Test Management Tool for Risk-based Security Testing

Michael Berger (Fraunhofer FOKUS)

Michael.berger@fokus.fraunhofer.de
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Introduction of Traceability – Requirement Traceability

- Requirement Traceability

I. What is requirements traceability?

- A requirement may be traced in one of four distinct ways, according to Karl Weigers in his book *Software Requirements*

  - “Customer needs are traced forward to requirements, so that you can tell which requirements will be affected if those needs change.

  - Specific product elements [may be traced] backward to requirements so that you know why each item was created.”

- Conversely, you can trace backward from requirements to customer needs to identify the origin of each software requirement.
Introduction of Traceability – Requirement Traceability

- Requirement Traceability

II. Why is requirements traceability important?

- It ensures that final deliverables directly tie to initial business needs.

- Done properly, it ensures that organizations do not waste time and resources repeating research.

- It complies with established industry standards.

- If offers much easier impact analysis
Introduction of Traceability – Traceability in Safety

• Traceability in Safety

• For example, IEC 61508 recommends the usage of fault tree analysis, data flow diagrams, simulations, configuration management and structured programming.

• Most of the standards mandate traceability either as requirements that a system shall fulfil, or as a technique that shall be used while developing the system.

• Traceability is a mechanism for relating artefacts, and is vital for ensuring the completeness of the specification and for the system itself.
Introduction of Traceability – Definition

• Traceability defines relationships between different models.

• Such a relationship consists of at least a tuple of model elements and is called trace.

• The simplest kind is untyped tracing. A trace may be created between every kind of element in arbitrary models. It allows easily navigating between models and model elements.

• A more reasonable kind of tracing is typed tracing. That requires the definition of a trace metamodel to restrict the traces to specific element types.

• The typed tracing requires a defined trace metamodel. In trace metamodel the traces with related element types are specified.

• Services and analytic tools can get required information of the modelled parts and artefacts belonging the trace metamodel
Security Test Management Tool
CORAS, Papyrus, ProR and TTWorkbench
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Traceability Management Requirements

- The user has to define traces while developing model with a single tool.
- The traceability management has to offer tracepoint creation, editing and navigation.
- A special trace metamodel restrict the trace creation.
- The user can use supported services with this tool.
- The user can define new services for the model.
- If the user prefer another editor tool he has the chance to add this editor tool to the test managing tool.
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Abstract Domain Model Concept

• Each model is related to a certain **model domain**. We are currently supporting four domains:
  
  • **Risk Domain** (CORAS)
  
  • **System Domain** (UML, SysML)
  
  • **Requirement Domain** (ReqIF)
  
  • **Test Domain** (TTCN3)

• Each model domain is an abstraction of all different model kinds of this domain. To support another model kind, a **mapping to the domain model is necessary**.

© DIAMONDS Consortium 2010-2013
Abstract Domain Model Concept

- Each model is related to a certain **model domain**. We are currently supporting four domains:
  - **Risk Domain** (CORAS)
  - **System Domain** (UML, SysML)
  - **Requirement Domain** (ReqIF)
  - **Test Domain** (TTCN3)

- Each model domain is an abstraction of all different model kinds of this domain. To support another model kind, **a mapping to the domain model is necessary**.
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Test Managing Tool in Overview

Coverage Analyzing Service

Likelihood Analyzing Service

Services

Trace Management

Model Query Dispatcher

Trace Query Interpreter

Traceability Platform

Trace Metamodel

Trace Storage

Domain

Domain Query Interpreter

Metamodel Mapper

Tool Metamodell

Domain Metamodel

Domain Platform

Tool Metamodel

Metamodel Mapper

© DIAMONDS Consortium 2010-2013
Test Managing Tool in Overview – Trace Management

- Trace Menu
- Trace Editor
- Trace Explorer
- Trace Point Viewer

© DIAMONDS Consortium 2010-2013
Trace Meta-Model in Context of Risk-based Security Testing

Risk Domain

- Treatment
- Threat Scenario / Unwanted Incident
- Vulnerability

System Domain

- System Component
- Behavior Diagram
- Interaction Diagram

Requirement Domain

- Security Functional Requirement
- Security Test Requirement

Test Domain

- Test Case
- Test Environment

Model Relations

Trace Relations
Test Managing Tool in Overview – Domain Layer
Test Managing Tool in Overview

- Coverage Analyzing Service
- Likelihood Analyzing Service
- Trace Management
- Model Query Dispatcher
- Trace Query Interpreter
- Domain Query Interpreter
- Domain Metamodel
- Trace Metamodel
- Trace Storage
- Domain Platform
- Traceability Platform
- Services
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Test Managing Tool – Configuration Requirements

• To specify a new service you have to define queries on domain metamodel

• To integrate a new editor tool you have to select the belonging domain and

  • To specify the way to create a tracepoint for the selected element

  • To define a mapping between editor tool metamodel and domain metamodel
Model Relation Example between System Domain and UML

**Domain Level**

- Query Request

**Tool Level**

- Mapping

**Code Snippet**

```
mapElement
createDomainElement
getDomainElement
```
Model Relation Example between System Domain and UML

Domain Level

Query Request

Mapping

Tool Level

Domain Element

Tool Element

refElement

© DIAMONDS Consortium 2010-2013
Model Relation Example between System Domain and UML

System Domain

UML

© DIAMONDS Consortium 2010-2013
Outline

• Introduction to traceability

• Test Management Tool requirement and concept
  • Tool requirement
  • Domain concept

• Tool realization
  • Tool layers and components
  • Configuration requirements

• Conclusion
Conclusion

- Supporting of traceability by flexible defining of trace relations
- Services can also be used in the same way while developing to define test cases
- New tools can be added by providing a domain and corresponding tool metamodel
- New services can also be added by defining new queries
Thank you for your attention!

Any questions?