1 CORAS tool

Within the current document, CORAS tool refers to three standalone tools: the core CORAS tool, and two support tools (CAPEC2CORAS, and CorasAnalyzer) which provide additional functionality to the core tool. The core CORAS tool can be downloaded at http://coras.sourceforge.net. The support tools are available upon request.

General information	
Name	CORAS tool
Provider	SINTEF
Topic addressed	Model-based risk assessment
Description	The tool is based on Eclipse and the GMF/EMF framework, but it is distributed as a stand-alone tool
License	Eclipse Public License v1.0 (http://www.eclipse.org/legal/epl-v10.html)
Website	http://coras.sourceforge.net
Technical information	
Download site	http://coras.sourceforge.net/downloads.html
OS	Tested on Windows; a non-tested distribution is also available for Linux
Technology environment	The tool needs Java
Other dependencies	None
Additional information	
Known issues/risks	None
Additional useful information	None

1.1 Core CORAS tool

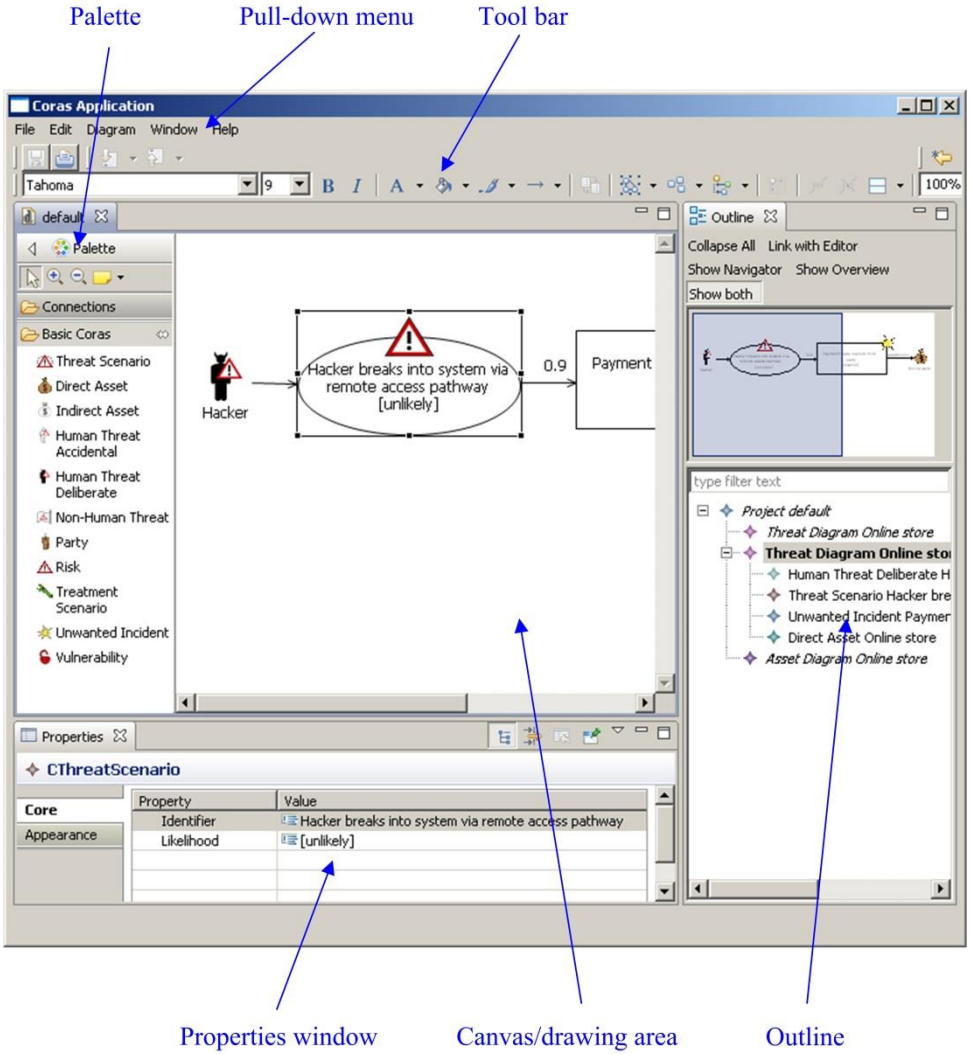

Figure 1 – Screenshot of the CORAS tool

The CORAS tool is an open source diagram editor that supports the CORAS risk analysis language. The CORAS language is a graphical language whose constructs correspond to notions that are relevant during a risk analysis, e.g. threats, vulnerabilities, risks, unwanted incidents, threat scenarios and assets. The CORAS tool is intended to be used intensively during workshops where information is gathered through structured brainstorming. The tool is also intended to be used to document a risk analysis and to present the risk analysis results.

The CORAS tool is designed to support on-the-fly modeling using all five kinds of basic CORAS diagrams, thus facilitating the entire CORAS risk analysis process. A screenshot of the CORAS diagram editor is given in Figure 1. As indicated in the figure, the editor has six main parts:

- A pull-down menu that offers standard functions such as open, save, copy, cut, paste, undo and print.
- A tool-bar that offers easy access to the standard functions of the pull-down menu.
- A palette that contains the model elements and relations for drawing the diagrams.
- A drawing area or canvas for drawing the diagrams.

- A properties window that lists the properties of a selected element, and that can be used to edit the values of the properties.
- An outline that presents a project and its diagrams as a tree.

Except for the pull-down menu and the tool bar, all parts of the tool can be closed or hidden.

In the tool, a project is a collection of diagrams, and each diagram must belong to a project. A project must therefore be created before any diagrams are created.

The outline contains a tree representation of the project. The diagrams of the project are listed at the first level, and under each diagram all the diagram elements are listed. When a new element is created in the drawing area, it is automatically added to the tree under the correct diagram.

The drawing area is the part of the tool where the diagrams are made by inserting, editing, annotating and deleting elements. This is also where likelihoods and consequences are inserted to diagrams as part of the risk estimation, and it is also where risk levels are inserted as part of the risk evaluation.

1.1.1 Installation Guidelines

CORAS tool version 1.4:

- Download the zip-file containing the current version of the tool.
- Extract the zip-file to any folder of your choosing.
- Double click the file "Coras.exe" located in the Coras folder of the distribution.

Capec2Coras

- Download the zip-file containing the current version of the tool.
- Extract the zip-file to any folder of your choosing.
- Double click the file "Capec2Coras.exe" located in the Capec2Coras folder of the distribution.

CorasAnalyzer

- Download the zip-file containing the current version of the tool.
- Extract the zip-file to any folder of your choosing.
- Double click the file "RUN_CORAS_ANALYZER.BAT" located in the CorasAnalyzer folder of the distribution.

1.1.2 User Guide: CORAS

Creating a new project

Before the CORAS tool can be properly used, a new project must be created. To create a new project:

- Select File -> New -> Coras Project in the File menu located near the top left corner of the window.
- Enter the desired file name and folder of the new project.
- Check “High Level CORAS”, “Dependent CORAS”, or “Legal CORAS” if you want to use these extensions of the basic CORAS language.
- Press “Finish” when done.

Creating a new diagram

The first time a new project is created, a new threat diagram called unnamed is automatically created within the project as shown in the outline view of the right hand side of the Figure 1. To create an additional diagram:

- Right-click the project entry in the tree outline located on the right hand side of the CORAS tool window.
- In the pull-down menu, select “new X diagram” (where X is the type of diagram that can be created e.g. threat, asset).
- The new diagram should appear in the tree outline. In order to open the new diagram, double click the diagram in the tree outline.

The difference between the diagrams is the kind of risk model elements that can be created within them. Note that the diagram type "Treatment" can contain (almost) all risk model elements; all other diagrams are subsets of this diagram (except the asset diagram).

Editing diagrams

Once a project and an appropriate diagram have been created, risk model elements can be added to the diagram by selection the appropriate element in the palette on the left hand side of the window, and left-clicking the diagram canvas. Relations can be created by selecting them from the pallet, or by holding the mouse over a source or target element until two small boxes appear on the border of the element. Holding down the left-mouse button on one of these small boxes will allow you to create a new relation.

1.2 Capec2Coras

Capec2Coras is a standalone tool which can be used to convert CAPEC attack patterns (represented in an XML-file) to CORAS risk diagrams, thus speeding up the process of creating the risk model.

1.2.1 Installation

- Download the zip-file containing the current version of the tool.
- Extract the zip-file to any folder of your choosing.
- Double click the file "RUN_CAPEC2CORAS.BAT" located in the Capec2Coras folder of the distribution.

1.2.2 User guide: Capec2Coras

Capec2Coras is program that enables the user to transform an XML file containing CAPEC attack patterns into a CORAS file which can be read by the CORAS tool. The tool allow the CAPEC-file to be viewed and edited before being transformed into CORAS. Parameters which are needed by the transformation can also be configured.

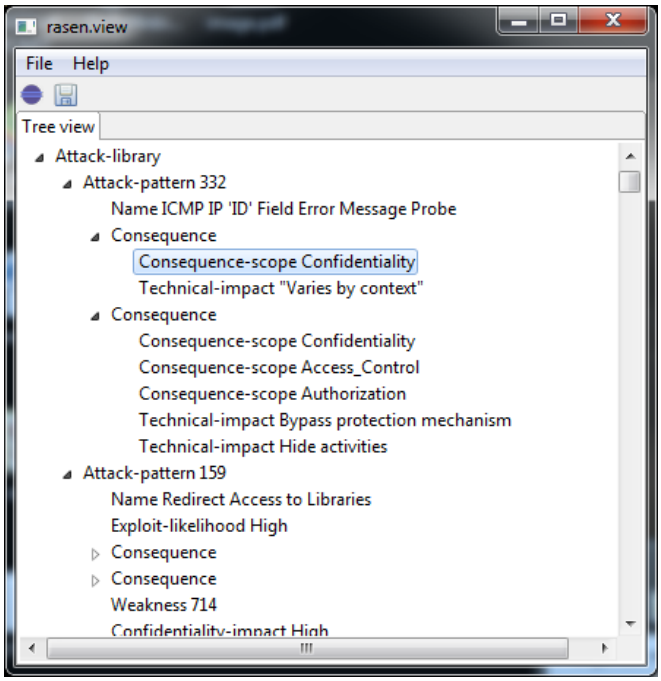

Figure 2 - Screenshot of Capec2Coras tool

After starting the Capec2Coras tool, the user is presented with a tree view as shown in Figure 2. This is where the CAPEC attack patterns and the parameters of the transformation can be edited. The tool has two main functions:

- Open: This function allows two kinds of files to be opened and displayed in the tree view of the Capec2Coras tool. The first kind of file is an XML-file containing a CAPEC library of attack patterns. The second kind is a file (not XML) which contains a previously edited version of the imported CAPEC library into the Capec2Coras tool.
- Save as: This function allows two kinds of saves (depending on the extension of the file being saved). The first kind transforms and saves the file into a CORAS-file (if the file extension ".coras_project" is used). The second kind saves the CAPEC library into a file so that he edits made to the file are not lost (if the file extension is not ".coras_project").

After a CAPEC library has been opened in the Capec2Coras tool, a tree view containing two top nodes is displayed as shown in Figure 3. These are "Attack-library" and "Default-likelihoods". The former contains all the attack-patterns which are defined in the CAPEC library, the second contains values which are used as parameters by the transformation. Specifically, default values which are produced by the transformation to CORAS can be configured as shown in Figure 3.

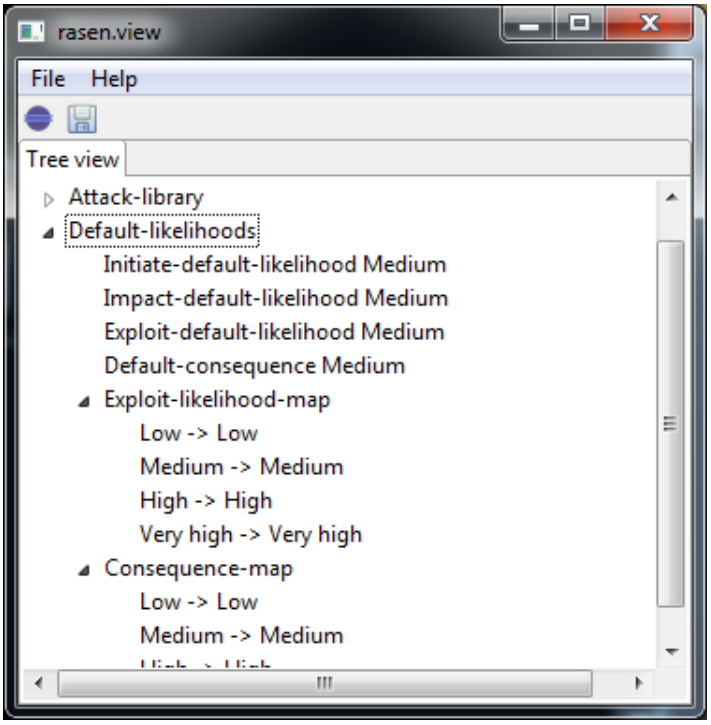

Figure 3 - Screenshot showing transformation parameters

1.3 CorasAnalyzer

Coras analyzer is a program that can be used to analyze Coras risk models. A screenshot of the tool is shown in Figure 4. Here we can see that the analysis functions are separated into two areas:

- *Core analysis* allows the user to perform the basic analysis functions: Check consistency, calculate likelihoods, and export risks.
- Risk-based-testing functions which allow the user to generate prioritized test procedures from a risk model and to aggregate test results to the risk model. Specifically, the following analysis features are supported: "Risk model to test procedures", Test report to measurements", "Calculate measurements", and "Measurements to risk model".

1.3.1 Installation

CorasAnalyzer

- Download the zip-file containing the current version of the tool.
- Extract the zip-file to any folder of your choosing.
- Double click the file "RUN_CORAS_ANALYZER.BAT" located in the CorasAnalyzer folder of the distribution.

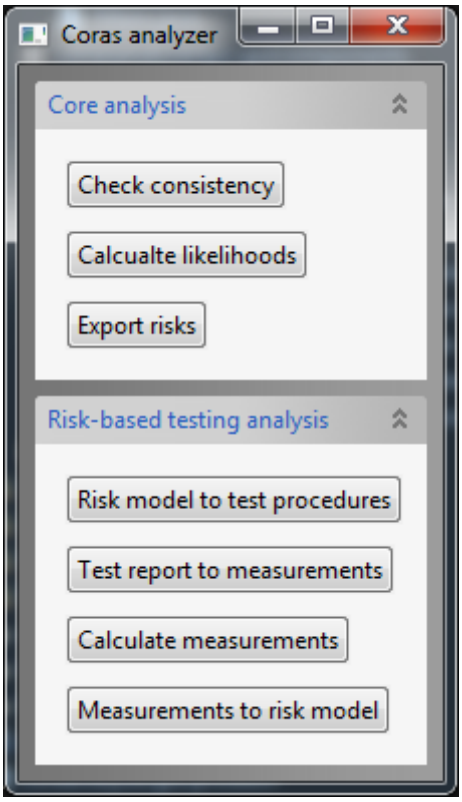

Figure 4 - Screenshot showing main Coras analysis window

1.3.2 User guide

In the following, we describe each analysis feature in more detail.

1.3.2.1 Check consistency

The check consistency feature allows the user to check whether the likelihood estimates of a Coras risk model are consistent. A screenshot of the check consistency feature is shown in Figure 5. Here we can see that the function takes three files as input (two of which are optional) in addition to one numeric parameter:

- Risk model infile: Refers to a Coras risk model file (with the .coras_project extension) whose consistency we want to check.
- Type definition (optional): Refers to an xml file defining the types of the likelihood values of the risk model. See Section **Error! Reference source not found.** for more details.
- Risk model out file (optional): This refers to a Coras risk model file which will contain an example of a consistent risk model which has been generated on the basis of the input risk model.
- Precision: This parameter specifies the precision with which consistency is checked.

By pressing the "Check consistency" button, the tool will load the risk model in file and check whether its likelihood values are consistent up to the specified level of precision. If the model is found to be consistent, and the risk model out file is specified, the tool will store an example of a consistent risk model to that file.

Example: The check consistency function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml which are located in the Coras Analyzer distribution.

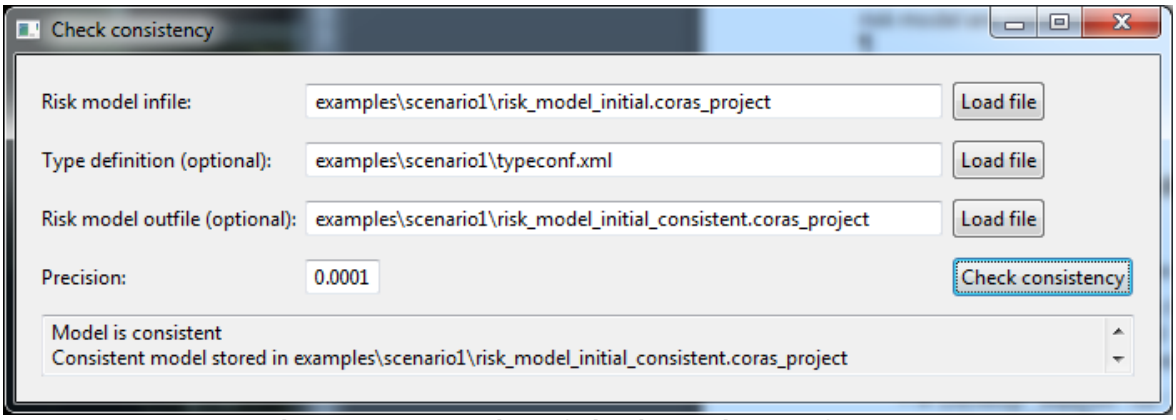

Figure 5 - Screenshot of check consistency feature

1.3.2.2 Calculate likelihoods

The calculate likelihoods function allows the likelihoods of a risk model which have not been estimated to be derived based on likelihood values that have been estimated. In addition, the function ensures that likelihood estimate ranges never include values which would always lead to inconsistency. As shown in Figure 6, the function takes three input values

- Risk model in file: Refers to the Coras risk model whose likelihood values will are to be calculated.
- Type definition (optional): Refers to an xml file defining the types of the likelihood, consequence, and risk values values of the risk model. See Section **Error! Reference source not found.** for more details.
- Risk model out file: Refers to the Coras risk model where the results of the likelihood calculation will be stored.

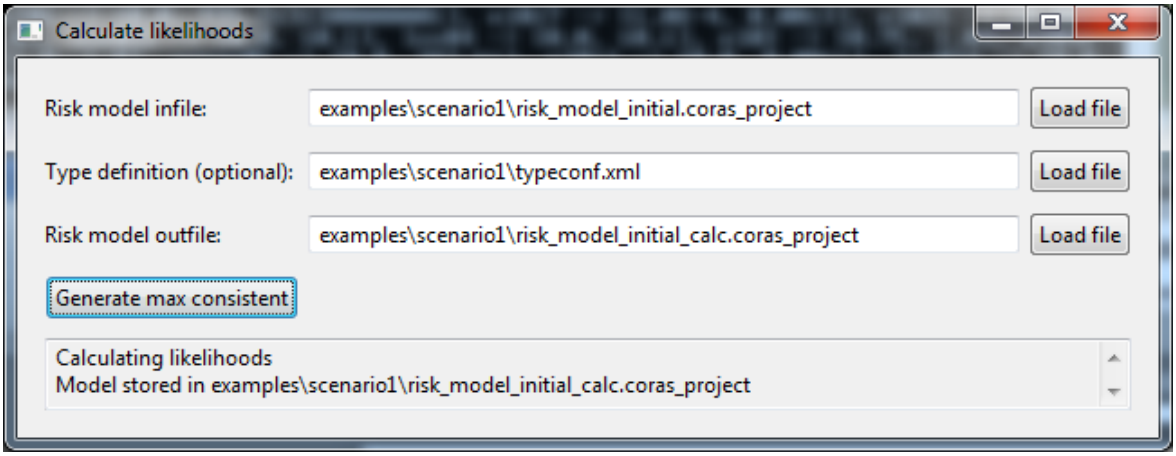

Figure 6 - Screenshot of the Calculate likelihoods function

If the "Generate max consistent" button is pressed, the tool will read the risk model in file and the type definition file (if any), calculate the likelihoods, and store the results in the risk model out file.

Example: The calculate likelihoods function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml which are located in the Coras Analyzer distribution.

1.3.2.3 Export risks

The export risks function allows the Coras risk model to be exported to comma-separated file containing all relevant information about all risk risks in the risk model. A risk in a Coras risk model is an unwanted incident which is connected to an asset. The function assumes that each unwanted incident constituting a risk is connected to exactly one asset. As shown in Figure 7, the export risks function takes three parameters:

- Risk model in file: Refers to the Coras risk model file whose risks will be exported
- Type definition (optional): Refers to an xml file defining the types of the likelihood values of the risk model. See Section **Error! Reference source not found.** for more details.
- Risks export file: Refers to the file where the risks will be stored. The file should have the extension .csv, since the risks are exported as comma-separated text file.

If the user presses the "Export risks" button, to tool will read the risk model in file and the type definition file (if any), calculate the risk values, and export the risks with the risk values to the risks export file in a comma separated text format.

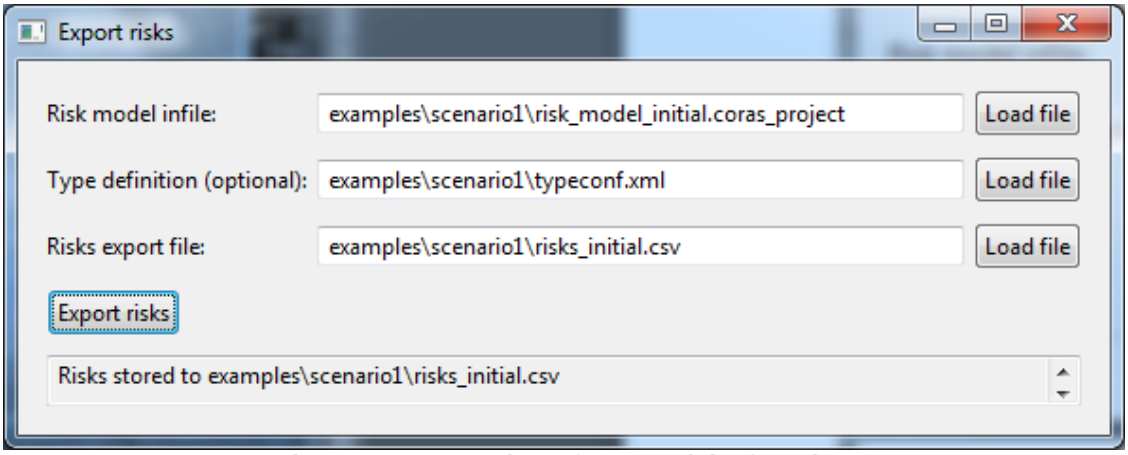

Figure 7 - Screenshot of export risks function

Example: The export risks function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml which are located in the Coras Analyzer distribution.

1.3.2.4 Risk model to test procedures

The risk model to test procedures function allows the user to generate a prioritized list of test procedure from a Coras risk model. These test procedures are meant to be used as a starting point for a test design and implementation activity. As shown in Figure 8, the function takes three input parameters:

- Risk model infile: Refers to the CORAS risk model on the basis of which the test procedures will be generated.
- Type definition (optional): Refers to an xml file defining the types of the likelihood, consequence, and risk values values of the risk model. See Section **Error! Reference source not found.** for more details.
- Test report out file: Refers to the file where the test procedures will be stored. Two different file formats are supported: If the out file ends with the .csv extension, then a comma-separated text file will be generated. Otherwise the test procedures will be stored in an XML file which conforms to the RASEN Test report XSD schema described in deliverable D5.4.3. This file

may be seen as an initial empty test report which can be populated with actual test results in the test execution phase.

If the user presses the button "Generate test report", then the tool will read the risk model in file and the type definition file (if any), generate a prioritized list of test procedures, and store the results to the test report out file.

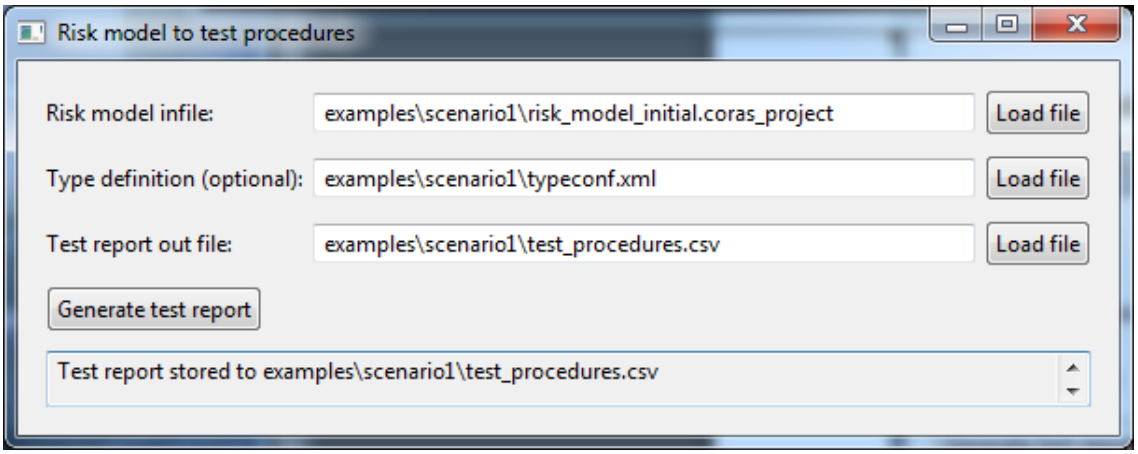

Figure 8 - Screenshot of risk model to test procedures function

Example: The "risk model to test procedures" function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml which are located in the Coras Analyzer distribution.

1.3.2.5 Test report to measurements

The "test report to measurements" function takes a test report as input, and generates an aggregated test report which contains measurements that can be used for the purpose of updating a risk model. As shown in Figure 9, the function takes four arguments:

- Risk model: Refers to a Coras risk model which was used to produce an initial version of the test report (using the risk model to test procedures function described in Section Figure 8.
- Type definition (optional): Refers to an xml file defining the types of the likelihood, consequence, and risk values values of the risk model. See Section **Error! Reference source not found.** for more details.
- Test report: An XML file containing the test results on the basis of which the aggregated test report with measurements will be generated. The test report must confirm to the TestReport XSD schema specified in deliverable D5.4.3.
- Aggregated test report (out): Refers to the file where the aggregated test report with the measurements are stored. Two formats are support. If the file ends with ".csv", the aggregated test report will be stored in a comma-separated text file. Otherwise the aggregated test report will be stored in an XML file which conforms to the AggregatedTestReport XSD schema specified in deliverable D5.4.3.

If the user presses the "Generate measurements" button, the tool will load the risk model, the test report, and the type definition file (if any), generate the measurements, and stored these as an aggregated test report in the measurement file. Note that the reason the risk model is loaded as opposed to only the test report, is that some of the measurements may be generated on the basis of information contained in the risk model (but not in the test report).

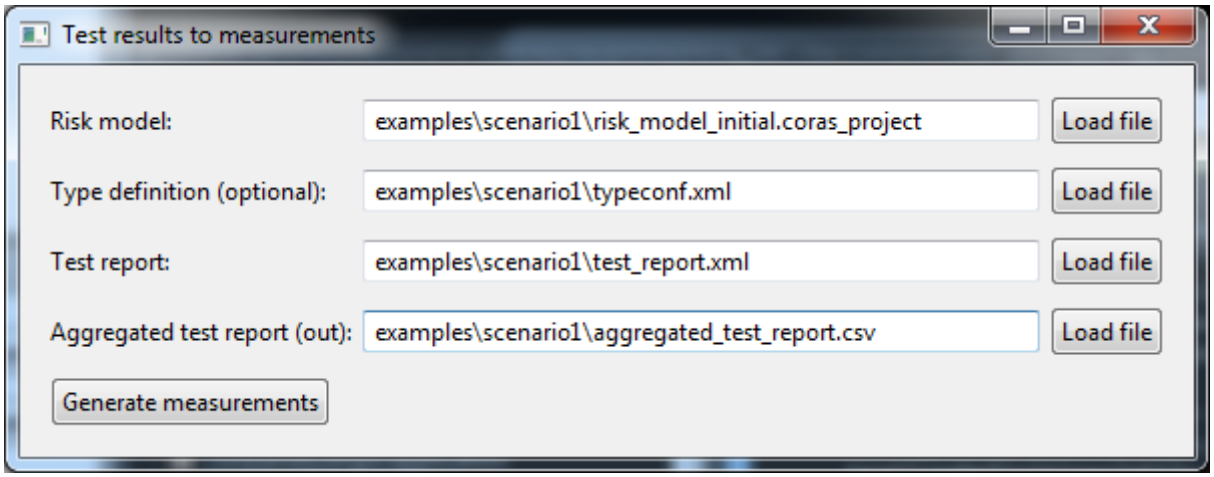

Figure 9 - Screenshot of the test results to measurements function

Example: The "test results to measurements" function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml, and the test report file examples/scenario1/test_report.xml" which are located in the Coras Analyzer distribution.

1.3.2.6 Calculate measurements

The "calculate measurements" function takes measurements of an aggregated test report as input, and based on some of these measurements, derives new measurements which are needed in order to update the a risk mode, and stores these together with the input measurements into a new aggregated test report file. As shown in Figure 10, the function takes two arguments

- Aggregated test report (in): The aggregated test report containing the measurements that are the basis for the derivation/calculation of new measurements. Two file formats are supported. If the file ends with the .csv extension, then it assumed that the aggregated test report is in a comma separated text file. Otherwise, the tool assumes that aggregated test report is in an XML file conforming to the AggregatedTestReport XSD schema documented in deliverable D5.4.3.
- Aggregated test report (out): Refers to the file where the new derived measurements together with the input measurements will be stored. As with the input file, xml and csv file formats are supported.

If the "calculate measurements" button is pressed, the tool will read the aggregated test report in file, calculate new measurements, and store these together with the input measurements to the out file.

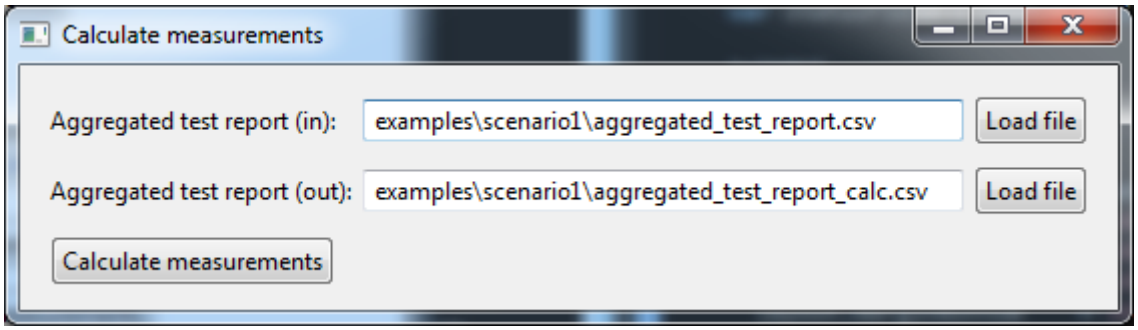

Figure 10 - Screenshot of the calculate measurements function

Example: The "calculate measurements" function can be tested on the aggregated test report in file examples/scenario1/aggregated_test_report.csv which is located in the Coras Analyzer distribution.

1.3.2.7 Measurements to risk model

The "measurements to risk model" function takes a risk model and an aggregated test report as input, and updates the likelihood estimates of the risk model based on the measurements contained in the aggregated test report. As shown in Figure 11, the function takes the following parameters:

- Risk model: Refers to a Coras risk model which contains the likelihood estimates that may be updated.
- Type definition (optional): Refers to an xml file defining the types of the likelihood, consequence, and risk values of the risk model. See Section **Error! Reference source not found.** for more details.
- Aggregated test report: An XML or CSV file an aggregated test report with the measurements that will be used to update (some of) the likelihood values of the risk model. This file should be generated using the "Calculate measurements" function described in Section 1.3.2.6.
- Risk model (out): Refers to the CORAS file where the updated risk model will be stored.

If the "Generate risk model" button is pressed, the tool will read the input risk model, the type definition, and the aggregated test report, update the likelihood values of the risk model based on the measurements of the aggregated test report, and store the results in the risk model out file.

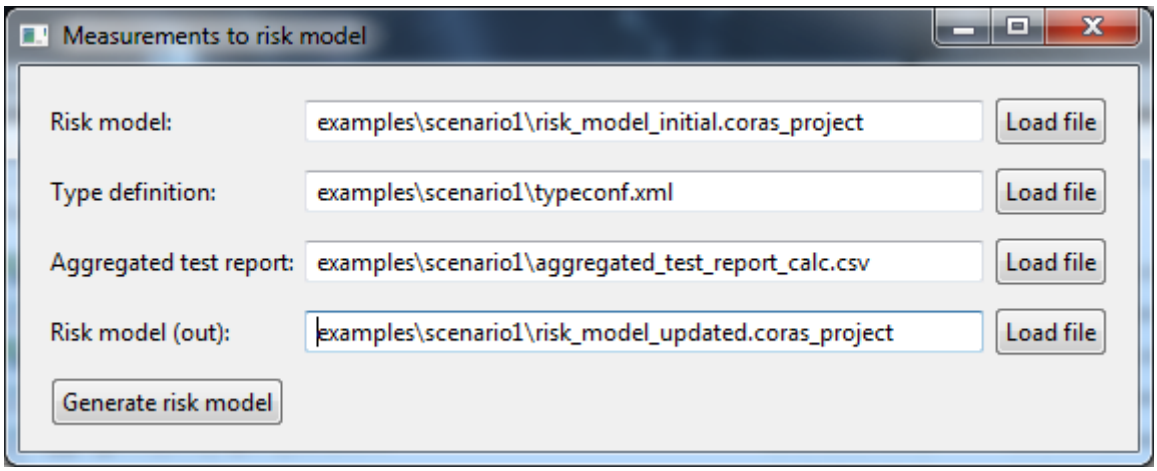

Figure 11 - Screenshot of the measurements to risk model function

Example: The "test results to measurements" function can be tested on the risk model in file examples/scenario1/risk_model_initial.coras_project and the type definition file examples/scenario1/typeconf.xml, and the test aggregated report file examples/scenario1/aggregated_test_report_calc.csv" which are located in the Coras Analyzer distribution.

1.3.2.8 The type configuration file

The type configuration file is an XML will which contains list of *type definitions*, and list of *type assignments* to elements of the risk model. The main purpose is to specify whether frequency or probability estimates should be used, the specify the risk function kind, and to specify precisely defined human readable qualitative values such as "Low", "High", etc. to numeric values which are need for automated analysis.

Each type definition specifies:

- The *name* of the type.
- A *basetype*, where the following values are possible: Frequency, Probability, Real, RiskFunctionAddition, RiskFunctionProduct. The latter two specifies whether the risk level be calculated by adding or multiplying the likelihood and consequence values that constitute the risk.
- A list of literal definition which map qualitative human readable values such as "Low", "High" into specific values. A literal definition consists of a *name*, a *description*, and a value definition.

An example of a type definition is shown below:

```
<typedef name="NodeLikelihoodType" basetype="Frequency">
  <literal name="Seldom" description="Attack initiation scale">
     <value><intervalReal min="0.0" max="0.1"/></value>
  </literal>
  <literal name="Unlikely" description="Attack initiation scale">
      <value><intervalReal min="0.1" max="1.0"/></value>
  </literal>
  <literal name="Possible" description="Attack initiation scale">
      <value><intervalReal min="1.0" max="13.0"/></value>
  </literal>
  <literal name="Probable" description="Attack initiation scale">
      <value><intervalReal min="13.0" max="61.0"/></value>
  </literal>
  <literal name="Certain" description="Attack initiation scale">
      <value><intervalReal min="61.0" max="10000.0"/></value>
  </literal>
</typedef>
```

In this example, we define a type with name NodeLikelihoodType, and basetype Frequency. In addition, there are five literals defined with names such as "Seldom", "Possible", etc. If these names are used in the risk model, they will be automatically replaced by their value as defined by the literals during analysis by the tool. As in the example, literals are usually defined as an interval with a minimum and maximum value.

The type assignments assign types to elements of the risk model. Each type assignment consists of:

- a *name* defining the kind of risk element that will be assigned a type, where the following values are possible: NodeLikelihood (referring to likelihood values of nodes in the risk model), EdgeLikelihood (referring to likelihood values of edges in the risk model), NodeConsequence (referring to consequence values of nodes in the risk model), NodeRisk (referring to risk values of nodes in the risk model).
- a *type* referring to the name of a type definition list.

The following shows an example of a list of type assignments:

```
<typeassignment name="EdgeLikelihood"  type="EdgeLikelihoodType"/>
<typeassignment name="NodeLikelihood"  type="NodeLikelihoodType"/>
<typeassignment name="NodeConsequence" type="NodeConsequenceType"/>
<typeassignment name="NodeRisk"        type="NodeRiskType"/>
```

In this example, four type assignments are defined. For instance, here the likelihood type of edges is set to the type with name EdgeLikelihoodType and the likelihood type of nodes is set of the type name NodeLikelihoodType.

The assignments may also be used to define mappings values to intervals. This is useful when risk evaluation criteria are defined in risk matrices. As an example, consider the risk matrix shown in the table below:

	Seldom	Unlikely	Possible	Probable	Certain
Insignificant	1	2	3	4	5
Small	2	3	4	5	6
Medium	3	4	5	6	7
High	4	5	6	7	8
Critical	5	6	7	8	9

Here columns represent likelihood values, and the rows represent consequence values. Combinations of likelihood and consequence values are mapped to one of the three risk levels: green, yellow, or red. In order to determine the risk level given a likelihood and a consequence value, we need to map these onto the appropriate rows and columns. This is the purpose of the mapping assignments.

A mappings assignment consists of

- a name defining the kind map assignment, where the following values are possible: LikelihoodMap (specifying that the map applies to likelihood values), ConsequenceMap (specifying that the map applies to consequence values), RiskMap (specifying that the map applies to risk values).
- a *type* referring to the name of a type definition list.

As an example, consider the following definitions:

```
<typeassignment name="LikelihoodMap"    type="NodeLikelihoodType"/>
<typeassignment name="ConsequenceMap"  type="NodeConsequenceMap"/>
<typeassignment name="RiskMap"          type="NodeRiskType"/>
```

For instance, here the map for likelihood values is defined by the type NodeLikelihoodType. The intervals of the likelihood values is defined by the literals in the NodeLikelihoodType. If we assume that this type is the one defined in the example above, then we can see that likelihood values in the range between 0 and 0.1, will be mapped to 1 (the first literal in the list), likelihood values in the range between 0.1 and 1.0 will be mapped to 2 etc.

The XSD schema defining the format of type definitions can be found in the folder *misc* located in the Coras analyzer tool distribution folder.