Compositional Risk Assessment and Security Testing of Networked Systems

Deliverable D5.4.1

A Toolbox for Security Risk Assessment and Security Testing

Project title:	RASEN
Project number:	316853
Call identifier:	FP7-ICT-2011-8
Objective:	ICT-8-1.4 Trustworthy ICT
Funding scheme:	STREP – Small or medium scale focused research project

Work package:	WP5
Deliverable number:	D5.4.1
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	2013-09-30
Actual delivery date:	2013-09-30
Responsible partner:	FhG FOKUS

Contributors

Editor(s)	Jürgen Großmann (FOKUS)
Contributor(s)	Fredrik Seehusen (SINTEF), Fabien Peureux, Bruno Legeard (Smartesting), Jürgen Großmann, Martin Schneider (FOKUS), Frank Werner (Software AG).
Quality assuror(s)	Arthur Molnar (Info World), Frank Werner (Software AG)

Version history

Version	Date	Description
0.1	13-06-10	TOC
0.2	13-08-20	Initial content
0.3	13-09-06	FOKUS and SAG input
0.4	13-09-06	Introduction and Conclusion
0.5	13-09-09	SINTEF input
0.6	13-09-10	Finalization
0.7	13-09-18	Reviewer feedback integration
1.0	13-09-26	Final quality check

Abstract

The RASEN risk assessment and security testing toolbox provides tool support for the RASEN approach to risk-based security testing and test-based security risk assessment. This deliverable contains the specification of the RASEN Data Integration Model and the specification of the interfaces between the tools of the RASEN toolbox.

Keywords

RASEN, Security Risk Assessment, Security Testing, Tool Integration

Executive Summary

The RASEN risk assessment and security testing toolbox provides tool support for the RASEN approach to risk-based security testing and test-based security risk assessment. This deliverable contains the specification of the RASEN Data Integration Model and the specification of the interfaces between the tools of the RASEN toolbox.

Table of Contents

1 INTRODUCTION 5

2 UPDATE OF THE CONCEPTUAL RASEN TOOLS 7

3 UPDATE OF THE DATA INTEGRATION MODELS 9

3.1 OVERALL STRUCTURE 9
3.2 FOUNDATION 9
3.3 RISK ASSESSMENT 10
3.3.1 Risk Assessment Elements and Relations 10
3.3.2 Security Risk Assessment Elements 12
3.3.3 Risk Model 12
3.3.4 Relation between Risk Assessment and System 13
3.4 TESTING 13
3.4.1 Testing Elements 13
3.4.2 Test Plan 14
3.4.3 Test Procedure 15
3.4.4 Test Report 15
3.5 RELATION BETWEEN RISK ASSESSMENT AND TESTING 15

4 INTEGRATION INTERFACES 17

5 SUMMARY AND OUTLOOK 21

REFERENCES 23

1 Introduction

The RASEN techniques and methodologies aim explicitly for the integration of data from security risk assessment and security testing. D5.2.1 [11] already provided first ideas how this integration could be addressed. This deliverable refines the ideas from D5.2.1 with respect to the work that has been done for D5.3.1 [12], i.e. for the RASEN methodologies and the RASEN conceptual model. This deliverable specifies the integration interfaces of the RASEN tools. Since we are currently planning the integration on the level of data exchange and exchange formats, the interface definitions are given by means of the data that are to be exchanged. In general the tool integration approach has been defined as follows:

1. Create deployable generic security testing (ST model) and security risk assessment and management model (SRAM model). These models will be designed in UML.
2. Create a deployable generic risk assessment and testing model (SRAT model) on basis of the ST model and the SRAM model.
3. Define conceptual tools and their interfaces
4. Instantiate a common XML-based exchange format on basis of the SRAT model.
5. Develop export/import adapter for the relevant instantiations of the conceptual tools (concrete tools) with respect to the XML format from 4 and the interfaces from 3.
6. Identify additional integration use cases and update the models if necessary.

This document describes the results for the steps 1 to 3 (bold face). Section 2 presents the definition of the conceptual RASEN tools and their relationship to the concrete RASEN tools, which are provided by the different partners. The concrete RASEN tools have been described more closely in D5.2.1. Section 3 provides an update of the data integration model and updates generic data models that are used as a basis for describing the data flow between the RASEN tools. The data models are intended to cover the major security risk assessment and testing tools that are used in RASEN. Based on these data models, Section 4 specifies the integration interfaces for the RASEN toolbox. Section 5 concludes the results of the deliverable and provides an outlook on the next steps to be taken to integrate the RASEN tools. Since there is an overlap between this deliverable and D5.3.1 we recommend to read D5.3.1 before reading this deliverable.

2 Update of the Conceptual RASEN Tools

A closer investigation of subject and partner tools shows that a more precise differentiation of tools is necessary to profoundly describe the interaction between them. The following table introduces six different conceptual tools as targeted by the RASEN project. This table is an update of a similar table in D5.2.1 [11].

Conceptual tool	Concrete tool
Security Risk Assessment Tool (SRAT) for supporting compositional risk assessment	**CORAS** risk assessment tool (developed by SINTEF) **ARIS Business Architect** – security risk assessment tool by RASEN Extension (Developed by Software AG)
Security Test Pattern Database (PDB) for managing test patterns	**Fokus!MBT** (developed by Fraunhofer)
Security Test Derivation Tool (STDT) for supporting the derivation and prioritization of test cases based on the risk assessment	**CertifyIt for Security Testing** (developed by Smartesting) **Fokus!MBT** (developed by Fraunhofer)
Security Testing Tool (STT) for adapting the test item and executing test cases	**CertifyIt for Security Testing** (developed by Smartesting) **Fokus!MBT** (developed by Fraunhofer) Any other testing tool
Security Test Management Tool and Test Result Aggregation Tool (STRAT) for supporting the aggregation of security test results into a format that allows us to verify the risk picture and to update the risk picture based on the results	**Fokus!MBT** (developed by Fraunhofer) **CORAS risk monitor prototype** (developed by SINTEF), which defines rules for updating the risk assessment at run-time
Security Risk Management Tool (SRMT) for continuously managing the results from risk assessments.	**ARIS Business Architect** (developed by Software AG).

Table 1 – Conceptual RASEN tools

Figure 1 shows the direct relationship between the conceptual RASEN tools and the concrete RASEN tools. The conceptual RASEN tools are represented by UML interfaces and the concrete RASEN tools are specified by UML classes. We consider that the integration interfaces for data exchange are defined on the level of the RASEN conceptual tools, thus by means by the UML interfaces that represent these tools. Thus, we can provide a flexible mapping between the conceptual RASEN tools and the concrete RASEN tools by considering that each concrete RASEN tool that takes the role of a conceptual RASEN tool has to formally implement the interface that represents the conceptual RASEN tool.

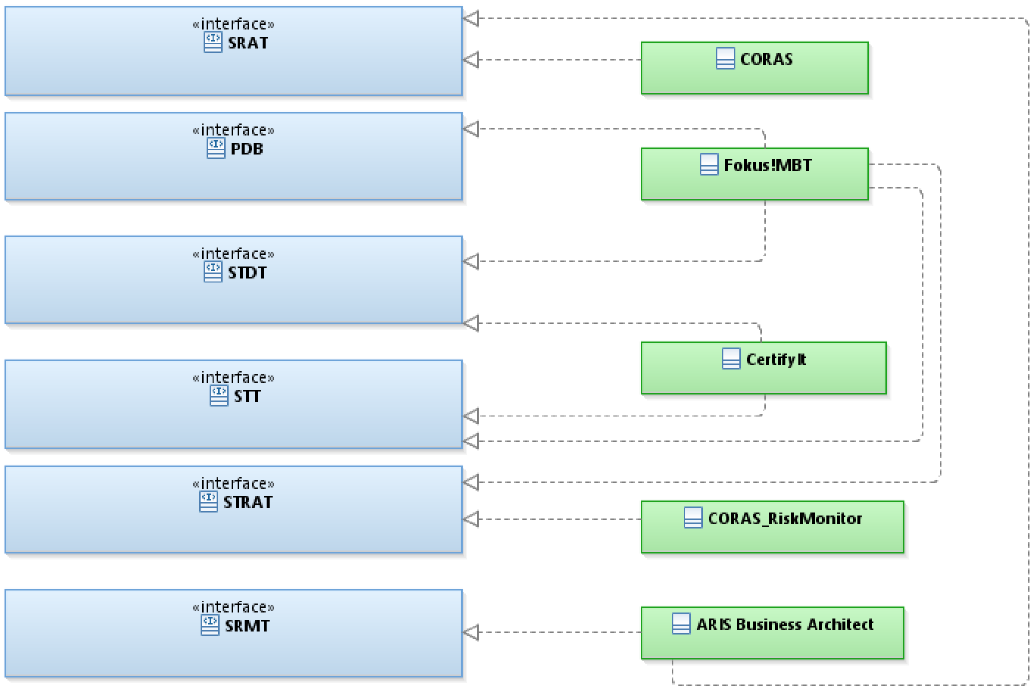

Figure 1 – UML diagram modeling the relationship between conceptual and concrete tools

3 Update of the Data Integration Models

In this section we define data models that serve as a basis for tool integration at the data level. The models are based on the initial models defined in D5.2.1 and are refined with respect to the RASEN conceptual model given in D5.3.1. Since the data integration models address the technical level of data integration they differ from the RASEN Conceptual Model in the following aspects:

- the Data Integration Model provides a set of common base classes that allow for a common identification scheme of the model elements
- the Data Integration Model provides a set of classes that allow for depicting values, their types and respective scales. This is required to provide a flexible solution to exchange probability values, frequency values or other forms of priority values that are to be exchanged
- the Data Integration Model lacks classes for some of the concepts from D5.3.1 that are not exchanged between tools or can be integrated by means of attributes that contain informative text
- the Data Integration Model contains classes that represent model elements from other models (e.g. system models). These references are covered by an attribute that allows for specifying an URI that addresses elements that are not covered by the integration model

However, since the RASEN Conceptual Model serves as a template for the Data Integration Model, we carefully ensure that the model elements in this deliverable have the same name and meaning than the model elements in D5.3.1.

3.1 Overall Structure

The RASEN Data Integration Model consists of three packages. The *testing* package contains the classes that describe the test data to be exchanged, the *riskassessment* package contains the classes that describe the risk assessment data to be exchange and the *foundation* package contains foundation classes that are used by the *testing* and *riskassessment* packages. Figure 2 shows the relationship between the packages.

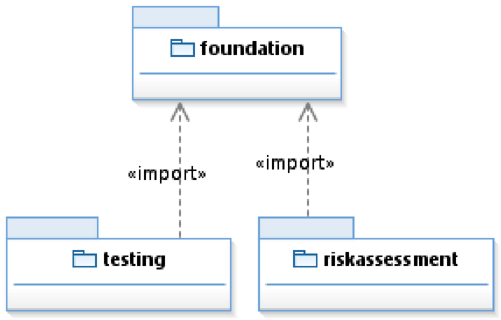

Figure 2 – Package structure

3.2 Foundation

The *foundation* package contains the abstract class *Element* and set of classes that allow for depicting values, their types and respective scales. The *Element* class has three attributes that allow for providing a unique identifier (attribute *identifier*), a descriptive user defined name (attribute *name*), and a user defined description (attribute *description*). The class Element is the basis for the definition of a set of concrete classes in the packages *testing* and *riskassessment.*

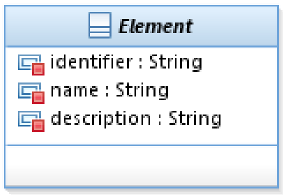

Figure 3 – The Element class

Figure 4 shows the classes, which are used to express types and values such as likelihood values, consequence values, priority values and etc. The class *ValueType* is used to express basic types such a Real or Integer, whereas the class *CustomType* together with *TypeProperty* and *LiteralDefinition* are used to express more complex custom types. This is for instance needed in order to capture likelihood and consequence scales (in the risk assessment domain) and to distinguish between different kinds of likelihood values such as probability, frequencies, intervals of these and properties of likelihoods such as mutual exclusiveness or statistical independence. The *ValueElement* class is the class used to express any kind of type value.

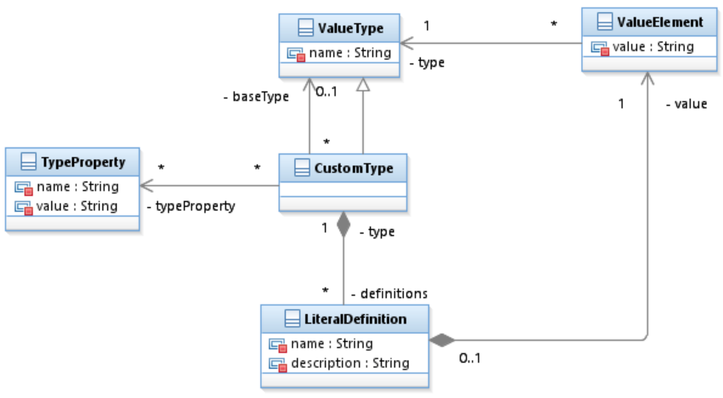

Figure 4 – The ValueElement and type classes

3.3 Risk Assessment

3.3.1 Risk Assessment Elements and Relations

In this section we define classes for expressing risk assessment notions. First, in Figure 5, we define the two abstract classes that will be the super-types of all other risk assessment classes: The class *RAElement* is used to express risk assessment elements whereas the class *RARelation* is use to express relations between these elements.

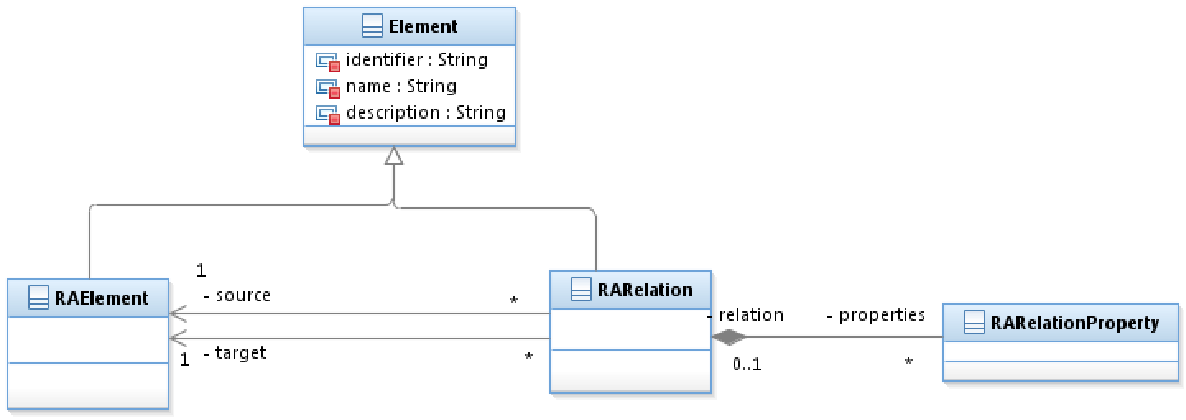

Figure 5 – Abstract risk assessment elements and relations

Figure 6 shows the main risk assessment elements (identified in the conceptual model) as well as the values that are commonly used in a risk assessment. As can been seen in the diagram, all the risk assessment classes are subtypes of the *RAElement* class, whereas all the risk assessment values are subtypes of the *ValueElement* class.

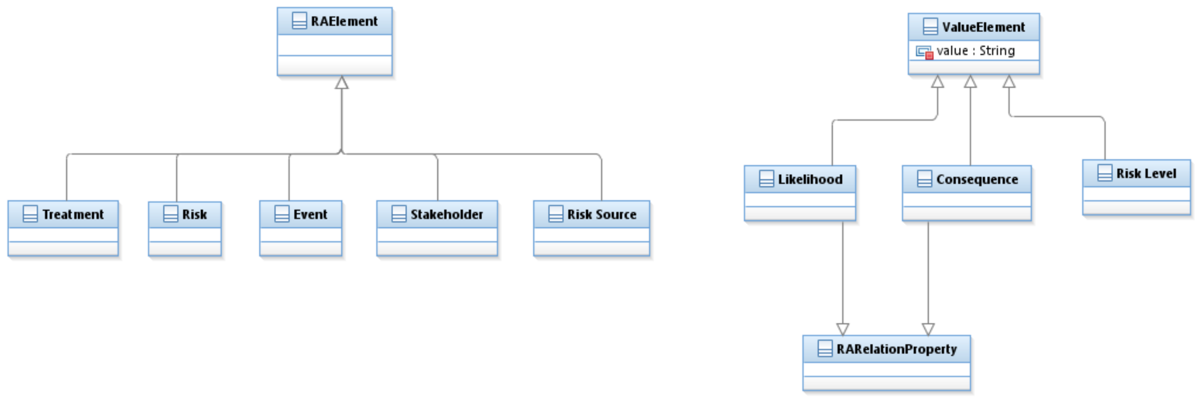

Figure 6 – Concrete risk assessment elements and values

Figure 8 shows the how the main risk assessment elements are related. This corresponds to the conceptual model defined in D5.3.1.

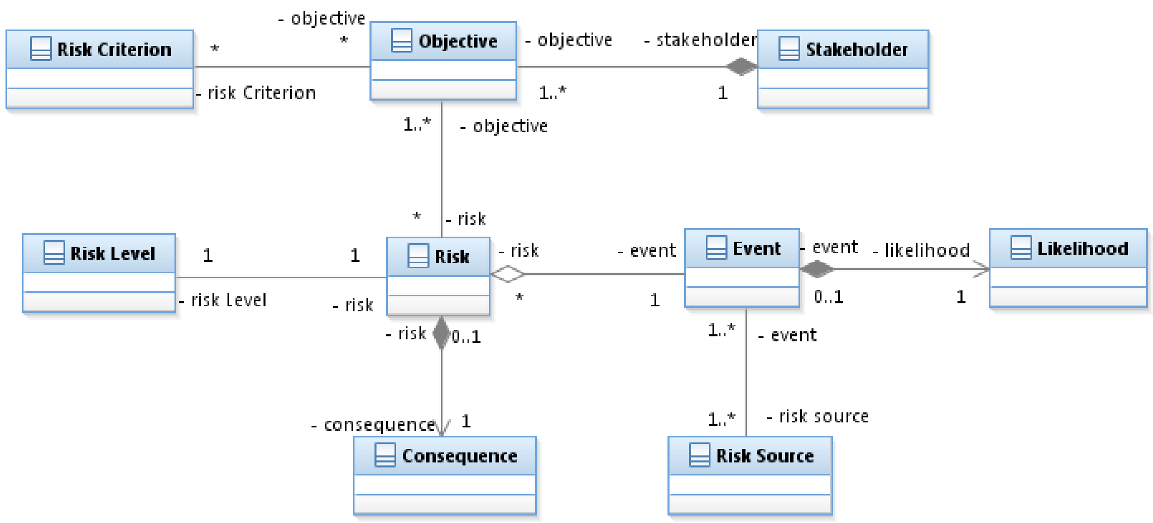

Figure 7 – Relationship between risk assessment elements

3.3.2 Security Risk Assessment Elements

The classes used to express the notions related to security risk assessment are introduced in Figure 8. Again, these classes correspond to the notions defined in the conceptual model (D5.3.1 [12]).

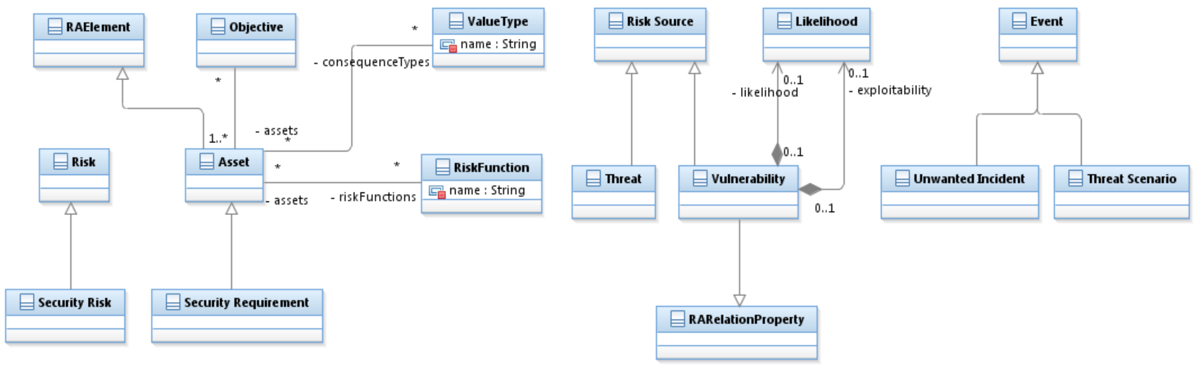

Figure 8 – Security risk assessment elements

3.3.3 Risk Model

Figure 9 defines the classes for expressing a risk model. As can be seen from the diagram, a risk model consists of a set of risk assessment elements and a set of risk assessment relations between these elements. In addition, the risk model may contain a set of types used to express likelihood and consequence scales, as well as a set of risk functions.

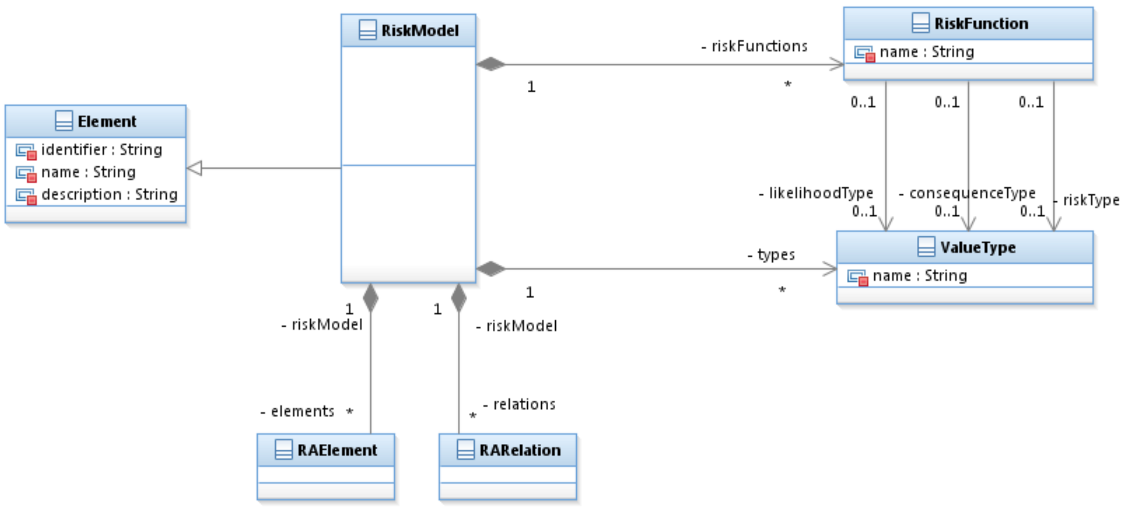

Figure 9 – The risk model

3.3.4 Relation between Risk Assessment and System

Figure 10 shows the *SystemElement,* which used to indirectly express traceability between system elements and risk assessment elements or relations. This class contains attribute *url* which should be used to refer to an element of the system model.

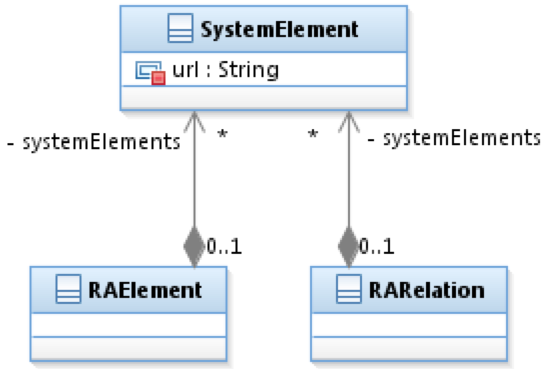

Figure 10 – Relation between risk assessment and system

3.4 Testing

The testing package is conceptually aligned with the testing and security testing packages of the RASEN conceptual model and thus conform to the IEEE standards 829 [4] and 29119 [3]. The most central elements describe classes for typical testing artifacts or documents, namely a **test plan**, a **test procedure** and a **test report** that are the subject of exchange between process steps and thus between the related tools. The following sections provide an overview over the whole set of testing elements defined in the testing model. In particular, it explicitly define the central artifacts test plan, test procedure and test report.

3.4.1 Testing Elements

The *testing* package defines the set of elements that are depicted in Figure 11. The individual elements and their meaning are described in Deliverable D5.3.1.

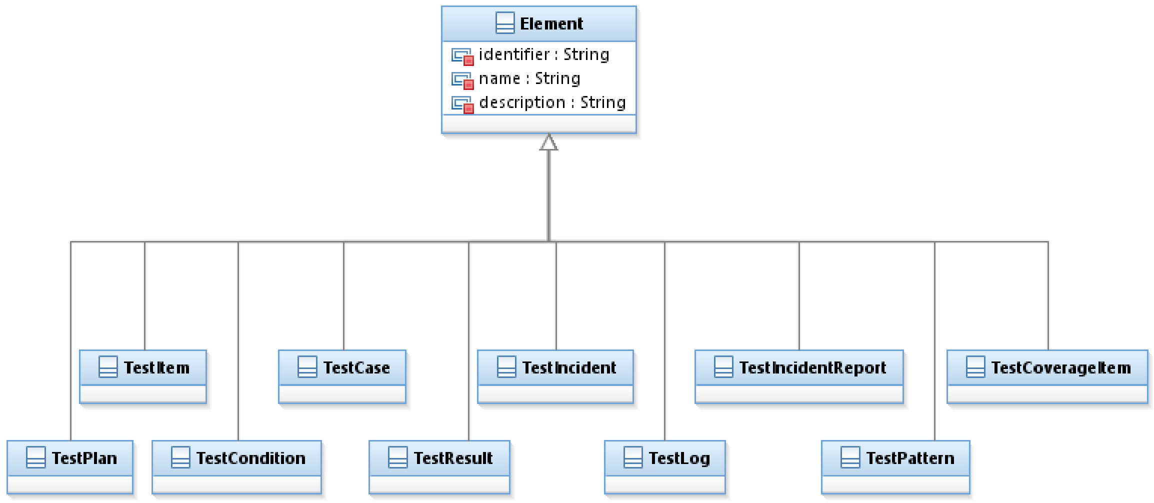

Figure 11 – Testing elements

3.4.2 Test Plan

A *TestPlan* provides a detailed description of the test objectives to be achieved and specifies the means and schedule for achieving them. It is used to organize and coordinate testing activities for a test item or a set of test items. The *TestPlan* class specifies the properties of such a *TestPlan*. A *TestPlan* contains a list of *CoverageItem*s, a set of *TestPattern* and a set of *TestConditions*. It is associated to a *TestItem* that specifies the item under test. A *TestItem* may refer to elements of an external model by means of the attribute *systemModelElement*.

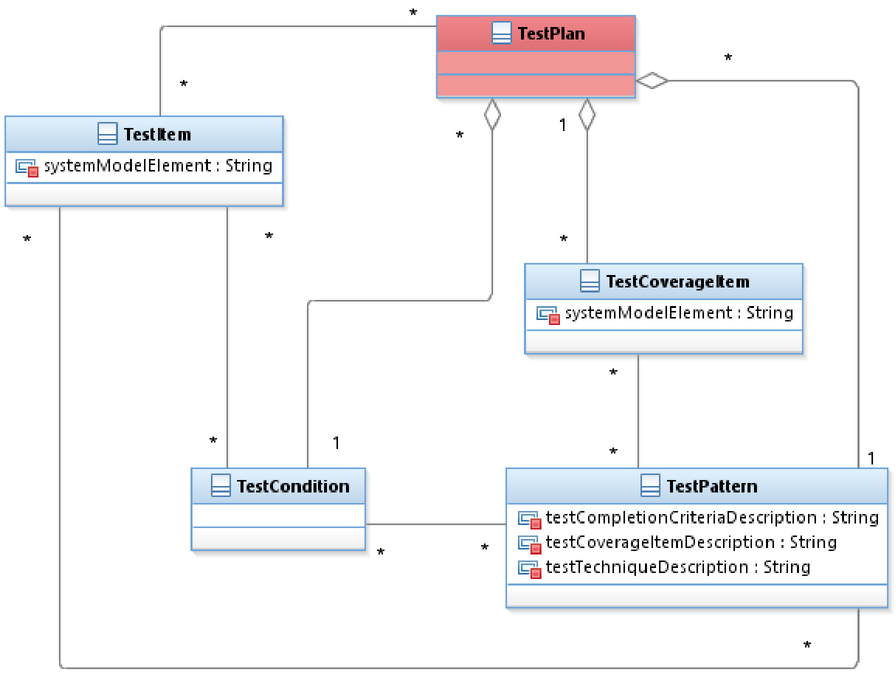
TestPlan
TestItem
systemModelElement : String
TestCoverageItem
systemModelElement : String
TestCondition
TestPattern
testCompletionCriteriaDescription : String
testCoverageItemDescription : String
testTechniqueDescription : String
*
1

Figure 12 – Test plan structure

3.4.3 Test Procedure

A *TestProcedure* contains a sequence of *TestCases* in execution order. It holds a reference to the *TestItem* that refers to the item under test.

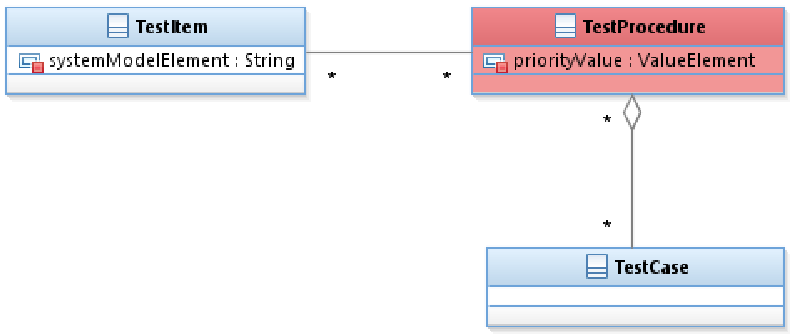

Figure 13 – Test procedure structure

3.4.4 Test Report

A *TestReport* contains the *TestLog* and the *TestIncidentReport* and associates a *TestItem.* Each *TestLog* contains the *TestResults* of a test run. The *TestResult* is qualified by its *verdict* attribute that may evaluate to *none, pass, inconclusive, error* or *fail*. Each *TestIncidentReport* contains the *TestIncidents* of a test run.

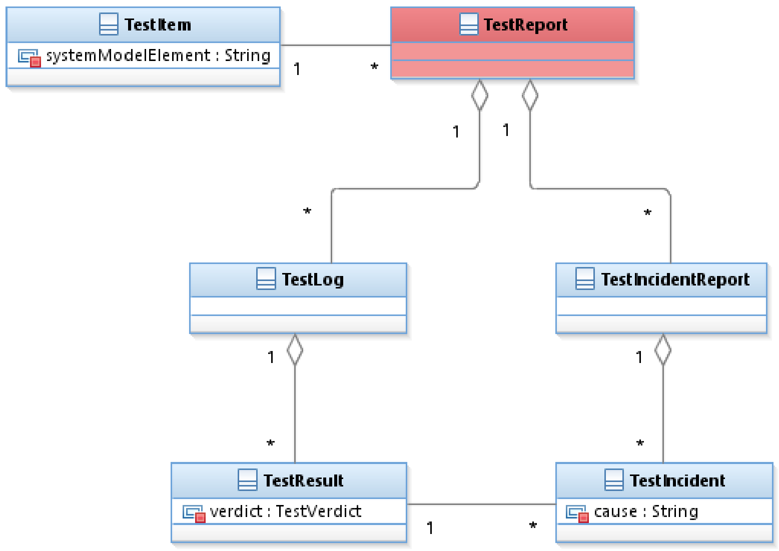

Figure 14 – Test report structure

3.5 Relation Between Risk Assessment and Testing

The Test Pattern supported Risk Based Security Testing methodology in D5.3.1 describes test pattern as a central element of the integration of risk assessment and testing. This relationship is depicted in Figure 15 by the associations between the *TestPattern* class and the *RAElement* class.

The *TestProcedure* is the other main class used for linking the risk assessment to the testing domain. In Figure 15 this is expressed by the association between the *TestProcedure* class from the testing domain and the *RAElement* and *RARelation* classes from the risk assessment domain.

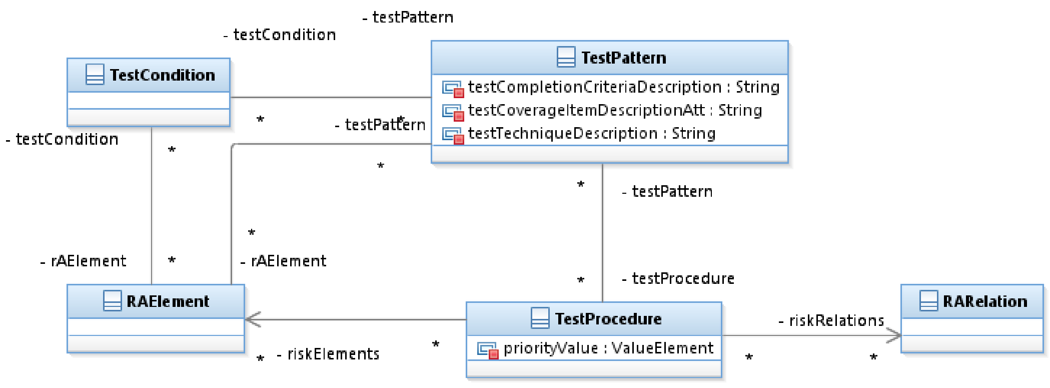

Figure 15 – Relation between risk assessment and testing

4 Integration Interfaces

This section specifies the integration interfaces for the conceptual tools that have been identified in Section 2. The interface definition is based on the integration use cases from [11] and the methodology definition in [12]. It denotes required and provided data for each of the conceptual tools. The tables below denote each of the conceptual RASEN tools, specify the concrete RASEN tools that take the role of the conceptual tool, and finally define the data, which is provided and required. Data specified in terms of the conceptual model from Section 2.

Conceptual Tool	**Security Risk Assessment Tool (SRAT)**
Concrete Tools	CORAS
Requires	testing::TestReport (see Section 3.4.3)
Provides	riskassessment::RiskModel 3.3.3

Table 2 – Security Risk Assessment Tool (SRAT)

Conceptual Tool	**Security Pattern Data Base (PDB)**
Concrete Tools	Fokus!MBT + RISKTest
Requires	
Provides	testing::TestPlan (see Section 3.4.2)

Table 3 – Security Risk Assessment Tool (SRAT)

Conceptual Tool	**Security Test Derivation Tool (STDT)**
Concrete Tools	Smartesting CertifyIt [14] Fokus!MBT + RISKTest [2]
Requires	riskassessment::RiskModel (see Section 3.3.3) testing::TestPlan (see Section 3.4.2)
Provides	testing::TestProcedure[*][1] (see Section 3.4.3)

Table 4 – Security Test Derivation Tool (STDT)

Conceptual Tool	**Security Testing Tool (STT)**

[1] Please note, the term [*] denotes that the signature contains a list of elements and not a single element.

Concrete Tools	Smartesting CertifyIt [14] Fokus!MBT + RISKTest [2]
Requires	testing::TestProcedure[*] (see Section 3.4.3)
Provides	testing::TestReport (see Section 3.4.4)

Table 5 – Security TestingTool (STT)

Conceptual Tool	**Security Test Management Tool and Test Result Aggregation Tool (STRAT)**
Concrete Tools	Fokus!MBT + RISKTest [2]
Requires	testing::TestReport (see Section 3.4.3) riskassessment::RiskModel (see Section 3.3.3)
Provides	testing::TestReport (see Section 3.4.3)

Table 6 – Security Test Management Tool and Test Result Aggregation Tool (STRAT)

Conceptual Tool	**Security Risk Management Tool (SRMT)**
Concrete Tools	ARIS Business Architect
Requires	testing::TestReport (see Section 3.4.3)
Provides	riskassessment::RiskModel (see section 3.3.3) testing::TestReport (see Section 3.4.3)

Table 7 – Security Risk Management Tool (SRMT)

The formal UML specification that is given in Figure 16 contains an *import* function for each entry in the required rows in the tables above and an *export* function for each entry in the provided rows in the tables above.

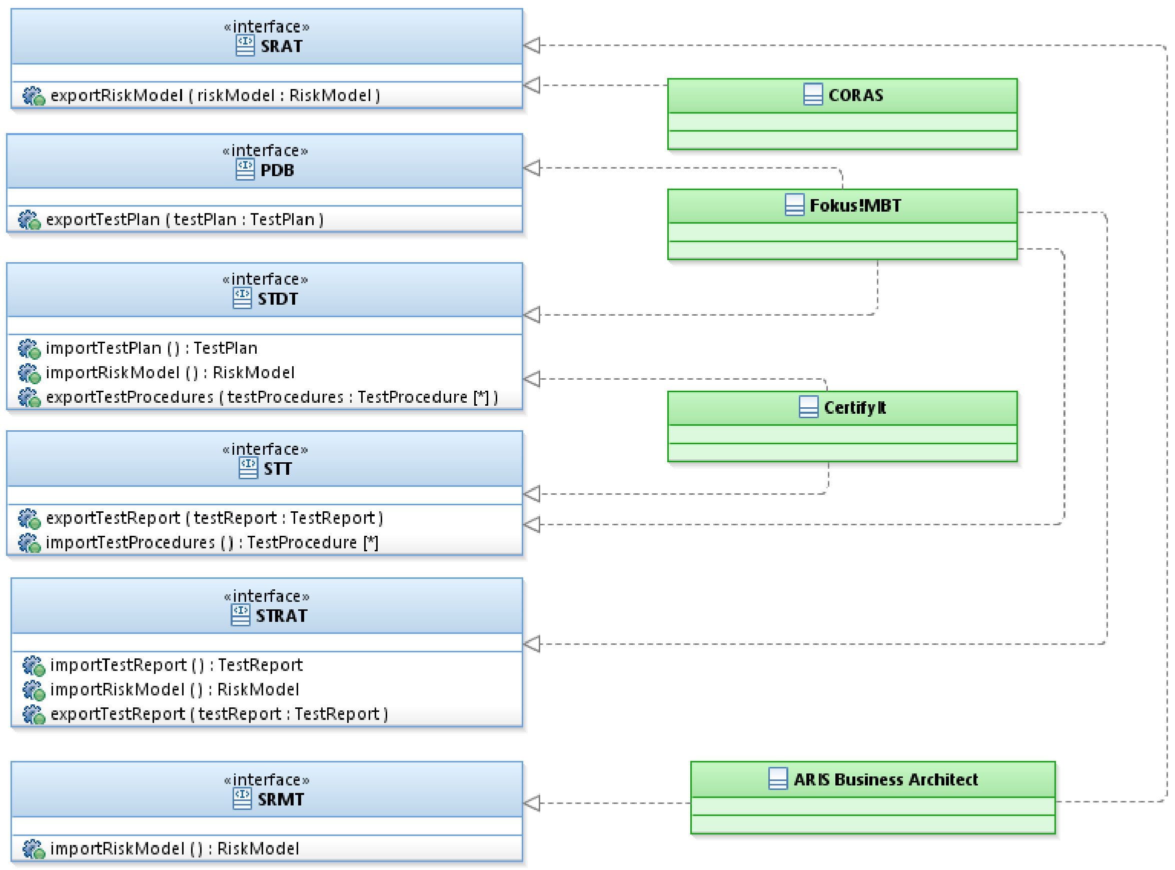
«interface»
SRAT
exportRiskModel (riskModel : RiskModel)
«interface»
PDB
exportTestPlan (testPlan : TestPlan)
«interface»
STDT
importTestPlan () : TestPlan
importRiskModel () : RiskModel
exportTestProcedures (testProcedures : TestProcedure [*])
«interface»
STT
exportTestReport (testReport : TestReport)
importTestProcedures () : TestProcedure [*]
«interface»
STRAT
importTestReport () : TestReport
importRiskModel () : RiskModel
exportTestReport (testReport : TestReport)
«interface»
SRMT
importRiskModel () : RiskModel
CORAS
Fokus!MBT
CertifyIt
ARIS Business Architect

Figure 16 – UML diagram modeling the interfaces for data exchange

5 Summary and Outlook

This deliverable has specified the RASEN Data Integration Model and interfaces for data exchange between RASEN tools. The Data Integration Model has been derived from the RASEN Conceptual Model from D5.3.1, which defines the basic terms and concepts for the RASEN methodologies and the Integration Use Cases from Deliverable D5.2.1. The Data Integration Model addresses the integration between risk assessment and testing and depicts the technical aspects that are necessary to practically exchange data between tools. The integration interfaces for the RASEN tools are specified on the level of so called conceptual tools that provide dedicated services for security risk assessment and security testing or a combination thereof. These interfaces directly refer to the data elements that are specified by the RASEN Data Integration Model.

Moreover this deliverables provides a mapping between the conceptual RASEN tools and the concrete RASEN tools. We consider that each concrete RASEN tool implements the interfaces of a subset of the conceptual RASEN tools, so that we can easily derive the interface implementation requirements for the concrete tools. As next step we will use the Data Integration Model to derive XML based exchange formats that can be used to persistently store the data to be exchanged. Thus the next deliverable will mainly address item 4 in the overall activity list below.

1. Create deployable generic security testing (ST model) and security risk assessment and management model (SRAM model). These models will be designed in UML.
2. Create a deployable generic risk assessment and testing model (SRAT model) on basis of the ST model and the SRAM model.
3. Define conceptual tools and their interfaces
4. Instantiate a common XML-based exchange format on basis of the SRAT model.
5. Develop export/import adapter for the relevant instantiations of the conceptual tools (concrete tools) with respect to the XML format from 4 and the interfaces from 3.
6. Identify additional integration use cases and update the models if necessary.

References

[1] F. John Reh: Glossary of business management terms and abbreviations. http://management.about.com/cs/generalmanagement/g/objective.htm [Accessed 19.04.2012]

[2] FhG FOKUS: Fokus!MBT + RISKTest, platform for model based testing from FhG FOKUS in [12] (2013)

[3] IEEE: IEEE 29119 Software and system engineering - Software Testing Part 1: Concepts and definitions. (2012)

[4] IEEE: IEEE Standard for Software and System Test Documentation (IEEE 829-2008). ISBN 978-0-7381-5747-4 (2008)

[5] IEEE: IEEE Standard Glossary of Software Engineering Terminology (IEEE 610.12-1990). ISBN 1-55937-067-7 (1990)

[6] International Standards Organization: ISO 27000:2009 (E), Information technology – Security techniques – Information security management systems – Overview and vocabulary (2009)

[7] International Standards Organization: ISO 31000:2009 (E), Risk management – Principles and guidelines (2009)

[8] ISTQB: ISTQB Glossary of testing terms version 2.2.http://www.istqb.org/downloads/finish/20/101.html, [Accessed 10.09.2013]

[9] OMG: UML testing profile version 1.1 (formal/2012-04-01). http://www.omg.org/spec/UTP/1.1 [Accessed 10.09.2013]

[10] RASEN: Deliverable 5.1.1, Baseline methodologies for legal, compositional, and continuous risk assessment and security testing (2013)

[11] RASEN: Deliverable 5.2.1, Risk assessment and security testing toolbox requirements and design (2013)

[12] RASEN: Deliverable 5.3.1, Methodologies for legal, compositional, and continuous risk assessment and security testing v.1 (2013)

[13] SINTEF: CORAS Risk Assessment Tool from SINTEF in [12] (2013)

[14] Smartesting: CertifyIt, tool suite for model based testing from Smartesting in [12] (2013)

[15] Software AG: ARIS Risk & Compliance Manager from Software AG in [12] (2013)