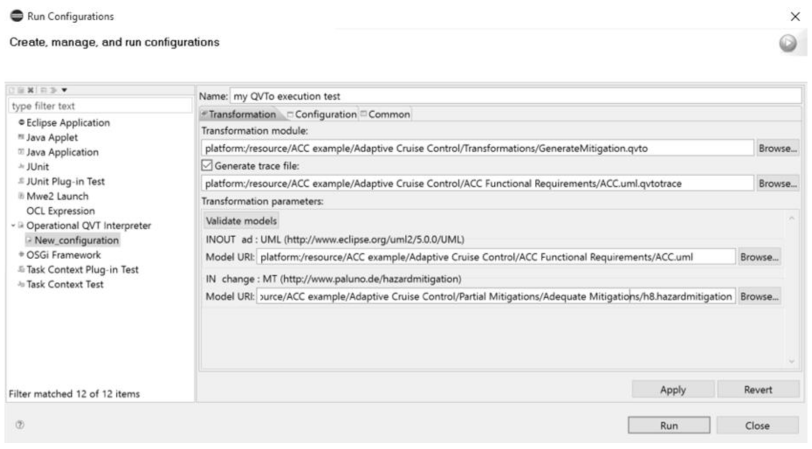


4. Depending on which QVTo script is selected, run configuration is amended by several Transformation parameters in the lower portion of the run configuration dialog. Input parameters are marked by the keyword “IN,” while output parameters are marked by the keyword “OUT.“ Parameters marked with the keyword “INOUT” represent input artifacts that are altered by the QVTo script. Each parameter bears the URI used to uniquely identify the type of the expected parameter. The following subsections illustrate how the transformation parameters must be setup in order to be able to execute each QVTo script. For a detailed discussion on the artifact dependencies, please refer to Section 7.3.1 of the dissertation.
5. Once all parameters are assigned, click “Apply” to save the run configuration and “Run” to execute the script.

Transformation Parameters for “GenerateMitigation.qvto”

From the available QVTo scripts in the “Transformations” directory of the ACC example project, select “**GenerateMitigation.qvto**”. The following transformation parameters must be configured:

- **INOUT ad: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the activity diagram containing hazard-inducing requirements. Please note, that the original activity diagram is changed.
- **IN change: MT (http://www.paluno.de/hazardmitigation):** The partial mitigation containing the transformation steps used to create the activity diagram containing hazard-mitigating requirements.


Run Configurations

Create, manage, and run configurations

Name: my QVTo execution test

Transformation module:
platform:/resource/ACC example/Adaptive Cruise Control/Transformations/GenerateMitigation.qvto

☑ Generate trace file:
platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.uml.qvtotrace

Transformation parameters:

Validate models

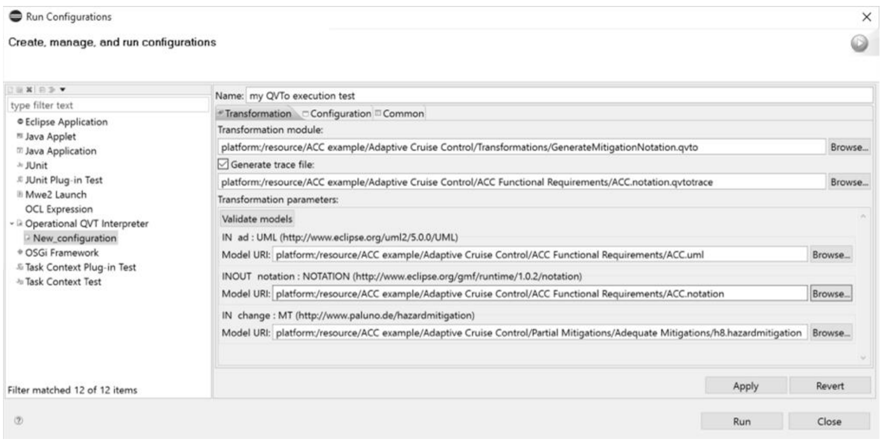
INOUT ad : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.uml

IN change : MT (http://www.paluno.de/hazardmitigation)
Model URI: ›urce/ACC example/Adaptive Cruise Control/Partial Mitigations/Adequate Mitigations/h8.hazardmitigation

Transformation Parameters for “GenerateMitigationNotation.qvto”

From the available QVTo scripts in the “Transformations” directory of the ACC example project, select “**GenerateMitigationNotation.qvto**”. The following transformation parameters must be configured:

- **IN ad: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the activity diagram containing hazard-mitigating requirements created through execution of the QVTo script “GenerateMitigation.qvto”.
- **INOUT notation: NOTATION (http://www.eclipse.org/gmf/runtime/1.0.2/…):** The .notation file for the activity diagram containing hazard-inducing requirements. Please note, that the original activity diagram notation is changed to conform to the changed activity diagram created through execution of the QVTo script “GenerateMitigation.qvto”.
- **IN change: MT (http://www.paluno.de/hazardmitigation):** The partial mitigation containing the transformation steps used to create the activity diagram containing hazard-mitigating requirements.


Run Configurations

Create, manage, and run configurations

Name: my QVTo execution test

Transformation module:
platform:/resource/ACC example/Adaptive Cruise Control/Transformations/GenerateMitigationNotation.qvto

☑ Generate trace file:
platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.notation.qvtotrace

Transformation parameters:

Validate models

IN ad : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.uml

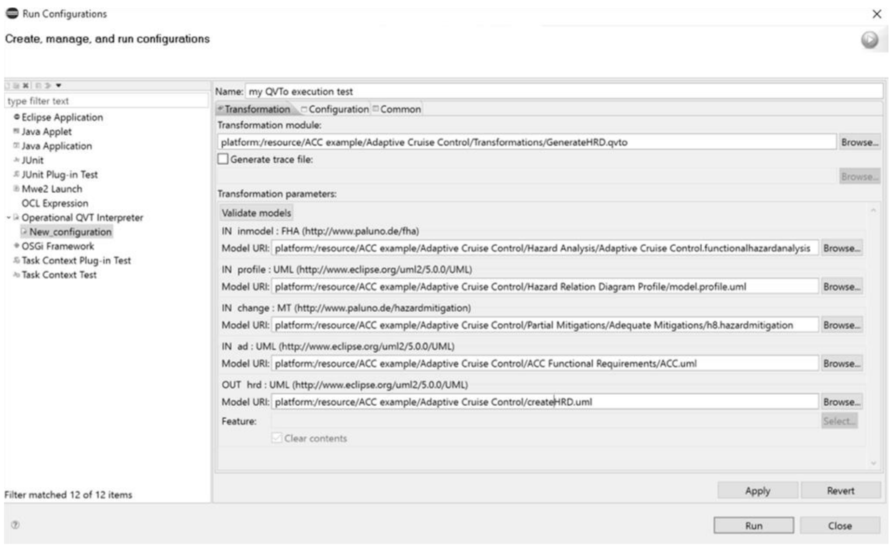
INOUT notation : NOTATION (http://www.eclipse.org/gmf/runtime/1.0.2/notation)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.notation

IN change : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/Partial Mitigations/Adequate Mitigations/h8.hazardmitigation

Transformation Parameters for “GenerateHRD.qvto”

From the available QVTo scripts in the “Transformations” directory of the ACC example project, select “**GenerateHRD.qvto**”. The following transformation parameters must be configured:

- **IN inmodel: FHA (http://www.paluno.de/FHA):** The Hazard Analysis Result table containing the results of hazard analysis, documented as outlined in Section C2.2.2.
- **IN profile: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the Hazard Relation Diagram Ecore profile.
- **IN change: MT (http://www.paluno.de/hazardmitigation):** The partial mitigation containing the transformation steps used to create the activity diagram containing hazard-mitigating requirements.
- **IN ad: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the activity diagram containing the hazard-mitigating requirements created using the QVTo script “GenertateMitigation.qvto”.
- **OUT hrd: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the newly created Hazard Relation Diagram.


Run Configurations
Create, manage, and run configurations

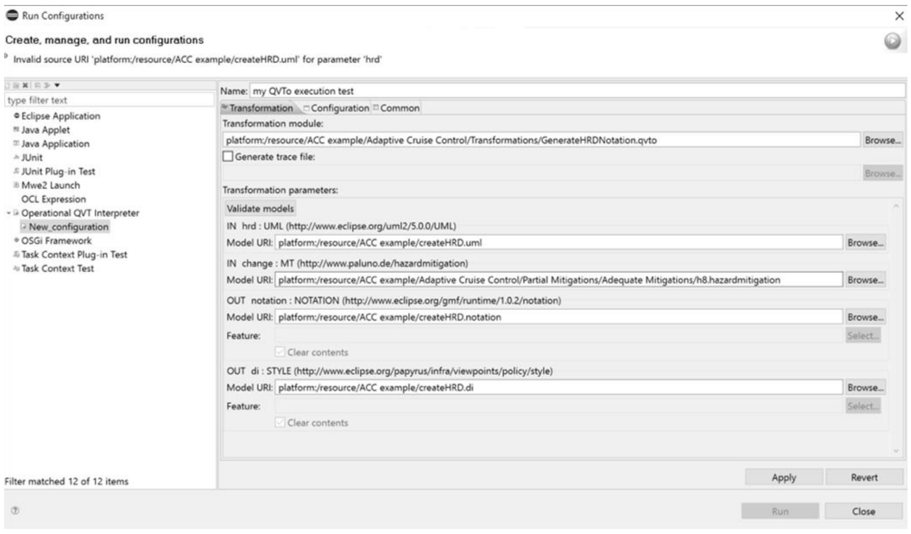
Name: my QVTo execution test
Transformation module: platform:/resource/ACC example/Adaptive Cruise Control/Transformations/GenerateHRD.qvto
Generate trace file: ☐
Transformation parameters:
IN inmodel : FHA (http://www.paluno.de/fha)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/Hazard Analysis/Adaptive Cruise Control.functionalhazardanalysis
IN profile : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/Hazard Relation Diagram Profile/model.profile.uml
IN change : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/Partial Mitigations/Adequate Mitigations/h8.hazardmitigation
IN ad : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/ACC Functional Requirements/ACC.uml
OUT hrd : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/createHRD.uml
Feature:
Clear contents ✓
Filter matched 12 of 12 items

Transformation Parameters for “GenerateHRDNotation.qvto”

From the available QVTo scripts in the “Transformations” directory of the ACC example project, select “**GenerateHRDNotation.qvto**”. The following transformation parameters must be configured:

- **IN hrd: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the newly created Hazard Relation Diagram using the QVTo script “GenerateHRD.qvto”.

- **IN change: MT (http://www.paluno.de/hazardmitigation):** The partial mitigation containing the transformation steps used to create the activity diagram containing hazard-mitigating requirements.
- **OUT notation: NOTATION (http://www.eclipse.org/gmf/runtime/1.0.2/…):** The .notation file to be created for the newly created Hazard Relation Diagram.
- **OUT di: STYLE (http://www.eclipse.org/papyrus/infra/viewpoints/policy/style):** The .di file to be created for the newly created Hazard Relation Diagram.


Run Configurations

Create, manage, and run configurations

Invalid source URI 'platform:/resource/ACC example/createHRD.uml' for parameter 'hrd'

type filter text
- Eclipse Application
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Mwe2 Launch
- OCL Expression
- Operational QVT Interpreter
  - New_configuration
- OSGi Framework
- Task Context Plug-in Test
- Task Context Test

Filter matched 12 of 12 items

Name: my QVTo execution test

Transformation | Configuration | Common

Transformation module: platform:/resource/ACC example/Adaptive Cruise Control/Transformations/GenerateHRDNotation.qvto

Generate trace file:

Transformation parameters:

Validate models

IN hrd : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/createHRD.uml

IN change : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/Adaptive Cruise Control/Partial Mitigations/Adequate Mitigations/h8.hazardmitigation

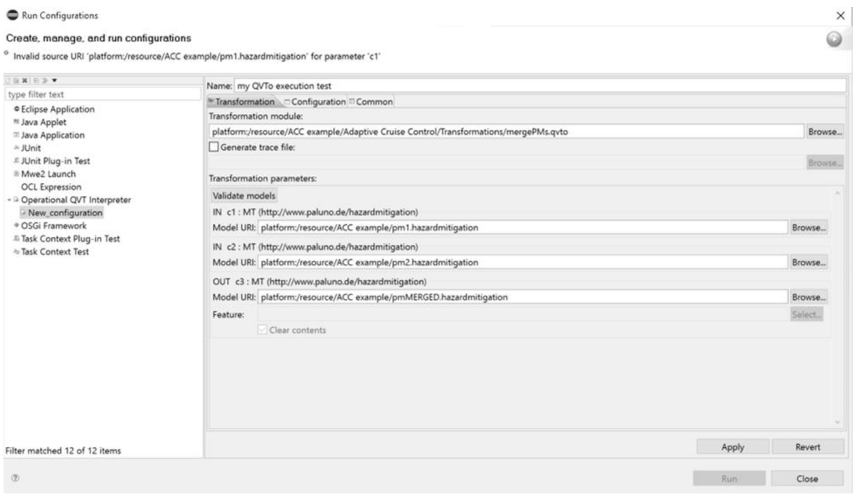
OUT notation : NOTATION (http://www.eclipse.org/gmf/runtime/1.0.2/notation)
Model URI: platform:/resource/ACC example/createHRD.notation
Feature:
Clear contents

OUT di : STYLE (http://www.eclipse.org/papyrus/infra/viewpoints/policy/style)
Model URI: platform:/resource/ACC example/createHRD.di
Feature:
Clear contents

Transformation Parameters for "mergePMs.qvto"

From the available QVTo scripts in the "Transformations" directory of the ACC example project, select "**mergePMs.qvto**". The following transformation parameters must be configured:

- **IN c1: MT (http://www.paluno.de/hazardmitigation):** The first partial mitigation to be merged with the second.
- **IN c2: MT (http://www.paluno.de/hazardmitigation):** The second partial mitigation to be merged with first.
- **OUT c3: MT (http://www.paluno.de/hazardmitigation):** The merged partial mitigations.



Run Configurations

Create, manage, and run configurations

Invalid source URI 'platform:/resource/ACC example/pm1.hazardmitigation' for parameter 'c1'

Name: my QVTo execution test

Transformation module: platform:/resource/ACC example/Adaptive Cruise Control/Transformations/mergePMs.qvto

Generate trace file: ☐

Transformation parameters:

IN c1 : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/pm1.hazardmitigation

IN c2 : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/pm2.hazardmitigation

OUT c3 : MT (http://www.paluno.de/hazardmitigation)
Model URI: platform:/resource/ACC example/pmMERGED.hazardmitigation

Feature:

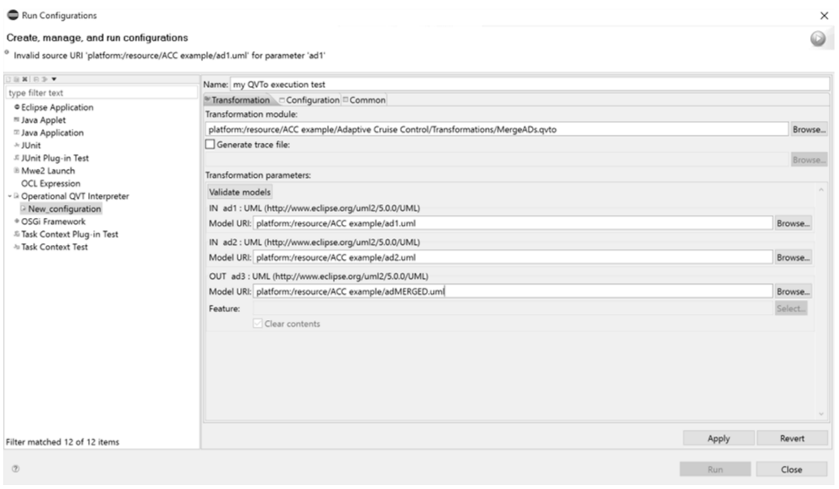
Clear contents

Filter matched 12 of 12 items

Transformation Parameters for "mergeADs.qvto"

From the available QVTo scripts in the "Transformations" directory of the ACC example project, select "**mergeADs.qvto**". The following transformation parameters must be configured:

- **IN ad1: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the first activity diagram to be merged with the second.
- **IN ad2: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the second activity diagram to be merged with the first.
- **OUT ad3: UML (http://www.eclipse.org/uml2/5.0.0/UML):** The .uml file for the merged activity diagrams.



Run Configurations

Create, manage, and run configurations

Invalid source URI 'platform:/resource/ACC example/ad1.uml' for parameter 'ad1'

Name: my QVTo execution test

Transformation module: platform:/resource/ACC example/Adaptive Cruise Control/Transformations/MergeADs.qvto

Generate trace file: ☐

Transformation parameters:

IN ad1 : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/ad1.uml

IN ad2 : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/ad2.uml

OUT ad3 : UML (http://www.eclipse.org/uml2/5.0.0/UML)
Model URI: platform:/resource/ACC example/adMERGED.uml

Feature:

Clear contents

Filter matched 12 of 12 items

C2.2.5 Technical Limitations of the Tool Prototype

Due to the strict enforcement of UML's syntactic and semantic rules, some concessions to the implementations were necessary. In the following, these are briefly explained. It is to note, however, that none of these limitations impair the semantics of Hazard Relation Diagrams nor do they impair the validity of the approach outlined in this dissertation.

- **Missing support for multiple safety goals.** Although Section 5.4 of the dissertation acknowledges that multiple safety goals might be conceived per hazard, the approach thus far only supports specifying one safety goal in the Ecore artifact type "Hazard Analysis Result." This limitation was necessary in order to be able to unambiguously append the safety goal to the Hazard Relation Diagram for the safety goal's hazard. As outlined in Section 5.4 of the dissertation, if conceptual mitigation alternatives or safety goal alternatives shall be evaluated, distinct Hazard Relation Diagrams must be created for each alternative combination of hazard, safety goal, and conceptual mitigation. To accommodate this, using this tool prototype, it is necessary to specify a new Hazard Analysis Result table for each alternative.
- **Cumbersome Binary Tree Structure for Trigger Condition Conjunctions and Trigger Condition Disjunctions.** As outlined in Section C2.2.2, Trigger Conditions in the Ecore artifact type "Hazard Analysis Result" contain children of type "Trigger Condition," "andnode," and "ornode". While andnodes represent trigger condition conjunctions, ornodes represent trigger condition disjunctions. However, atomic trigger conditions that are subordinate to an andnode or ornode are not represented in the same manner, but as siblings on the same level, and must be "linked" by specifying the IDs of the subordinate trigger conditions in the property fields of the andnode or ornode. This was a deliberate design choice allowing trigger conditions to be reused in other hazards (as some operational condition could result in multiple hazards, see Section 5.4 of the dissertation).
- **Surrounding "Activity" for Hazard Relation Diagrams.** As a side effect of the strict enforcement of the UML infrastructure, modeling elements foreign to the diagram type may not be added in Eclipse UML2 tools. Furthermore, unlike in Enterprise Architect, new diagram types can only be defined by extending existing diagram types, thereby preventing using static-structural modeling elements (e.g., the n-ary association "Hazard Relation") in dynamic diagrams (i.e. Hazard Relation Diagrams as extensions of activity diagrams). As a workaround, the tool prototype will create a semantically empty activity to wrap the modeling elements of Hazard Relation Diagrams and display them in the manner consistent with the ontological foundations as well as the visual notation (see Section 5.3 of the dissertation).
- **Manually Assignment of the Visual Notation for Hazard Relation Diagram Stereotypes.** Unlike Enterprise Architect, support for stereotypes in UML profiles is limited to the assigning stereotype designations and tagged values to modeling

elements. Changing the visual style must be done manually for each modeling element rather than automatically.

Chapter 3 – QVT Operational Mapping Scripts

C3.1 GenerateMitigation.qvto

```
modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';
modeltype MT uses 'http://www.paluno.de/hazardmitigation';

transformation mitigationtoad(inout ad:UML, in change:MT );

main() {
  var remAE:= change.objectsOfType(RemoveActivityEdge);
  remAE->forEach(r){ r.deleteActivityEdge()};
  var remA:= change.objectsOfType(RemoveActivity);
  remA->forEach(r){  r.deleteActivity()};
  var remP:= change.objectsOfType(RemovePin);
  remP->forEach(r){  r.deletePin()};
  var remCN:= change.objectsOfType(RemoveControlNode);
  remCN->forEach(r)  { r.deleteControlNode()};
  var subA:=change.objectsOfType(SubstituteActivity);
  subA->forEach(s) { s.replaceActivity()  };
  var subP:=change.objectsOfType(SubstitutePin);
  subP->forEach(s){  s.replacePin()};
  change.objectsOfType(InsertActivity)->map addActivity();
  change.objectsOfType(InsertPin)->map addPin();
  change.objectsOfType(InsertControlNode)->map addControlNode();
  var subCN:=change.objectsOfType(SubstituteControlNode);
  subCN->forEach(s){ s.replaceControlNode()};
  change.objectsOfType(InsertActivityEdge)->map addActivityEdge();
  var subAE:=change.objectsOfType(SubstituteActivityEdge);
  subAE->forEach(s){ s.replaceActivityEdge()};
--add substistuted and added actions to parent activity
  ad.objectsOfType(Activity)->forEach(a){
    ad.objectsOfType(OpaqueAction)->forEach(oa){a.ownedNode+=oa};
    ad.objectsOfType(ObjectFlow)->forEach(of){a.edge+=of};
    ad.objectsOfType(ActivityParameterNode)->forEach(pa){a.ownedNode+=pa};
    ad.objectsOfType(ControlNode)->forEach(cn){a.ownedNode+=cn};
  };
  deleteunusedpins();
}

mapping inout InsertActivity::addActivity(): OpaqueAction when{self.Activityexists()=false} {
  name:=self.activityName;
}

mapping inout InsertActivityEdge::addActivityEdge() when{self.Edgeexists()=false} {
  var objectflow:= object ObjectFlow {
   name:=self.Message;
   if getIDofActivity(self.SourceName)=null then {
     if getIDofControlNode(self.SourceName)=null then source:=getIDofPin(self.SourceName)
      else source:=getIDofControlNode(self.SourceName) endif;
  } else {
       var outputport:= object OutputPin{};
       getIDofActivity(self.SourceName).outputValue+=outputport; source:=outputport;
  } endif;
  if getIDofActivity(self.targetName)=null then {
  if getIDofControlNode(self.targetName)=null then target:=getIDofPin(self.targetName)
  else target:=getIDofControlNode(self.targetName) endif;
  } else {
      var inputport:=object InputPin{};
      getIDofActivity(self.targetName).inputValue+=inputport; target:=inputport;
  } endif;

  if self.Guard.toString()!="null" then {
    var guardof:LiteralBoolean= object LiteralBoolean{};
    if self.Guard="true" then guardof.value:=true
    else guardof.value:=false endif;
    guardof.name:=self.Guard; guard:=guardof;
  } else{ } endif;
}

mapping inout InsertPin::addPin():ActivityParameterNode when{self.Pinexists()=false} {
  name:=self.pinName
}

mapping inout InsertControlNode::addControlNode() when{self.ControlNodeexists()=false} {
  if self.nodeType="Fork" then   {
    controlnode:ControlNode:= object ForkNode  { name:=self.NodeName };
  } else if self.nodeType="Join" then {
    var  controlnode:ControlNode:= object JoinNode {
      name:=self.NodeName
    };
  } else if self.nodeType="Merge" then {
      var  controlnode:ControlNode:= object MergeNode  { name:=self.NodeName };
  } else if self.nodeType="Decision" then {
      var  controlnode:ControlNode:= object DecisionNode { name:=self.NodeName };
  } endif endif endif endif;
}

helper RemoveActivity::deleteActivity() {
  var activities:=ad.objectsOfType(OpaqueAction);
  activities->forEach(a) {
  if self.activityName = a.name then
    ad.removeElement(a)
  endif;  }
}

helper RemovePin::deletePin() {
  var pins:=ad.objectsOfType(ActivityParameterNode);
  pins->forEach(p) {
  if self.pinName = p.name then
    ad.removeElement(p)
  endif; }
}

helper RemoveActivityEdge::deleteActivityEdge() {
  var messages1:=ad.objectsOfType(ObjectFlow);
   messages1->forEach(m) {
    if m.source.name=self.sourceName or
m.source.outputisPinofActivity(getIDofActivity(self.sourceName))then
      if self.targetName = m.target.name or
m.target.inputisPinofActivity(getIDofActivity(self.targetName)) then
        if ((m.name.toString()="null"  and self.message.toString()="")  or
self.message=m.name) then
```

```
          if ((m.guard.toString()="null"  and self.Guard.toString()="null") or self.Guard =
m.guard.name) then {
            if getIDofActivity(self.sourceName)!=null then ad.removeElement(m.source)endif;
            if getIDofActivity(self.targetName)!=null then ad.removeElement(m.target)endif;
            ad.removeElement(m);
            log("message deleted");
          } endif
      endif
    endif
  endif; };
}

helper RemoveControlNode::deleteControlNode() {
  if self.nodeType="Fork" then {
    var forknodes:=ad.objectsOfType(ForkNode);
    forknodes->forEach(n) {
      if n.name=self.NodeName then ad.removeElement(n) endif;
    };
  } else if self.nodeType="Decision" then {
      var decisionnodes:=ad.objectsOfType(DecisionNode);
      decisionnodes->forEach(n) { if n.name=self.NodeName then ad.removeElement(n) endif; };
      } else if self.nodeType="Merge" then {
          var mergenodes:=ad.objectsOfType(MergeNode);
          mergenodes->forEach(n) { if n.name=self.NodeName then ad.removeElement(n) endif; };
    }else if self.nodeType="Join" then {
      var joinnodes:=ad.objectsOfType(JoinNode);
      joinnodes->forEach(n)  { if n.name=self.NodeName then ad.removeElement(n) endif; };
    } endif endif endif endif;
}

helper SubstituteActivity::replaceActivity() {
  var activities:=ad.objectsOfType(OpaqueAction);
  activities->forEach (a)  {
    if self.oldActivityName= a.name then a.name:= self.newActivityName endif;
  }
}

helper SubstituteActivityEdge::replaceActivityEdge() {
  log("oldname "+ self.oldMessage.toString());
  log("newname "+ self.newMessage.toString());
  log("oldguard "+ self.oldGuard.toString());
  log("newguard "+ self.newGuard.toString());
  log("oldsource "+ self.oldSourceName);
  log("newsource "+ self.newSourceName.toString());
  log("oldtarget "+ self.oldTargetName.toString());
  log("newtarget "+ self.newTargetName.toString());
  var sourceissub:=isSubstituted(self.oldSourceName);
  var targetissub:=isSubstituted(self.oldTargetName);
  log(sourceissub.toString());
  log(targetissub.toString());
  Var objects:=ad.objectsOfType(ObjectFlow);
  objects->forEach(o)   {
--   var sourceissub:=isSubstituted(self.oldSourceName);
--  Var targetissub:=isSubstituted(self.oldTargetName);
    if ((o.name.toString()="null"  and self.oldMessage.toString()="") or
o.name=self.oldMessage) then
      if o.source.name=self.oldSourceName or
o.source.outputisPinofActivity(getIDofActivity(self.oldSourceName)) or sourceissub then
```

```
      if o.target.name = self.oldTargetName or
o.target.inputisPinofActivity(getIDofActivity(self.oldTargetName)) or targetissub then
        if ((o.guard.toString()="null"  and self.oldGuard.toString()="null") or
o.guard.name=self.oldGuard)  then {
          log(o.name.toString() + " found");
          if o.source.outputisPinofActivity(getIDofActivity(self.oldSourceName))=true then
ad.removeElement(o.source) endif;
          if o.target.inputisPinofActivity(getIDofActivity(self.oldTargetName))=true then
ad.removeElement(o.target) endif;
          o.name:=self.newMessage;
          if (self.oldGuard.toString()!="null" and self.newGuard.toString()!="null")
then{o.guard.name:=self.newGuard;}
          else if (self.oldGuard.toString()="null" and self.newGuard.toString()!="null")
then{var guardof:LiteralBoolean=object LiteralBoolean{name:=self.newGuard; value:=true};}
        else{ } endif
     endif;
     o.guard.name:=self.newGuard;
     if getIDofActivity(self.newSourceName)=null then {
      if getIDofControlNode(self.newSourceName)=null then
o.source:=getIDofPin(self.newSourceName)
      else o.source:=getIDofControlNode(self.newSourceName) endif;
    } else {
        var outputport:= object OutputPin{};
getIDofActivity(self.newSourceName).outputValue+=outputport; o.source:=outputport;
    } endif;
    if getIDofActivity(self.newTargetName)=null then {
      if getIDofControlNode(self.newTargetName)=null then
o.target:=getIDofPin(self.newTargetName)
      else o.target:=getIDofControlNode(self.newTargetName) endif;
    } else {
      var inputport:=object InputPin{};
getIDofActivity(self.newTargetName).inputValue+=inputport; o.target:=inputport;
    } endif;
    } endif endif endif endif;
  };
}

helper SubstitutePin::replacePin() {
  var pins:=ad.objectsOfType(ActivityParameterNode);
  pins->forEach(p) {
    if p.name= self.oldPinName then p.name:=self.newPinName endif;
  }
}

helper SubstituteControlNode::replaceControlNode() {
  var nodes:=ad.objectsOfType(ControlNode);
  nodes->forEach(n) {
    if n.name=self.oldNodeName then {
      ad.removeElement(n);
      if self.newNodeType="Fork" then {
        var  controlnode:ControlNode:= object ForkNode { name:=self.newNodeName };
      } else if self.newNodeType="Join" then {
        var  controlnode:ControlNode:= object JoinNode{ name:=self.newNodeName };
      }   else if self.newNodeType="Merge" then {
        var  controlnode:ControlNode:= object MergeNode { name:=self.newNodeName };
      }   else if self.newNodeType="Decision" then {
        var  controlnode:ControlNode:= object DecisionNode { name:=self.newNodeName };
      }   endif endif endif endif
  } endif;
  }
}

helper getIDofActivity(mystring:String):OpaqueAction {
  var activities:=ad.objectsOfType(OpaqueAction); var opaque:OpaqueAction:=null;
  activities->forEach(a) { if a.name=mystring then {return opaque:=a; break }endif; };
  return opaque;
}

helper getIDofPin(mystring:String):ActivityParameterNode {
  var pins:=ad.objectsOfType(ActivityParameterNode); var pin:ActivityParameterNode:=null;
  pins->forEach(p) { if p.name=mystring then {return pin:=p; break }endif; };
  return pin;
}

helper getIDofControlNode(mystring:String):ControlNode {
  var nodes:=ad.objectsOfType(ControlNode); var node:ControlNode:=null;
  nodes->forEach(n) {   if n.name=mystring then {return node:=n; break } endif; };
  return node;
}

helper ActivityNode::outputisPinofActivity(OA:OpaqueAction):Boolean {
  var exist:=false; var ports:=ad.objectsOfType(OutputPin);
  ports->forEach(p) {
    if self=p then if OA.output->includes(p)=true  then exist:=true endif endif;
  };
  return exist;
}

helper ActivityNode::inputisPinofActivity(OA:OpaqueAction):Boolean {
  var exist:=false; var ports:=ad.objectsOfType(InputPin);
  ports->forEach(p) {
    if self=p then if OA.inputValue->includes(p)=true  then exist:=true endif endif;
  };
  return exist;
}

helper InsertActivity::Activityexists():Boolean {
  var exists:=false; var activities:=ad.objectsOfType(OpaqueAction);
  activities->forEach(a) {
    if self.activityName = a.name then {exists:=true; break;}endif;
  };
  return exists;
}

helper InsertPin::Pinexists():Boolean {
  var exists:=false; var pins:=ad.objectsOfType(ActivityParameterNode);
   pins->forEach(p) {
    if self.pinName = p.name then {exists:=true; break;}endif;
  };
  return exists;
}

helper InsertActivityEdge::Edgeexists():Boolean {
  var exists:=false; var messages1:=ad.objectsOfType(ObjectFlow);
   messages1->forEach(m) {
    if self.SourceName = m.source.name then
```

```
      if self.targetName = m.target.name then
        if self.Message=m.name then
          if self.Guard = m.guard.name then {exists:=true; break; } endif
        endif
      endif
    endif;
  };
  return exists;
}

helper InsertControlNode::ControlNodeexists():Boolean {
  var exists:=false;
  if self.nodeType="Fork" then {
    var forknodes:=ad.objectsOfType(ForkNode);
     forknodes->forEach(n)  { if n.name=self.NodeName then {exists:=true; break;} endif;};
  }else if self.nodeType="Decision" then {
    var decisionnodes:=ad.objectsOfType(DecisionNode);
    decisionnodes->forEach(n) {
      if n.name=self.NodeName then {exists:=true; break;} endif; };
  } else if self.nodeType="Merge" then {
    var mergenodes:=ad.objectsOfType(MergeNode);
    mergenodes->forEach(n) {
      if n.name=self.NodeName then{exists:=true; break;} endif; };
    } else if self.nodeType="Join" then {
      var joinnodes:=ad.objectsOfType(JoinNode);
      joinnodes->forEach(n)  {
        if n.name=self.NodeName then {exists:=true; break;} endif; };
  } endif endif endif endif;
  return exists;
}

helper isSubstituted(str:String):Boolean {
  var activities:=change.objectsOfType(SubstituteActivity);
  var pin:=change.objectsOfType(SubstitutePin);
  var cnode:= change.objectsOfType(SubstituteControlNode);
  var remactivities:=change.objectsOfType(RemoveActivity);
  var rempin:=change.objectsOfType(RemovePin);
  var remcnode:= change.objectsOfType(RemoveControlNode);
  var substituted:=false;
  activities->forEach(sa) {
    if sa.oldActivityName=str then substituted:=true endif;
  };
  remactivities->forEach(ra) {
    log(ra.activityName+"    "+ str);
    if ra.activityName=str then substituted:=true endif;
  };
  pin->forEach(sp) {
    if sp.oldPinName=str then substituted:=true endif;
  };
  rempin->forEach(rp)  {
    if rp.pinName=str then substituted:=true endif;
  };
  cnode->forEach(scn) {
    if scn.oldNodeName=str then substituted:=true endif;
  };
  remcnode->forEach(rcn) {
    if rcn.NodeName=str then substituted:=true endif;
  };
  return substituted;
}

helper deleteunusedpins() {
  var messages:=ad.objectsOfType(ObjectFlow); var pins:= ad.objectsOfType(Pin);
  pins->forEach(p) {
    var used:=false;
    messages->forEach(m) {
      if m.source=p or m.target=p then used:=true endif;
    };
    if used=false then ad.removeElement(p) endif;
  }
}

}
```

C3.2 GenerateMitigationNotation.qvto

```
modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';
modeltype MT uses 'http://www.paluno.de/hazardmitigation';
modeltype NOTATION uses 'http://www.eclipse.org/gmf/runtime/1.0.2/notation';
modeltype ECORE uses "http://www.eclipse.org/emf/2002/Ecore";

transformation grafic_for_mitigations(in ad:UML, inout notation:NOTATION, in change:MT);
property diagram:NOTATION::Diagram=null;
property rootnode:NOTATION::DecorationNode=null;
property wert1: Integer=20;
property wert2: Integer=400;
property counter: Integer=1;

main() {
  notation.objectsOfType(NOTATION::Diagram)->forEach(d){ diagram:=d};
  notation.objectsOfType(NOTATION::DecorationNode)->forEach(dn) {
    if dn.type="7004" then rootnode:=dn endif; };
  deleteunusedshapes();
  rootnode.children+=change.objectsOfType(InsertActivity)->map OpaquetoShape();
  // add in and output port to shapes
  var list:Set(Pin)=getPinsnotinNotation();
  list->forEach(p) {
    var activity:OpaqueAction=p.getOpaqueAction();
    notation.objectsOfType(NOTATION::Shape) ->forEach(s) {
      if s.type="3007" then
        if s.element.toString()=activity.toString() then s.children+= p->map PintoShape()
        endif
      endif;
    };
  };

  notation.objectsOfType(NOTATION::Shape) ->forEach(s) {
    if s.type="2001" then s.children+= change.objectsOfType(MT::InsertPin)
      ->map ParameterNodetoShape() endif;
    };

  rootnode.children+=change.objectsOfType(MT::InsertControlNode)->ControlNodetoShape();
  var subCN:=change.objectsOfType(MT::SubstituteControlNode);
  subCN->forEach(s) {
    rootnode.children+=s->map replaceControlNode()
  };

  diagram.edges+=change.objectsOfType(MT::InsertActivityEdge)->map addActivityEdge();
  var subAE:=change.objectsOfType(MT::SubstituteActivityEdge);
  subAE->forEach(s) {
    s.replaceActivityEdge()
  };
  deleteunusedshapes();
}

mapping InsertActivity::OpaquetoShape(): NOTATION::Shape {
  type:="3007";
  var decnode:notation::Node:= object DecorationNode{type:="5003"};
  children+=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
  element:=self.getOpaqueAction().oclAsType(ecore::EObject);
  var bound:notation::LayoutConstraint := object Bounds{x:=wert1; y:=wert2};
  if counter.toString()="4" then (wert1:=20 ) else wert1:=wert1+200 endif;
  if wert1.toString()="20" then wert2:=wert2+200 endif;
  counter:=counter+1;
  layoutConstraint:=bound;
}

mapping Pin::PintoShape():NOTATION::Shape {
  if self.incoming->isEmpty() then type:="3014"
  else if self.outgoing->isEmpty() then type:="3013" endif endif;
  var decnode:notation::Node:= object DecorationNode{type:="5009"};
  var bound:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=bound;
  children+=decnode;
  var decnode2:notation::Node:= object DecorationNode{type:="5085"};
  var bound2:notation::LayoutConstraint := object Location{};
   decnode2.layoutConstraint:=bound2;
  children+=decnode2;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound3:notation::LayoutConstraint := object Bounds{};
  layoutConstraint:= bound3;
}

mapping InsertPin::ParameterNodetoShape(): NOTATION::Shape {
  type:="3059";
  var decnode:notation::Node:= object DecorationNode{type:="5071"};
  decnode.element:= self.getPin().oclAsType(ecore::EObject);
  children:=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
   var bound:notation::LayoutConstraint := object Bounds{x:=-20; y:=wert1};
   layoutConstraint:=bound;
   element:= self.getPin().oclAsType(ecore::EObject);
   wert1:=wert1+40;
};

mapping InsertControlNode::ControlNodetoShape():NOTATION::Shape {
  if self.nodeType="Decision" then type:="3038"
  else if self.nodeType="Merge" then type:="3039"
  else if self.nodeType="Fork" then type:="3040"
  else if self.nodeType="Join" then type:="3041"
    endif endif endif endif;
  var decnode:notation::Node:= object DecorationNode{};
  var bound:notation::LayoutConstraint := object Location{};
   decnode.layoutConstraint:=bound;
   decnode.element:=getControlNode(self.NodeName).oclAsType(ecore::EObject);
   if type="3038" then decnode.type:="5043"
   else if type="3039" then decnode.type:="5099"
   else if type="3040" then decnode.type:="5100"
   else if type="3041" then decnode.type:="5042"
     endif endif endif endif;
  children+=decnode;
  if type="3038" or type="3041"  then {
    var decnode2:notation::Node:= object DecorationNode{};
    var bound2:notation::LayoutConstraint := object Location{};
    decnode2.layoutConstraint:=bound2;
    decnode2.element:=getControlNode(self.NodeName).oclAsType(ecore::EObject);
    if type="3038" then decnode2.type:="5098" else decnode2.type:="5101" endif;
    children+=decnode2;
  } endif;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=getControlNode(self.NodeName).oclAsType(ecore::EObject);
  var bound3:notation::LayoutConstraint := object Bounds{x:=wert1; y:=wert2};
  layoutConstraint:=bound3;
  if counter.toString()="4" then (wert1:=20 ) else wert1:=wert1+200 endif;
  if wert1.toString()="20" then wert2:=wert2+200 endif;
  counter:=counter+1;
}

mapping MT::SubstituteControlNode::replaceControlNode():Shape {
  if self.newNodeType="Decision" then type:="3038"
  else if self.newNodeType="Merge" then type:="3039"
  else if self.newNodeType="Fork" then type:="3040"
  else if self.newNodeType="Join" then type:="3041"
    endif endif endif endif;
  var decnode:notation::Node:= object DecorationNode{type:="5042"};
  var bound:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=bound;
  children+=decnode;
  var decnode2:notation::Node:= object DecorationNode{type:="5042"};
  var bound2:notation::LayoutConstraint := object Location{};
   decnode2.layoutConstraint:=bound2;
  children+=decnode2;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
   element:=getControlNode(self.newNodeName).oclAsType(ecore::EObject);
   var bound3:notation::LayoutConstraint := object Bounds{x:=wert1; y:=wert2};
   layoutConstraint:=bound3;
  if counter.toString()="4" then (wert1:=20 ) else wert1:=wert1+200 endif;
  if wert1.toString()="20" then wert2:=wert2+200 endif;
  counter:=counter+1;
}

helper MT::SubstituteActivityEdge::replaceActivityEdge() {
  // suche Nachricht mit neuen Parametern
  var objectflow:=self.getnewobjectFlow();
  log(objectflow.toString());
  notation.objectsOfType(NOTATION::Connector)->forEach(con) {
    if con.element.toString()=objectflow.toString() then {
      log(objectflow.source.toString());
      log(objectflow.target.toString());
      con.source:=getShape(objectflow.source);
      con.target:=getShape(objectflow.target);
    } endif;
  }
}

mapping MT::InsertActivityEdge::addActivityEdge():NOTATION::Connector {
  var nodes:= notation.objectsOfType(Shape);
  var sourceshape:NOTATION::Shape:=null;
  var targetshape:NOTATION::Shape:=null;
  var objectFlow:UML::ObjectFlow=self.getobjectFlow();
  nodes-> forEach(n) {
    if objectFlow.source.toString() = n.element.toString() then sourceshape:=n endif;
    if objectFlow.target.toString()= n.element.toString() then targetshape:=n endif;
  };
  element:=objectFlow.oclAsType(ecore::EObject);
  target:= targetshape;
  source:= sourceshape;
  type:="4003";
  var decnode:notation::Node:= object DecorationNode{type:="6001"};
  var layout:notation::LayoutConstraint:= object Location{ };
  decnode.layoutConstraint:= layout;
  decnode.element:=objectFlow.oclAsType(ecore::EObject);
  children+=decnode;
  var decnode1:notation::Node:= object DecorationNode{type:="6002"};
  var layout1:notation::LayoutConstraint:= object Location{};
  decnode1.layoutConstraint:= layout1;
  decnode1.element:=objectFlow.oclAsType(ecore::EObject);
  children+=decnode1;
  var decnode2:notation::Node:= object DecorationNode{type:="6005"};
  var layout2:notation::LayoutConstraint:= object Location{};
  decnode2.element:=objectFlow.oclAsType(ecore::EObject);
  decnode2.layoutConstraint:= layout2;
  children+=decnode2;
  var decnode3:notation::Node:= object DecorationNode{type:="6006"};
  decnode3.element:=objectFlow.oclAsType(ecore::EObject);
  var layout3:notation::LayoutConstraint:= object Location{};
  decnode3.layoutConstraint:= layout3;
  children+=decnode3;
  var decnode4:notation::Node:= object DecorationNode{type:="6007"};
  decnode4.element:=objectFlow.oclAsType(ecore::EObject);
  var layout4:notation::LayoutConstraint:= object Location{};
  decnode4.layoutConstraint:= layout4;
  children+=decnode4;
  var decnode5:notation::Node:= object DecorationNode{type:="6008"};
  decnode5.element:=objectFlow.oclAsType(ecore::EObject);
  var layout5:notation::LayoutConstraint:= object Location{};
  decnode5.layoutConstraint:= layout5;
  children+=decnode5;
  var decnode6:notation::Node:= object DecorationNode{type:="6010"};
  decnode6.element:=objectFlow.oclAsType(ecore::EObject);
  var layout6:notation::LayoutConstraint:= object Location{};
  decnode6.layoutConstraint:= layout6;
  children+=decnode6;
  var fontstyle:notation::Style:= object FontStyle{};
  styles+=fontstyle;
  var bends: Bendpoints:= object RelativeBendpoints{};
  bendpoints:=bends;
}

helper MT::InsertActivity::getOpaqueAction():UML::OpaqueAction {
  ad.objectsOfType(OpaqueAction)->forEach(oa) {
    if oa.name=self.activityName then {return oa; break;} endif;
  };
  return null;
}

helper MT::InsertPin::getPin():UML::ActivityParameterNode {
  ad.objectsOfType(ActivityParameterNode)->forEach(pn) {
    if pn.name=self.pinName then {return pn; break;} endif;
  };
  return null;
}

helper getControlNode(str:String):UML::ControlNode {
  ad.objectsOfType(ControlNode)->forEach(cn) {
    if cn.name=str then {return cn; break;} endif;
  };
  return null;
}

helper MT::InsertActivityEdge::getobjectFlow():UML::ObjectFlow {
  ad.objectsOfType(ObjectFlow)->forEach(of) {
    if ((of.name.toString()="null"
      and self.Message.toString()="null")
      or of.name=self.Message) then
        if ((of.guard.name.toString()="invalid"
          and self.Guard.toString()="null")
          or of.guard.name=self.Guard) then
            if  of.source.isPin()=true then {
              if of.source.getnameofopaqueaction()=self.SourceName then
                if  of.target.isPin()=true then {
                  if of.target.getnameofopaqueaction()=self.targetName then {
                    return of;
                    break;
                  } endif; }
                else if of.target.isPin()=false then {
                  if of.target.name=self.targetName then {
                    return of;
                    break;
                  } endif;}
                endif
              endif
            endif;
          }
          else if  of.source.isPin()=false then {
            if of.source.name=self.SourceName then
              if of.target.isPin()=true then {
                if of.target.getnameofopaqueaction()=self.targetName then { return of; break;}
                endif;}
              else if of.target.isPin()=false then {
                if of.target.name=self.targetName then {return of ;break;} endif;}
              endif endif endif;
          } endif endif endif endif;
  };
  return null;
}

helper MT::SubstituteActivityEdge::getnewobjectFlow():UML::ObjectFlow {
  ad.objectsOfType(ObjectFlow)->forEach(of) {
  if (of.name.toString()=self.newMessage.toString()) then
    if ((of.guard.toString()="null"
      and self.newGuard.toString()="")
      or of.guard.name=self.newGuard) then
        if  of.source.isPin()=true then {
          if of.source.getnameofopaqueaction()=self.newSourceName then
            if  of.target.isPin()=true then {
              if of.target.getnameofopaqueaction()=self.newTargetName then {return of; break;}
                endif;}
              else if of.target.isPin()=false then {
                if of.target.name=self.newTargetName then {return of; break;} endif;}
              endif endif endif;
          } else if  of.source.isPin()=false then {
          if of.source.name=self.newSourceName then
            if  of.target.isPin()=true then {
              if of.target.getnameofopaqueaction()=self.newTargetName then {return of; break;}
                endif;}
              else if of.target.isPin()=false then {
                if of.target.name=self.newTargetName then {return of; break;} endif;}
              endif endif endif;
          } endif endif endif endif;
  };
  return null;
}

helper  UML::ActivityNode::isPin():Boolean {
  var isPin:=false;
  ad.objectsOfType(OutputPin)->forEach(op) {
    if self.toString()=op.toString() then isPin:=true endif;
  };
  ad.objectsOfType(InputPin)->forEach(ip) {
    if self.toString()=ip.toString() then isPin:=true endif;
  };
  return isPin;
}

helper ActivityNode::getnameofopaqueaction():String {
  ad.objectsOfType(OpaqueAction)->forEach(oa) {
    oa.output->forEach(op) {
      if op.toString()=self.toString() then return oa.name endif;
    };
  };
  return "";
}

helper InputPin::getnameofopaqueaction():String {
  ad.objectsOfType(OpaqueAction)->forEach(oa) {
    oa.input->forEach(ip) {
      if ip.toString()=self.toString() then return oa.name endif;
    };
  };
  return "";
}

helper deleteunusedshapes() {
  //delete unused activities, pins and control nodes
  notation.objectsOfType(NOTATION::Shape)->forEach(s) {
    var exists:=false;
    if s.type="3007" then {
      ad.objectsOfType(OpaqueAction)->forEach(oa) {
        if s.element.toString()=oa.toString() then exists:=true endif;
      };
      if exists=false then notation.removeElement(s) endif;}
    endif;
    if s.type="3013" or s.type="3014" then {
      ad.objectsOfType(Pin)->forEach(p) {
        if s.element.toString()=p.toString() then exists:=true endif;
      };
      if exists=false then notation.removeElement(s) endif; }
    endif;
    if s.type="3059" then {
      ad.objectsOfType(ActivityParameterNode)->forEach(pn) {
        if s.element.toString()=pn.toString() then exists:=true endif;
      };
      if exists=false then notation.removeElement(s) endif; }
      endif;
      if s.type="3038" then {
        ad.objectsOfType(DecisionNode)->forEach(dn) {
          if s.element.toString()=dn.toString() then exists:=true endif;
        };
        if exists=false then notation.removeElement(s) endif;
      } endif;
      if s.type="3039" then {
        ad.objectsOfType(MergeNode)->forEach(mn) {
          if s.element.toString()=mn.toString() then exists:=true endif;
        };
      if exists=false then notation.removeElement(s) endif;
    } endif;
    if s.type="3040" then {
      ad.objectsOfType(ForkNode)->forEach(fn) {
        if s.element.toString()=fn.toString() then exists:=true endif;
      };
      if exists=false then notation.removeElement(s) endif;
    } endif;
    if s.type="3041" then {
      ad.objectsOfType(JoinNode)->forEach(jn) {
        if s.element.toString()=jn.toString() then exists:=true endif;
      };
      if exists=false then notation.removeElement(s) endif;
    } endif;
  };
    // delete unused edges
    notation.objectsOfType(NOTATION::Connector)->forEach(c) {
    var exists:=false;
    ad.objectsOfType(ObjectFlow)->forEach(of) {
      if c.element.toString()=of.toString() then exists:=true endif;
    };
    if exists=false then notation.removeElement(c) endif;
  }
}

helper  getPinsnotinNotation():Set(Pin) {
  var list:Set(Pin)=null;
  ad.objectsOfType(Pin)->forEach(p) {
    var exists:=false;
    notation.objectsOfType(NOTATION::Shape)->forEach(s) {
      if s.element.toString()=p.toString() then exists:=true endif
    };
    if exists=false then list+=p endif;
  };
  return list;
}

```

```
helper Pin::getOpaqueAction():OpaqueAction {
  ad.objectsOfType(OpaqueAction)->forEach(oa) {
    if oa.input.toString()->includes(self.toString())
      or oa.output.toString()->includes(self.toString()) then {return oa; break;} endif;
  };
  return null;
}

helper getShape(node: ActivityNode):NOTATION::Shape {
  notation.objectsOfType(NOTATION::Shape)->forEach(s) {
     if s.element.toString()=node.toString() then {return s; break;} endif;
  };
  return null;
}
```

C3.3 GenerateHRD.qvto

```
modeltype FHA uses 'http://www.paluno.de/fha';
modeltype MT uses 'http://www.paluno.de/hazardmitigation';
modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';

transformation createHRD(in inmodel:FHA, in profile:UML, in change:MT,in ad:UML, out hrd:UML);

property model : UML::Model = null;
property hazard:Stereotype=null;
property safetygoal:Stereotype=null;
property trigger_condition:Stereotype=null;
property andnodes:Stereotype=null;
property ornodes:Stereotype=null;
property hazardassociation:Stereotype=null;
property hazardrelation:Stereotype=null;
property mitigation:Stereotype=null;
property FunctionalHazard:Hazard=null;
property HazardRelation:MergeNode=null;

main() {
  model := object Model { name :='model' };
  var stereo:= profile.objects()[Stereotype] ->any(name='profile:'+ name);
  model.applyProfile(profile.objectsOfType(Profile)![name='profile']);
  getStereotypes();
  var StartActivity:= map createActivity();
  model.packagedElement += StartActivity;
  applyStereotypes();
  createEdgestoHazardRelation();
  hrd.objectsOfType(ControlFlow)->forEach(cf){
    StartActivity.edge+=cf; cf.applyStereotype(hazardassociation)
  };
  hrd.objectsOfType(InitialNode)->forEach(inode){inode.name:=""};
  StartActivity.edge+=hrd.objectsOfType(ConditionalNode)-> map createMitigationLink();
}

mapping  createActivity():Activity {
  name:="FHA";
  ownedNode+=inmodel.objectsOfType(Hazard)->map createHazard();
  ownedNode+= inmodel.objectsOfType(Safety_Goal)->map createSafetyGoal();
  ownedNode+= inmodel.objectsOfType(Trigger_Condition)->map createTriggerCondition();
  inmodel.objectsOfType(andnode)->forEach(an){
    if FunctionalHazard.TC._and->includes(an) then ownedNode+= an.createAndNode() endif;};
  inmodel.objectsOfType(ornode)->forEach(on){
    if FunctionalHazard.TC._or->includes(on)then ownedNode+= on.createOrNode() endif;};
  applyStereotypes();
  if FunctionalHazard!=null then {
    var relation:= createHazardRelation(); ownedNode+=relation; HazardRelation:=relation
  } endif;
  ownedBehavior+=ad.objectsOfType(Activity);
  ownedGroup+=change.objectsOfType(MitigationList)->map createMitigation();
  edge+=hrd.objectsOfType(ConditionalNode)-> map createMitigationLink();
}

mapping Hazard::createHazard():OpaqueAction when { self.isHazardReferencedInMitigation() }
  { name:=self.H_name; FunctionalHazard:=self; }

```

```
mapping Safety_Goal::createSafetyGoal():OpaqueAction when{FunctionalHazard.SG=self}
  { name:=self.SG_name }

mapping Trigger_Condition::createTriggerCondition():OpaqueAction when {
  FunctionalHazard.TC.cond->includes(self) }
  { name:=self.TC_name }

helper FHA::andnode::createAndNode(): InitialNode {
  var andn:InitialNode=object InitialNode {
    name:=self.and_id
  };
  var str:String=self.linked_by;
  var ending:Boolean=false;
  while (ending=false) {
    if str.indexOf(",")=0 then {andn->map createHazardAssociation(str); ending:=true}
    else andn->map createHazardAssociation(str.substringBefore(","))endif;
    str:= str.substringAfter(",");
  };
  return andn;
}

helper FHA::ornode::createOrNode(): InitialNode {
  var orn:InitialNode=object InitialNode {
    name:=self.or_id
  };
  var str:String=self.linked_by;
  var ending:Boolean=false;
  while (ending=false) {
    if str.indexOf(",")=0 then { orn->map createHazardAssociation(str); ending:=true }
    else orn->map createHazardAssociation(str.substringBefore(",")) endif;
    str:= str.substringAfter(",")
  };
  return orn;
}

mapping MitigationList::createMitigation():ConditionalNode when {FunctionalHazard!=null}
  { name:=self.ActivityDiagramName; }

helper createHazardRelation():MergeNode {
  var HazardRelation:MergeNode= object MergeNode{};
  -- erstelle Nachrichten hierhin von Hazard, Safety Goal
  -- und And/OrNode ohne ausgehenden Nachrichten
  -- falls keine AND/OrNodes vorhanden sind dann direkt von Trigger Conditions
  var initials:=hrd.objectsOfType(InitialNode);
  var counter:=0;
  initials->forEach(i) {
    if i.outgoing->isEmpty() then {HazardRelation->map createHazardAssociation(i)}endif;
    counter:=counter+1
  };
  var opaques:=hrd.objectsOfType(OpaqueAction);
  opaques->forEach(o) {
    if o.isStereotypeApplied(hazard) then {HazardRelation->map createHazardAssociation(o)}
    elif o.isStereotypeApplied(safetygoal) then {
      HazardRelation->map createHazardAssociation(o)}
    elif counter=0 then{ if o.isStereotypeApplied(trigger_condition) then {
      HazardRelation->map createHazardAssociation(o) }
    endif; } endif;
  };
  return HazardRelation;
}

mapping InitialNode::createHazardAssociation(str: String): ControlFlow {
  --durchlaufe alle Trigger Conditions, AndNodes und OrNodes
  --wo die ID mit der übergebenen übereinstimmt
  var trigger:=inmodel.objectsOfType(Trigger_Condition);
  trigger->forEach(tc) {
    if tc.TC_ID=str then {
      var opaques:=hrd.objectsOfType(OpaqueAction);
      opaques->forEach(oa) {
        if oa.name= tc.TC_name then source:=oa endif;
      }
    }
    endif;
  };
  var ands:=inmodel.objectsOfType(andnode);
  ands->forEach(a) {
    if a.and_id=str then {
      var initials:=hrd.objectsOfType(InitialNode);
      initials->forEach(ins) {
        if ins.name= a.and_id then source:=ins endif;
      }
    } endif;
  }; var ors:=inmodel.objectsOfType(ornode);
  ors->forEach(o) {
    if o.or_id=str then {
    var initials:=hrd.objectsOfType(InitialNode);
    initials->forEach(ins) {
    if ins.name= o.or_id then source:=ins endif;
  } } endif; };
  target:=self;
}

mapping MergeNode::createHazardAssociation(n:ActivityNode): ControlFlow
  { source:=n; target:=self; }

helper getStereotypes() {
  profile.objectsOfType(Stereotype)->forEach(s) {
    if s.name="Hazard" then{hazard:=s}
    elif s.name="Safety_Goal" then{safetygoal:=s}
    elif s.name="trigger_Condition" then{trigger_condition:=s}
    elif s.name="Context_Conjunction" then{andnodes:=s}
    elif s.name="Context_Disjunction" then{ornodes:=s}
    elif s.name="Hazard_Association" then{hazardassociation:=s}
    elif s.name="Mitigation" then{mitigation:=s}
    elif s.name="Hazard_Relation" then{hazardrelation:=s}endif;
  }
}

mapping ConditionalNode::createMitigationLink():ControlFlow
  { target:=self; source:=HazardRelation; }

helper applyStereotypes() {
  var trigger:=inmodel.objectsOfType(Trigger_Condition);
  trigger->forEach(tc) {
    var opaques:=hrd.objectsOfType(OpaqueAction);
    opaques->forEach(oa) {
      if oa.name= tc.TC_name then oa.applyStereotype(trigger_condition) endif;
    }
  };
  var haz:=inmodel.objectsOfType(Hazard);
  haz->forEach(h) {
    var opaques:=hrd.objectsOfType(OpaqueAction);
    opaques->forEach(oa) {
      if oa.name= h.H_name then oa.applyStereotype(hazard) endif;
    }
  }; var safgoal:=inmodel.objectsOfType(Safety_Goal);
  safgoal->forEach(sg) {
    var opaques:=hrd.objectsOfType(OpaqueAction);
    opaques->forEach(oa) {
      if oa.name= sg.SG_name then oa.applyStereotype(safetygoal) endif;
    }
  }; var andnod:=inmodel.objectsOfType(andnode);
  andnod->forEach(a) {
    var initials:=hrd.objectsOfType(InitialNode);
      initials->forEach(i) {
        if i.name= a.and_id then i.applyStereotype(andnodes) endif;
      }
    }; var ornod:=inmodel.objectsOfType(ornode);
  ornod->forEach(o) {
    var initials:=hrd.objectsOfType(InitialNode);
    initials->forEach(i) {
      if i.name= o.or_id then i.applyStereotype(ornodes) endif;
    }
  }; hrd.objectsOfType(MergeNode)->forEach(mn) {
    mn.applyStereotype(hazardrelation)
  }; var conditionals:=hrd.objectsOfType(ConditionalNode)->forEach(cn) {
    cn.applyStereotype(mitigation)
  };
}

helper createEdgestoHazardRelation() {
  var HazardRelation:=hrd.objectsOfType(MergeNode);
  HazardRelation->forEach(HR) {
    var initials:=hrd.objectsOfType(InitialNode);
    var counter:=0;
    initials->forEach(i){counter:=counter+1
  }; var opaques:=hrd.objectsOfType(OpaqueAction);
  opaques->forEach(o) {
    if o.isStereotypeApplied(hazard) then {HR->map createHazardAssociation(o)}
    elif o.isStereotypeApplied(safetygoal) then {HR->map createHazardAssociation(o)}
    elif counter=0 then {
      if o.isStereotypeApplied(trigger_condition) then {
        HR->map createHazardAssociation(o)
      } endif; } endif;
    };
  }
}
helper Hazard::isHazardReferencedInMitigation():Boolean {
  var exists:=false; change.objectsOfType(MitigationList)->forEach(ml) {
    if self.H_ID= ml.Hazard_ID then exists:=true endif;
  }; return exists;
}
```

C3.4 GenerateHRDNotation.qvto

```
modeltype NOTATION uses 'http://www.eclipse.org/gmf/runtime/1.0.2/notation';
modeltype ECORE uses "http://www.eclipse.org/emf/2002/Ecore";
modeltype STYLE uses 'http://www.eclipse.org/papyrus/infra/viewpoints/policy/style';
modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';
modeltype MT uses 'http://www.paluno.de/hazardmitigation';

transformation graphicalHRD(in hrd:UML,in change:MT, out notation:NOTATION, out di:STYLE);
property xwert: Integer=20;
property ywert: Integer=20;
property activityshapes:Set(Shape)=null;
property counter: Integer=0;

main() {
  hrd.objectsOfType(Activity)->map ActivitytoDiagram();
  hrd.objectsOfType(ControlFlow)->map CFlowtoEdge();
  hrd.objectsOfType(ObjectFlow)->map OFlowtoEdge();
  notation.objectsOfType(Diagram)->forEach(d) {
    d.edges+=notation.objectsOfType(Connector)
  }
  -- fuege Konnektoren Diagramm hinzu.
}

mapping  Activity::ActivitytoDiagram():NOTATION::Diagram when {
  self.ownedBehavior->notEmpty() } {
  type:= 'PapyrusUMLActivityDiagram';
  name:= "HazardRelationDiagram";
  measurementUnit:= MeasurementUnit::Pixel;
  var shape:notation::Shape:= object Shape{type:="2001"};
  var layout:notation::LayoutConstraint:=object Bounds{};
  shape.element:= self.oclAsType(ecore::EObject);
  shape.layoutConstraint:=layout;
  children+=shape;
  var decnode1:notation::Node:= object DecorationNode{type:="5001"};
  shape.children+=decnode1;
  var decnode2:notation::Node:= object DecorationNode{type:="7001"};
  var style1:notation::Style:= object SortingStyle{};
  var style2:notation::Style:= object FilteringStyle{};
  var layout1:notation::LayoutConstraint:= object Bounds{};
  decnode2.styles+=style1;decnode2.styles+=style2; decnode2.layoutConstraint:=layout1;
  shape.children+=decnode2;
  var decnode3:notation::Node:= object DecorationNode{type:="7002"};
  var style3:notation::Style:= object SortingStyle{};
  var style4:notation::Style:= object FilteringStyle{};
  var layout2:notation::LayoutConstraint:= object Bounds{};
  decnode3.styles+=style3;decnode3.styles+=style4; decnode3.layoutConstraint:=layout2;
  shape.children+=decnode3;
  var decnode4:notation::Node:= object DecorationNode{type:="7003"};
  var style5:notation::Style:= object SortingStyle{};
  var style6:notation::Style:= object FilteringStyle{};
  var layout3:notation::LayoutConstraint:= object Bounds{};
  decnode4.styles+=style5;decnode4.styles+=style6; decnode4.layoutConstraint:=layout3;
  shape.children+=decnode4;
  var decnode:notation::Node:= object DecorationNode{type:="7004"};
  var bound:notation::LayoutConstraint := object Bounds{};
  self.ownedElement->forEach(oe) {
    if oe.isOpaqueAction() then{decnode.children+= oe.getOpaqueAction()->map OpaquetoShape() }
    endif;
  };
  self.ownedElement->forEach(oe) {
    if oe.isInitialNode() then {
      decnode.children+=oe.getInitialNode()->map InitialNodetoShape() } endif;
  };
  self.ownedElement->forEach(oe) {
    if oe.isMergeNode() then {
      decnode.children+=oe.getMergeNode()->map MergeNodetoShape() }endif;
  };
  self.ownedElement->forEach(oe) {
    if oe.isActivity() then{decnode.children+=oe.getActivity()->map ActivitytoShape();} endif;
  };
  self.ownedElement->forEach(oe) {
    if oe.isConditionalNode() then {
      decnode.children+=oe.getConditionalNode()->map ConditionalNodetoShape()}endif;
  };
  decnode.layoutConstraint:=bound;
  shape.children+=decnode;
  var stringstyle:NOTATION::Style:=object StringValueStyle {
    name:="diagram_compability_version"; stringValue:="1.0.0" };
  var diastyle:NOTATION::Style:=object DiagramStyle{};
  var papyrusstyle:NOTATION::Style:=object STYLE::PapyrusViewStyle {
    hrd.objectsOfType(Model)->forEach(m){ owner:=m.oclAsType(ecore::EObject)}};
  styles+=stringstyle;
  styles+= diastyle;
  styles+=papyrusstyle;
  element:=self.oclAsType(ecore::EObject);
}

mapping OpaqueAction::OpaquetoShape(): NOTATION::Shape {
  type:="3007";
  --Anzeigen von Stereotypes
  var detail1:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry{
    key:="StereotypeWithQualifiedNameList"; value:=""};
  var detail2:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry{
    key:="StereotypeList"};
  var Stereotypes:= self.getAppliedStereotypes();
  Stereotypes->forEach(st) {
    if st.name="Hazard" then detail2.value:="profile::Hazard" endif;
    if st.name="Safety_Goal" then detail2.value:="profile::Safety_Goal" endif;
    if st.name="trigger_Condition" then detail2.value:="profile::trigger_Condition" endif;
  };
  var detail3:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="Stereotype_Presentation_Kind"; value:="HorizontalStereo"};
  var detail4:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="PropStereoDisplay"; value:=""};
  var detail5:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypePropertyLocation"; value:="Compartment"};
  var annotate:ecore::EAnnotation:=object EAnnotation {
    source:="Stereotype_Annotation";
    details+=detail1; details+=detail2; details+=detail3; details+=detail4; details+=detail5
  };
  eAnnotations:=annotate;
  --Ende Anzeigen Stereotype
  var decnode:notation::Node:= object DecorationNode{type:="5003"};
  children+=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound:notation::LayoutConstraint := object Bounds{x:=xwert; y:=ywert};
  if counter=4 then {xwert:=20; counter:=0} else xwert:=xwert+200 endif;
  if xwert=20 then ywert:=ywert+50 endif;
  counter:=counter+1;
  layoutConstraint:=bound;
}

mapping OpaqueAction::OpaqueActivitytoShape(referenced:Boolean): NOTATION::Shape {
  type:="3007";
  var decnode:notation::Node:= object DecorationNode{type:="5003"};
  children+=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound:notation::LayoutConstraint := object Bounds {
    if referenced=true then x:=20 else x:=450 endif;
    y:=20; -- of set new item by a few pixels to avoid exact overlap
  };
  layoutConstraint:=bound;
  if self.input->notEmpty() then {
    self.input->forEach(i){children+=i->map InputtoShape()}} endif;
  if self.output->notEmpty() then {
    self.output->forEach(o){children+=o->map OutputtoShape()}} endif;
}

mapping ActivityParameterNode::ParameterNodetoShape(referenced:Boolean): NOTATION::Shape {
  type:="3059";
  var decnode:notation::Node:= object DecorationNode{type:="5071"};
  children:=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
   styles:=style;
  var bound:notation::LayoutConstraint := object Bounds {
    if referenced=true then x:=20 else x:=450 endif;};
  layoutConstraint:=bound;
   element:= self.oclAsType(ecore::EObject);};

mapping InputPin::InputtoShape():NOTATION::Shape {
  type:="3013";
  var decnode:notation::Node:= object DecorationNode{type:="5009"};
  var bound:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=bound;
  children+=decnode;
  var decnode2:notation::Node:= object DecorationNode{type:="5085"};
..var bound2:notation::LayoutConstraint := object Location{};
   decnode2.layoutConstraint:=bound2;
  children+=decnode2;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound3:notation::LayoutConstraint := object Bounds{};
  layoutConstraint:= bound3;
}

mapping OutputPin::OutputtoShape():NOTATION::Shape {
  type:="3014";
  var decnode:notation::Node:= object DecorationNode{type:="5009"};
  var bound:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=bound;
  children+=decnode;
  var decnode2:notation::Node:= object DecorationNode{type:="5085"};
  var bound2:notation::LayoutConstraint := object Location{};
  decnode2.layoutConstraint:=bound2;
  children+=decnode2;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound3:notation::LayoutConstraint := object Bounds{};
  layoutConstraint:= bound3;
}

mapping InitialNode::InitialNodetoShape(): NOTATION::Shape {
  type:="3004";
--Anzeigen von Stereotypes
  var detail1:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
  key:="StereotypeWithQualifiedNameList"; value:=""};
  var detail2:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypeList"};
  var Stereotypes:= self.getAppliedStereotypes();
  Stereotypes->forEach(st) {
    if st.name="Context_Conjunction" then detail2.value:="profile::Context_Conjunction" endif;
    if st.name="Context_Disjunction" then detail2.value:="profile::Context_Disjunction" endif;
  };
  var detail3:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="Stereotype_Presentation_Kind"; value:="HorizontalStereo"};
  var detail4:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="PropStereoDisplay"; value:=""};
  var detail5:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypePropertyLocation"; value:="Compartment"};
  var annotate:ecore::EAnnotation:=object EAnnotation {
    source:="Stereotype_Annotation";
    details+=detail1; details+=detail2; details+=detail3; details+=detail4 details+=detail5
  };
  eAnnotations:=annotate;
--Ende Anzeigen Stereotype
  var decnode:notation::Node:= object DecorationNode { type:="5080" };
  var locate:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=locate;
  children+=decnode;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound:notation::LayoutConstraint := object Bounds{x:=xwert; y:=ywert};
  if counter=4 then {xwert:=20; counter:=0} else xwert:=xwert+200 endif;
  if xwert=20 then ywert:=ywert+50 endif;
  counter:=counter+1;
  layoutConstraint:=bound;
}

mapping MergeNode::MergeNodetoShape(): NOTATION::Shape {
  type:="3039";
--Anzeigen von Stereotypes
  var detail1:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypeWithQualifiedNameList"; value:=""};
  var detail2:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypeList"};
  var Stereotypes:= self.getAppliedStereotypes();
  Stereotypes->forEach(st) {
    if st.name="Hazard_Relation" then detail2.value:="profile::Hazard_Relation" endif;
  };
  var detail3:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="Stereotype_Presentation_Kind"; value:="HorizontalStereo"};
  var detail4:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="PropStereoDisplay"; value:=""};
  var detail5:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypePropertyLocation"; value:="Compartment"};
  var annotate:ecore::EAnnotation:=object EAnnotation {
    source:="Stereotype_Annotation";
    details+=detail1; details+=detail2; details+=detail3; details+=detail4;  details+=detail5
  };
  eAnnotations:=annotate;
--Ende Anzeigen Stereotype
  var decnode:notation::Node:= object DecorationNode{type:="5099"};
  var locate:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=locate;
  children+=decnode;
   var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
   var bound:notation::LayoutConstraint := object Bounds{x:=xwert; y:=ywert};
  if counter=4 then {xwert:=20; counter:=0} else xwert:=xwert+200 endif;
  if xwert=20 then ywert:=ywert+50 endif;
  counter:=counter+1;
  layoutConstraint:=bound;
}

mapping ConditionalNode::ConditionalNodetoShape(): NOTATION::Shape {
  type:="3069";
  transparency:=100;
--Anzeigen von Stereotypes
  var detail1:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypeWithQualifiedNameList"; value:=""};
  var detail2:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypeList"};
  var Stereotypes:= self.getAppliedStereotypes();
  Stereotypes->forEach(st) {
    if st.name="Mitigation" then detail2.value:="profile::Mitigation" endif;
  };
  var detail3:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="Stereotype_Presentation_Kind"; value:="HorizontalStereo"};
  var detail4:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="PropStereoDisplay"; value:=""};
  var detail5:ecore::EStringToStringMapEntry:= object EStringToStringMapEntry {
    key:="StereotypePropertyLocation"; value:="Compartment"};
  var annotate:ecore::EAnnotation:=object EAnnotation {
    source:="Stereotype_Annotation";
    details+=detail1; details+=detail2; details+=detail3; details+=detail4; details+=detail5
  };
  eAnnotations:=annotate;
--Ende Anzeigen Stereotype
  var decnode:notation::Node:= object DecorationNode{type:="5119"};
  var decnode2:notation::Node:= object DecorationNode{type:="7008"};
  var bounds:notation::LayoutConstraint := object Bounds{};
  decnode2.layoutConstraint:=bounds;
  children+=decnode;
  children+=decnode2;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
--Position muss angepasst werden.
  var count:=0;
  var bound:notation::LayoutConstraint := object Bounds {
    x:=20;width:=200;
    unusedactivityShapes()->forEach(as) {
      if count=0 then {
        var boundsshape:=notation.objectsOfType(Bounds)->forEach(b) {
          if as.layoutConstraint=b then {
            y:=b.y-30; height:=b.height+80;
            activityshapes+=as;
          } else{} endif;
        }
      } else {} endif;
      count:=1;
    }
  };
  layoutConstraint:=bound;
}

mapping Activity::ActivitytoShape():NOTATION::Shape {
  type:="3083";
  var decnode:notation::Node:= object DecorationNode{type:="5142"};
  var decnode1:notation::Node:= object DecorationNode{type:="5143"};
  children+=decnode; children+=decnode1;
  var decnode2:notation::Node:= object DecorationNode{type:="7014"};
  var style:notation::Style:= object SortingStyle{};
  var style1:notation::Style:= object FilteringStyle{};
  var layout:notation::LayoutConstraint:= object Bounds{};
  decnode2.styles+=style;decnode2.styles+=style1; decnode2.layoutConstraint:=layout;
  children+=decnode2;
  var decnode3:notation::Node:= object DecorationNode{type:="7015"};
  var style2:notation::Style:= object SortingStyle{};
  var style3:notation::Style:= object FilteringStyle{};
  var layout1:notation::LayoutConstraint:= object Bounds{};
  decnode3.styles+=style2;decnode3.styles+=style3; decnode3.layoutConstraint:=layout1;
  children+=decnode3;
  var decnode4:notation::Node:= object DecorationNode{type:="7016"};
  var style4:notation::Style:= object SortingStyle{};
  var style5:notation::Style:= object FilteringStyle{};
  var layout2:notation::LayoutConstraint:= object Bounds{};
  decnode4.styles+=style4;decnode4.styles+=style5; decnode4.layoutConstraint:=layout2;
  children+=decnode4;
  element:=self.oclAsType(ecore::EObject);
  var ywert2:=0;
  var layout4:notation::LayoutConstraint:= object Bounds {
    x:=20;y:=ywert+200; height:=200;ywert2:=y+height; width:=600} ;
  layoutConstraint:=layout4;var decnode5:notation::Node:= object DecorationNode{type:="7013"};
  xwert:=20;ywert:=ywert+200;
  var layout3:notation::LayoutConstraint:= object Bounds{};
  decnode5.layoutConstraint:=layout3;
-- erstelle Elemente, die in der Activity sind
```

```
  self.ownedElement->forEach(on) {
    if on.isOpaqueAction() then {
      decnode5.children+=on.getOpaqueAction()->
        map OpaqueActivitytoShape(on.isReferencedinMitigationList())} endif;
  };
  self.ownedElement->forEach(on) {
    if on.isControlNode() then {
      decnode5.children+=on.getControlNode()->
        map ControlNodetoShape(on.isReferencedinMitigationList())} endif;
  };
  xwert:=20;
  self.ownedElement->forEach(on) {
    if on.isParameterNode() then {
      children+=on.getParameterNode()->
        map ParameterNodetoShape(on.isReferencedinMitigationList())} endif;
  };
  children+=decnode5;
  ywert:=ywert2+50; xwert:=20;
  counter:=0;
}

mapping ControlNode::ControlNodetoShape(referenced:Boolean):NOTATION::Shape {
  var cn:=self.toString();
  if cn.substringBefore("DecisionNode")!=null then type:="3038"
  else if cn.substringBefore("MergeNode")!=null then type:="3039"
  else if cn.substringBefore("ForkNode")!=null then type:="3040"
  else if cn.substringBefore("JoinNode")!=null then type:="3041"
  endif endif endif endif;
  var decnode:notation::Node:= object DecorationNode{};
  var bound:notation::LayoutConstraint := object Location{};
  decnode.layoutConstraint:=bound;
  if type="3038" then decnode.type:="5043"
  else if type="3039" then decnode.type:="5099"
  else if type="3040" then decnode.type:="5100"
  else if type="3041" then decnode.type:="5042"
  endif endif endif endif;
  children+=decnode;
  if type="3038" or type="3041" then {
    var decnode2:notation::Node:= object DecorationNode{};
    var bound2:notation::LayoutConstraint := object Location{};
    decnode2.layoutConstraint:=bound2;
    if type="3038" then decnode2.type:="5098" else decnode2.type:="5101" endif;
    children+=decnode2;
  } endif;
  var style:notation::Style := object HintedDiagramLinkStyle{};
  styles:=style;
  element:=self.oclAsType(ecore::EObject);
  var bound3:notation::LayoutConstraint := object Bounds {
    if referenced=true then x:=20 else x:=450 endif;
    y:=20;
   };
  layoutConstraint:=bound3;
}

mapping ControlFlow::CFlowtoEdge(): NOTATION::Connector {
  var nodes:= notation.objectsOfType(Shape);
  var sourceshape:NOTATION::Shape:=null;
  var targetshape:NOTATION::Shape:=null;
  var selfmessage:Boolean:=false;
  nodes-> forEach(n) {
    if self.source.toString() = n.element.toString() then sourceshape:=n endif;
    if self.target.toString()= n.element.toString() then targetshape:=n endif;
  };
  element:=self.oclAsType(ecore::EObject);
  source:= sourceshape;
  target:= targetshape;
  type:="4004";
  var decnode:notation::Node:= object DecorationNode{type:="6003"};
  var layout:notation::LayoutConstraint:= object Location{ y:=20};
  decnode.layoutConstraint:= layout;
  children+=decnode;
  var decnode1:notation::Node:= object DecorationNode{type:="6004"};
  var layout1:notation::LayoutConstraint:= object Location{ y:=20};
  decnode1.layoutConstraint:= layout1;
  children+=decnode1;
  var decnode2:notation::Node:= object DecorationNode{type:="6009"};
  var layout2:notation::LayoutConstraint:= object Location{ y:=20};
  decnode2.layoutConstraint:= layout2;
  children+=decnode2;
  var decnode3:notation::Node:= object DecorationNode{type:="6011"};
  var layout3:notation::LayoutConstraint:= object Location{ y:=-20};
  decnode3.layoutConstraint:= layout3;
  children+=decnode3;
  var fontstyle:notation::Style:= object FontStyle{};
  styles+=fontstyle;
  var bends: Bendpoints:= object RelativeBendpoints{};
  bendpoints:=bends;
  var sanchor:notation::Anchor:= object IdentityAnchor{};
  var tanchor:notation::Anchor:= object IdentityAnchor{};
  sourceAnchor:=sanchor;
  targetAnchor:=tanchor
}

mapping ObjectFlow::OFlowtoEdge(): NOTATION::Connector {
  var nodes:= notation.objectsOfType(Shape);
  var sourceshape:NOTATION::Shape:=null;
  var targetshape:NOTATION::Shape:=null;
  var selfmessage:Boolean:=false;
  nodes-> forEach(n) {
    if self.source.toString() = n.element.toString() then sourceshape:=n endif;
    if self.target.toString()= n.element.toString() then targetshape:=n endif;
  };
  element:=self.oclAsType(ecore::EObject);
  source:= sourceshape;
  target:= targetshape;
  type:="4003";
  var decnode:notation::Node:= object DecorationNode{type:="6001"};
  var layout:notation::LayoutConstraint:= object Location{ y:=20};
  decnode.layoutConstraint:= layout;
  children+=decnode;
  var decnode1:notation::Node:= object DecorationNode{type:="6002"};
  var layout1:notation::LayoutConstraint:= object Location{ y:=20};
  decnode1.layoutConstraint:= layout1;
  children+=decnode1;
  var decnode2:notation::Node:= object DecorationNode{type:="6005"};
  var layout2:notation::LayoutConstraint:= object Location{ y:=20};
  decnode2.layoutConstraint:= layout2;
  children+=decnode2;
  var decnode3:notation::Node:= object DecorationNode{type:="6006"};
  var layout3:notation::LayoutConstraint:= object Location{ y:=-20};
  decnode3.layoutConstraint:= layout3;
  children+=decnode3;
  var decnode4:notation::Node:= object DecorationNode{type:="6007"};
  var layout4:notation::LayoutConstraint:= object Location{ y:=-20};
  decnode4.layoutConstraint:= layout4;
  children+=decnode4;
  var decnode5:notation::Node:= object DecorationNode{type:="6008"};
  var layout5:notation::LayoutConstraint:= object Location{ y:=-20};
  decnode5.layoutConstraint:= layout5;
  children+=decnode5;
  var decnode6:notation::Node:= object DecorationNode{type:="6010"};
  var layout6:notation::LayoutConstraint:= object Location{ y:=-20};
  decnode6.layoutConstraint:= layout6;
  children+=decnode6;
  var fontstyle:notation::Style:= object FontStyle{};
  styles+=fontstyle;
  var bends: Bendpoints:= object RelativeBendpoints{};
  bendpoints:=bends;
  var sanchor:notation::Anchor:= object IdentityAnchor{};
  var tanchor:notation::Anchor:= object IdentityAnchor{};
  sourceAnchor:=sanchor; targetAnchor:=tanchor
}

helper Element::isOpaqueAction(): Boolean {
  var correct:=false;
  hrd.objectsOfType(OpaqueAction)->forEach(oa) { if self=oa then correct:=true endif };
  return correct;
}

helper Element::isControlNode(): Boolean {
  var correct:=false;
  hrd.objectsOfType(ControlNode)->forEach(cn) { if self=cn then correct:=true endif };
  return correct;
}

helper Element::isParameterNode(): Boolean {
  var correct:=false;
  hrd.objectsOfType(ActivityParameterNode)->forEach(pn) {if self=pn then correct:=true endif};
  return correct;
}

helper Element::isConditionalNode(): Boolean {
  var correct:=false;
  hrd.objectsOfType(ConditionalNode)->forEach(cn) {if self=cn then correct:=true endif};
  return correct;
}
helper Element::isInitialNode(): Boolean {
  var correct:=false;
  hrd.objectsOfType(InitialNode)->forEach(inn) {if self=inn then correct:=true endif};
  return correct;
}

helper Element::isMergeNode(): Boolean {
  var correct:=false;
  hrd.objectsOfType(MergeNode)->forEach(mn) {if self=mn then correct:=true endif};
  return correct;
}

helper Element::isActivity(): Boolean {
  var correct:=false;
  hrd.objectsOfType(Activity)->forEach(a) {if self=a then correct:=true endif};
  return correct;
}

helper Element::getOpaqueAction(): OpaqueAction {
  var opaque:OpaqueAction=null;
  hrd.objectsOfType(OpaqueAction)->forEach(oa) {if self=oa then opaque:=oa endif};
  return opaque;
}

helper Element::getConditionalNode(): ConditionalNode {
  var conditional:ConditionalNode=null;
  hrd.objectsOfType(ConditionalNode)->forEach(cn) {if self=cn then conditional:=cn endif};
  return conditional;
}

helper Element::getInitialNode(): InitialNode {
  var initial:InitialNode=null;
  hrd.objectsOfType(InitialNode)->forEach(inn) {if self=inn then initial:=inn endif};
  return initial;
}

helper Element::getMergeNode(): MergeNode {
  var merge:MergeNode=null;
  hrd.objectsOfType(MergeNode)->forEach(mn) {if self=mn then merge:=mn endif};
  return merge;
}

helper Element::getActivity(): Activity {
  var activity:Activity=null;
  hrd.objectsOfType(Activity)->forEach(a) { if self=a then activity:=a endif };
  return activity;
}

helper Element::getControlNode(): ControlNode {
  var correct:ControlNode=null;
  hrd.objectsOfType(ControlNode)->forEach(cn) {if self=cn then correct:=cn endif};
  return correct;
}

helper Element::getParameterNode(): ActivityParameterNode {
  var correct:ActivityParameterNode=null;
  hrd.objectsOfType(ActivityParameterNode)->forEach(pn) {if self=pn then correct:=pn endif};
  return correct;
}

helper Element::isReferencedinMitigationList():Boolean {
  var referenced:=false;
  change.objectsOfType(InsertActivity)->forEach(iA) {
    if self.isOpaqueAction()=true then
      if self.getOpaqueAction().name=iA.activityName then referenced:=true endif endif;
  };
  change.objectsOfType(SubstituteActivity)->forEach(sA) {
    if self.isOpaqueAction()=true then
      if self.getOpaqueAction().name=sA.newActivityName then referenced:=true endif endif;
  };
  change.objectsOfType(InsertPin)->forEach(iP) {
    if self.isParameterNode()=true then
      if self.getParameterNode().name=iP.pinName then referenced:=true endif endif;
  };
  change.objectsOfType(SubstitutePin)->forEach(sP) {
    if self.isParameterNode()=true then
      if self.getParameterNode().name=sP.newPinName then referenced:=true endif endif;
    };
  change.objectsOfType(InsertControlNode)->forEach(iCN) {
    if self.isParameterNode()=true then
      if self.getParameterNode().name=iCN.NodeName then referenced:=true endif endif;
    };
  change.objectsOfType(SubstituteControlNode)->forEach(sCN) {
    if self.isControlNode()=true then
      if self.getControlNode().name=sCN.newNodeName then referenced:=true endif endif;
  };
  change.objectsOfType(InsertActivityEdge)->forEach(iAE) {
    if self.isControlNode()=true then
      if self.getControlNode().name=iAE.SourceName
        or self.getControlNode().name=iAE.targetName then referenced:=true endif endif;
    if self.isParameterNode()=true then
    if self.getParameterNode().name=iAE.SourceName
        or self.getParameterNode().name=iAE.targetName then referenced:=true endif endif;
    if self.isOpaqueAction()=true then
      if self.getOpaqueAction().name=iAE.SourceName
        or self.getOpaqueAction().name=iAE.targetName then referenced:=true endif endif;
  };
  change.objectsOfType(SubstituteActivityEdge)->forEach(sAE) {
    if self.isControlNode()=true then
      if self.getControlNode().name=sAE.newSourceName
        or self.getControlNode().name=sAE.newTargetName then referenced:=true endif endif;
    if self.isParameterNode()=true then
      if self.getParameterNode().name=sAE.newSourceName
        or self.getParameterNode().name=sAE.newTargetName then referenced:=true endif endif;
    if self.isOpaqueAction()=true then
      if self.getOpaqueAction().name=sAE.newSourceName
        or self.getOpaqueAction().name=sAE.newTargetName then referenced:=true endif endif;
  };
  return referenced;
}

helper unusedactivityShapes():Set(Shape) {
  var unused:Set(Shape)=null;
  notation.objectsOfType(Shape)->forEach(s) {
    if s.type="3083" then {
      if activityshapes->includes(s) =false then {unused+=s}endif;} endif;
  };
  return unused;
}
```

C3.5 mergePMs.qvto

```
modeltype MT uses 'http://www.paluno.de/hazardmitigation';
transformation mergeMTs(in c1:MT, in c2:MT, out c3:MT);

main() {
  c1.objectsOfType(MitigationList)-> map addMitigations();
  c2.objectsOfType(MitigationList)-> map addMitigations();
}

mapping MitigationList::addMitigations():MitigationList {
  umlModelFile:=self.umlModelFile;
  ActivityDiagramName:=self.ActivityDiagramName;
  Mitigations+=self.Mitigations;
}
```

C3.6 mergeADs.qvto

```
modeltype UML uses 'http://www.eclipse.org/uml2/5.0.0/UML';

transformation MergeADs(in ad1:UML, in ad2:UML, out ad3:UML);

property model : UML::Model = null;

main() {
  model := object Model { name :='model' };
  model.packagedElement += ad1.objectsOfType(Activity);
  model.packagedElement += ad2.objectsOfType(Activity);
}
```