D.3.4 PROOF-OF-CONCEPT CASE TOOL

Gábor Bergmann (BME), Elisa Chiarani (UNITN),Edith Felix (THA), Benjamin Fontan (THA), Charles Haley (OU), Fabio Massacci (UNITN), Zoltán Micskei (BME), Bashar Nuseibeh (OU), Federica Paci (UNITN), Thein Tun (OU) Yijun Yu (OU), Dániel Varró (BME)

Document information

Document Number	D.3.4
Document Title	Proof-of-Concept CASE Tool
Version	1.0
Status	Draft
Work Package	WP 3
Deliverable Type	Prototype
Contractual Date of Delivery	31 January 2011
Actual Date of Delivery	14 January 2011
Responsible Unit	BME
Contributors	OU, UNITN, BME, THA
Keyword List	requirements evolution, argumentation, evolution rules,
Dissemination level	PU

Document change record

Version	Date	Status	Author (Unit)	Description
0.1	15 December 2010	Draft	Federica Paci (UNITN)	First Draft of Demo Scenario
0.2	17 December 2010	Draft	Gábor Bergmann (BME)	Started the executive summary. Elaborated details of each demo script, except for the argumentation Started Technical Overview
0.3	23 December 2010	Draft	Gábor Bergmann (BME)	Completed the executive summary. Improved demo scripts, added argumentation; attached screenshots Resolved issues raised before and during WP3 telephone conference Added Transformation to Technical Overview Added References
0.4	23 December 2010	Draft	Dániel Varró (BME)	Revised the executive summary.
0.5	24 December 2010	Draft	Gábor Bergmann (BME)	Added explanatory Figure to Section 2.
0.6	28 December 2010	Draft	Gábor Bergmann (BME)	Added Figure 2 to Ex. Summary.
0.7	12 January 2011	Draft	Karmel Bekoutou	Quality check completed- minor remarks

			(UNITN)	
0.9	13 January 2011	Revised Draft	Gábor Bergmann (BME)	Updated Scenario descriptions to match reviewer comments.
1.0	14 January 2011	Revised Draft	Gábor Bergmann (BME)	Addressed all minor remarks

Executive summary

This document is the description of the WP3 CASE tool prototype that was built to demonstrate the concepts and workflow of SecMER, the SecureChange Methodology of Evolutionary Requirements.

The demonstrator is an Eclipse-based heterogeneous modelling environment for evolving requirements models that are formulated in different languages appropriate for different work phases and domain expertise. It also provides an analysis toolset to conduct interactive or automated, formal and informal security analysis. Deliverable D.3.4 consists of two components: a software prototype (called SecMER Demonstrator) and this document that presents the research results, the demonstrator architecture and a sample case study.

The main results presented in the demonstrator (and thus in the current document) are the following:

- **Multi-aspect modelling** approach where the security requirements model is composed of several views in different modeling languages, and each work phase can use the facet of the model that is most appropriate to represent their tasks and domain expertise. To cope with the evolving nature of the model, changes made to any of the view models can be **incrementally synchronized** to all other views where appropriate using change-driven transformations.
- **Automated pattern-based analysis** for certain security properties. This analysis is resilient to change and the results are continuously and incrementally kept up-to-date.
- **Interactive and formal argumentation analysis** to support security experts in conducting arguments to verify security properties. Taking up the challenge of evolving models, these argumentation features are complemented by partially automated strategies to cope with changes to the requirements model.

Position of the deliverable in the project timeline

The main artifacts of WP3 are the SeCMER conceptual model, the SeCMER methodology for changing requirements, and a CASE tool prototype that supports the different steps of SeCMER methodology. Considering the SecureChange project timeline depicted in , the SecMER conceptual model and the SeCMER methodology have been conceived during the M0-M24 timeframe, while the CASE tool is now under development during the M12-M36 timeframe. The prototype tool presented in this deliverable thus belongs to the timeframe M12-M24.

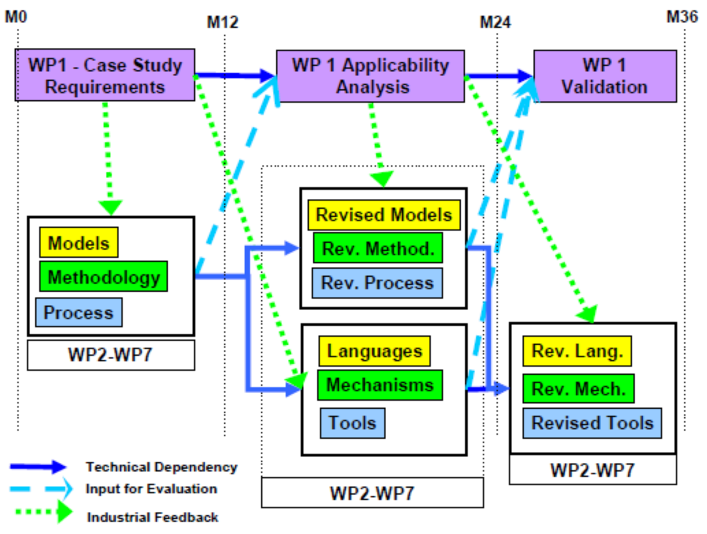


Figure 1. SecureChange Timeline

Integration

This document currently provides an overview of the Year 2 results of WP3 in developing a prototype tool. At its current stage, the tool is a proof-of-concept prototype, which integrates results in requirements engineering developed by different WP3 academic partners using state-of-the-art modeling and transformation techniques. In Year 3, the prototype will demonstrate the feasibility of integration with (1) industrial requirements engineering frameworks and (2) prototype tools of other SecureChange WPs. See Figure 2 for a quick overview of the components involved (provided by multiple project partners) and the roadmap for establishing links.

For demonstration purposes, this deliverable contains an instantiation of the SecMER conceptual model and the methodology based on the ATM case study.

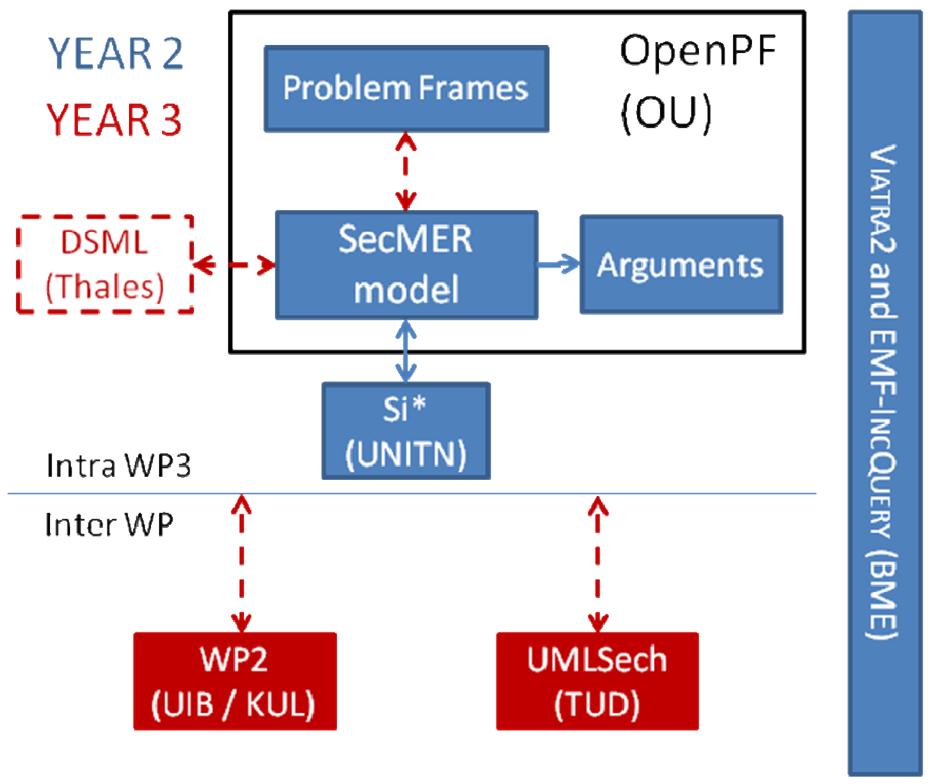


Figure 2. A roadmap for component integration

Validation

During year 3 of the project, WP3 will validate both artifacts and also the tool; the planned method of validation is the tentative application of the tool to the ATM Case Study for the collection, modeling and analysis of the evolution of security requirements in a real industrial context and a final an evaluation workshop with ATM experts.

Table of Contents

List of Figures

1 Demo Description

We are going to illustrate the features supported by WP3 CASE tool prototype using the Process Level Change Requirement of the Air Traffic Management (ATM) case study. The ATM case study will be used during the whole demo since it features requirements-based early security analysis and was used in the previous deliverables of WP3.

1.1 Demo Scenario

The Process Level Change is about the introduction of the Arrival Manager (AMAN), which is an aircraft arrival sequencing tool helping to manage and better organize the air traffic flow in the approach phase. The introduction of the AMAN requires new operational procedures and functions (as described in Deliverable D1.1) that are supported by a new information management system for the whole ATM, an IP based data transport network called System Wide Information Management (SWIM) that will replace the current point to point communication systems with a ground/ground data sharing network which connects all the principal actors involved in the Airports Management and the Area Control Centers.

The entities involved in the simple scenario used for this demo are the AMAN, the Meteo Data Center (MDC), the SWIM-Box and the SWIM-Network. The SWIM-Box is the core of the SWIM information management system which provides access via defined services data that belong to different domain such as flight, surveillance, meteo, etc.

The introduction of the SWIM requires suitable security properties to be satisfied which prevent from corruption, accidental or intentional loss of data and guarantee the integrity and confidentiality of the aircraft sensible data against malicious attacks or intrusions. In the demo, we will focus on information access (access control) and information protection (e.g. integrity) properties on the requirements level. In particular, we will show how to achieve information access by enforcing access control policies on Meteo Data (MD) transmission and how to ensure integrity of FDD data by using digital signature or a trusted communication path.

As exploring design alternatives is inherent to the methodology, we will also show how a cheaper alternative is investigated after the default option of the planned change. One of the elements in the model will be modified to represent the second alternative, demonstrating how the analysis is adapted to the new situation. Note that even though technically the model is modified for a second time, the whole demo revolves around a single evolution of the system.

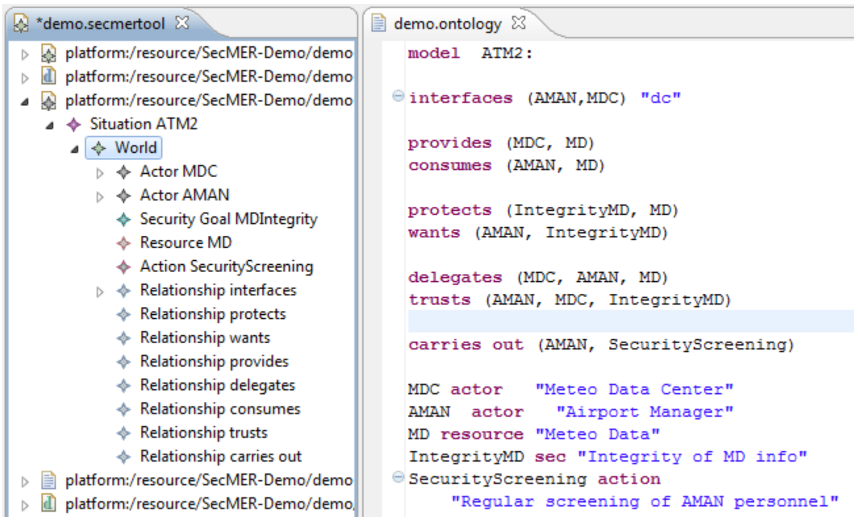
1.2 Overview of demonstrated use cases

The audience of the demo will be able to see the following scenario steps, each of which is supported by the tool: The steps of the demo were chosen such that they follow a typical requirements evolution workflow, also featuring the contributions of WP3.

1) **Synchronization between models in different requirements formalisms.** We will illustrate e.g how the addition of a new actor “SWIM network” in the SI* model for the process level change is reflected on the SecMER model.
2) **Detection of a violation of information protection (integrity in particular) and automatic corrective action based on evolution rules.** We will show how the integrity of Meteo Data is violated in the post-state SecMER model because there is a security goal “Integrity of Meteo Data” protecting the asset Meteo Data, and the asset is delegated to SWIM actors that are initially trusted (transitively) with neither the asset nor the goal. This step requires an evolution rule where the event and condition represent this pattern and where the action part suggests a corrective action, such as adding to the model the missing trust relationship, or creating a digital signature action that explicitly fulfills the goal.
3) **Argumentation for the information access control property**. We will show how argumentation analysis [5] can be carried out for the access control property applied to the Flight Data.
4) **Revising argumentation after the effects of change.** We will show how the revision of the previously existing argument can be automatically triggered when a change is inflicted on its ground facts. We also show how the argument is revised to adapt to the changed situation. This feature is not demonstrated by a subsequent evolution of the system, but by switching the model to a different design alternative.

1.3 Synchronization between models in different requirements formalisms

According to SecMER, a single requirement model can be composed of views in multiple modeling languages, based on the expertise of requirements engineers and the domain-specific style of modeling. The challenge lies in preserving consistency while correctly mapping modifications in a (source) modeling formalism to incremental changes in another (target) modeling language, especially on-the-fly as the modifications are being made. The following is a demonstration of incremental synchronization between views of the requirement model in different formalisms. The concept is demonstrated by the transformation between Si* and the SecMER conceptual model in the ATM case study.



*demo.secmertool

- platform:/resource/SecMER-Demo/demo
- platform:/resource/SecMER-Demo/demo
- platform:/resource/SecMER-Demo/demo
  - Situation ATM2
    - World
      - Actor MDC
      - Actor AMAN
      - Security Goal MDIntegrity
      - Resource MD
      - Action SecurityScreening
      - Relationship interfaces
      - Relationship protects
      - Relationship wants
      - Relationship provides
      - Relationship delegates
      - Relationship consumes
      - Relationship trusts
      - Relationship carries out
- platform:/resource/SecMER-Demo/demo
- platform:/resource/SecMER-Demo/demo

demo.ontology

```
model  ATM2:

interfaces (AMAN,MDC) "dc"

provides (MDC, MD)
consumes (AMAN, MD)

protects (IntegrityMD, MD)
wants (AMAN, IntegrityMD)

delegates (MDC, AMAN, MD)
trusts (AMAN, MDC, IntegrityMD)

carries out (AMAN, SecurityScreening)

MDC actor   "Meteo Data Center"
AMAN  actor   "Airport Manager"
MD resource "Meteo Data"
IntegrityMD sec "Integrity of MD info"
SecurityScreening action
    "Regular screening of AMAN personnel"
```

Figure 3. Evolution pre-state in abstract SecMER model and concrete syntax

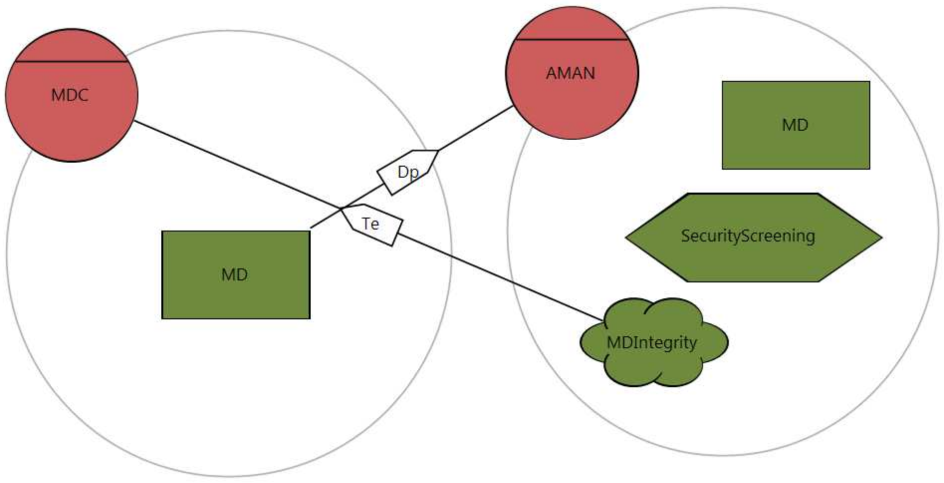


Figure 4. Evolution pre-state on Si* diagram

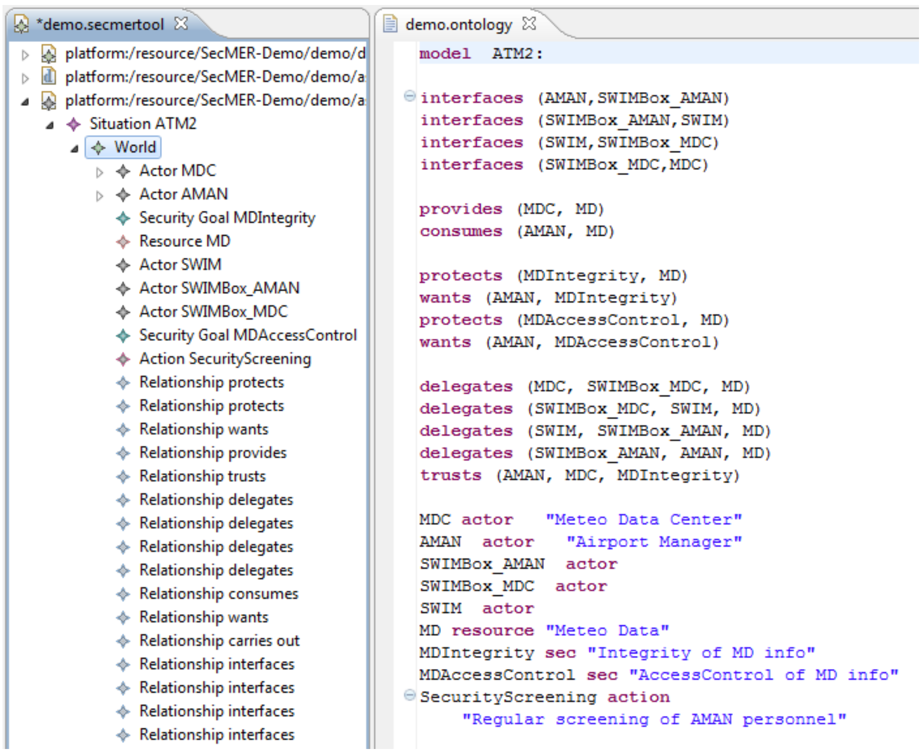
Initially, the two views in the model are consistent, reflecting the same model (the pre-state). See Figure 3 for the SecMER model (intentionally simplified for the sake of demonstration) and Figure 4 for its Si* translation. In this state, the two actors are AMAN and MDC interfacing over a direct connection. MDC provides the asset Meteo Data (MD) which is given directly to AMAN. AMAN has a security goal requiring the integrity of MD, and MDC is trusted to comply with this security requirement. AMAN also performs an Action, SecurityScreening, to regularly conduct a background check on its employees to ensure that they do not pose a risk of internal compromise.

Then the change occurs, and following editing actions are performed:

- The new Actor “SWIM” is created in Si*. A corresponding Actor immediately appears in the SecMER model.

- Likewise, “SWIMBox_MDC” and “SWIMBox_AMAN” are created in the SecMER model, as communication intermediaries. They are immediately propagated to the Si* model as well.
- Also in the SecMER model, “Interfaces” relationships are established between the new Actors and AMAN and MDC. As these connections cannot be represented in Si*, no synchronization action is performed.
- As the data asset is no longer communicated directly between MDC and AMAN, the existing direct delegation of MDC can be removed e.g. in Si*. The corresponding delegation relationship will also disappear from the SecMER model.
- To represent that the data assets are communicated between the actors over the new interfaces, delegation relationships are established between the Actors and dependums in the Si* model. These relationships are then transformed to their SecMER counterparts.
- As the SWIM network can be accessed by multiple parties, the AMAN has the new security goal MDAccessControl (“Access Control of MD info”).

See Figure 5 and Figure 6 for the post-state of the model.



```
platform:/resource/SecMER-Demo/demo/d
platform:/resource/SecMER-Demo/demo/a
platform:/resource/SecMER-Demo/demo/a
  Situation ATM2
    World
      Actor MDC
      Actor AMAN
      Security Goal MDIntegrity
      Resource MD
      Actor SWIM
      Actor SWIMBox_AMAN
      Actor SWIMBox_MDC
      Security Goal MDAccessControl
      Action SecurityScreening
      Relationship protects
      Relationship protects
      Relationship wants
      Relationship provides
      Relationship trusts
      Relationship delegates
      Relationship delegates
      Relationship delegates
      Relationship delegates
      Relationship consumes
      Relationship wants
      Relationship carries out
      Relationship interfaces
      Relationship interfaces
      Relationship interfaces
      Relationship interfaces
```

```
model  ATM2:

interfaces (AMAN,SWIMBox_AMAN)
interfaces (SWIMBox_AMAN,SWIM)
interfaces (SWIM,SWIMBox_MDC)
interfaces (SWIMBox_MDC,MDC)

provides (MDC, MD)
consumes (AMAN, MD)

protects (MDIntegrity, MD)
wants (AMAN, MDIntegrity)
protects (MDAccessControl, MD)
wants (AMAN, MDAccessControl)

delegates (MDC, SWIMBox_MDC, MD)
delegates (SWIMBox_MDC, SWIM, MD)
delegates (SWIM, SWIMBox_AMAN, MD)
delegates (SWIMBox_AMAN, AMAN, MD)
trusts (AMAN, MDC, MDIntegrity)

MDC actor   "Meteo Data Center"
AMAN  actor   "Airport Manager"
SWIMBox_AMAN  actor
SWIMBox_MDC  actor
SWIM  actor
MD resource "Meteo Data"
MDIntegrity sec "Integrity of MD info"
MDAccessControl sec "AccessControl of MD info"
SecurityScreening action
    "Regular screening of AMAN personnel"
```

Figure 5. Post-state in concrete syntax and abstract SecMER model

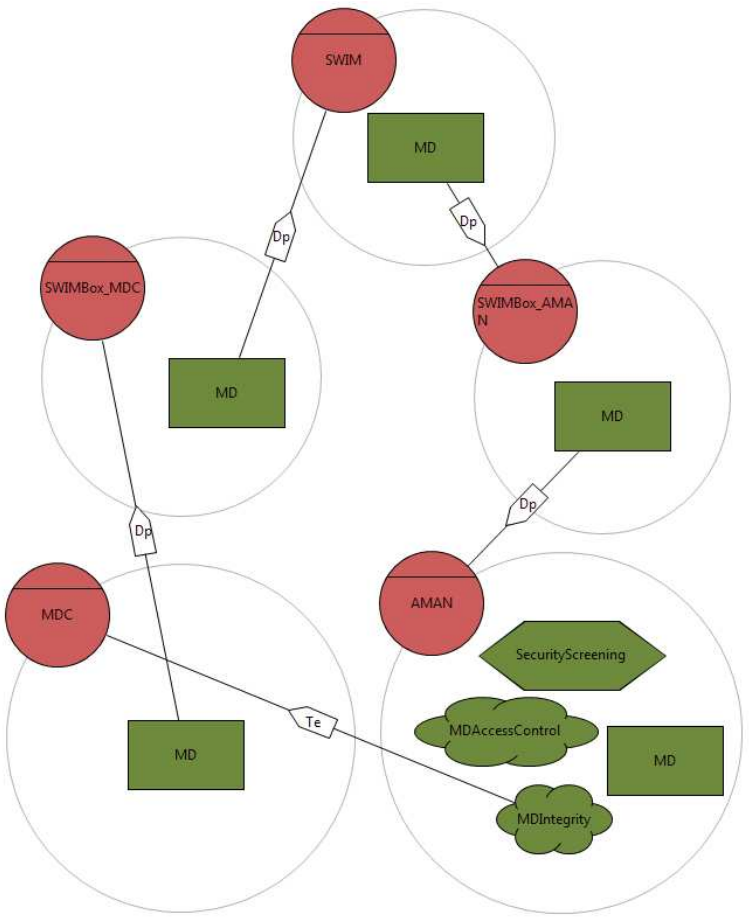


Figure 6. Evolution post-state on Si* diagram

1.4 Detection of a violation of information protection and automatic corrective action based on evolution rules

Some security properties can be effectively evaluated or approximated based on the requirements model in a completely automated fashion. It is also possible to suggest default solutions. SecMER includes the language of Evolution Rules, providing a declarative guard that can detect undesired situations and the possibility to include actions.

The following is a demonstration of on-the-fly evaluation of security properties and offering automated corrections using Evolution Rules.

Actor MDC provides Meteo Data and wants to communicate it to Actor AMAN with the integrity preserved. Before the evolution, MDC delegated the asset directly to AMAN, and trusted AMAN with the asset.

- In the post-state, the asset is delegated first to SwimBox_MDC, then to SWIM network, then to SwimBox_AMAN, and finally to AMAN. Although MDC is trusted by AMAN with the integrity security goal, the intermediate actors are not,

not even transitively. The tool marks all three intermediate actors as security violations (see Figure 7).

- Automatic reactions (quick fixes) are also provided for this security pattern. Candidate solutions (see Figure 8) are automatically offered for each of the violating actors: (a) to add a trust relationship from MDC to the actor in question, over the asset as dependum (b) use the goal protecting the asset as dependum instead, or (c) add an action (i.e. digital signature) that explicitly fulfils the goal.
- Assuming that it is in fact true that MDC trusts SWIM Network over the security goal, so this quick fix can be accepted.
- However, MDC does not have trust in the SWIM-Boxes, as it is not directly concerned with them. Fortunately though, the SWIM Network trusts its Boxes over the goal, so these trusts relationships can also be added. In the end, the SWIM-Boxes are also transitively trusted by MDC, so the security property will cease to be violated.
- Alternatively, an Action such as "MD is digitally signed" can be created to protect the integrity of MD even when handled by untrusted actors.

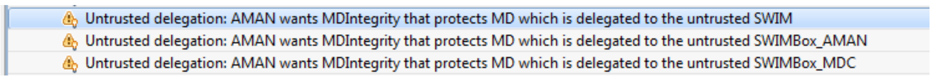
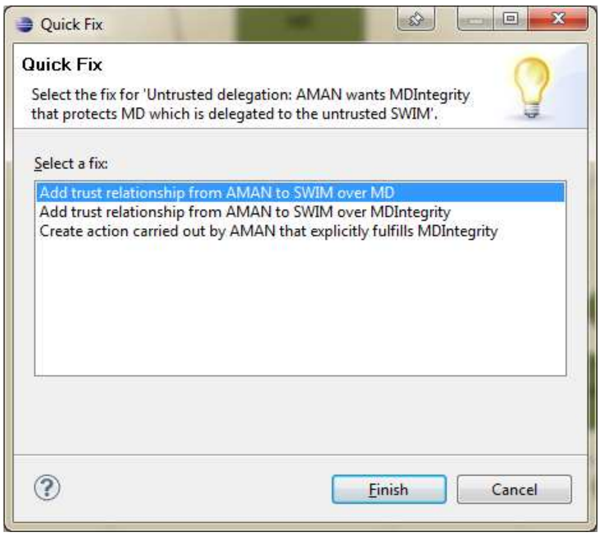

Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIM
Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIMBox_AMAN
Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIMBox_MDC

Figure 7. Detected security issues


Quick Fix

Quick Fix

Select the fix for 'Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIM'.

Select a fix:

Add trust relationship from AMAN to SWIM over MD
Add trust relationship from AMAN to SWIM over MDIntegrity
Create action carried out by AMAN that explicitly fulfills MDIntegrity

Figure 8. Automatic solutions suggested by evolution rules

1.5 Argumentation for the information access property

The scenario fragment we are going to consider is transmission of MD data to the AMAN via the new communication network, focusing on how to enforce access control policies on MD. Arguments will be conducted for the security goal of protecting MD from malicious attack.

- Based on the requirements model, the facts that can be used for argumentation are initialized with tool support.
- Argumentation experts build the argument to support the claim that the system is secure. Counter-arguments (rebuttals) can be voiced, as well as mitigations for rebuttals, ad infinitum.
- Assuming that an initial argument existed in the pre-state, the effects of changes can also be represented as Round 2 rebuttals to facts.

The resulting Argumentation model is visualized in Figure 9. The diagram says that the AMAN system is claimed to be secure before the change (Round #1), and the claim is warranted by be the facts the system is known to be a close system (F1), and the physical location of the system is protected (F2). This argument is rebutted in Round #2, in which another argument claims that the system is no longer secure because SWIM will not keep AMAN closed. The rebuttal argument is mitigated in Round #3 by three arguments, which suggest that the AMAN may still be secure given that the physical infrastructure is secure, personnel are trustworthy and access to data is controlled.

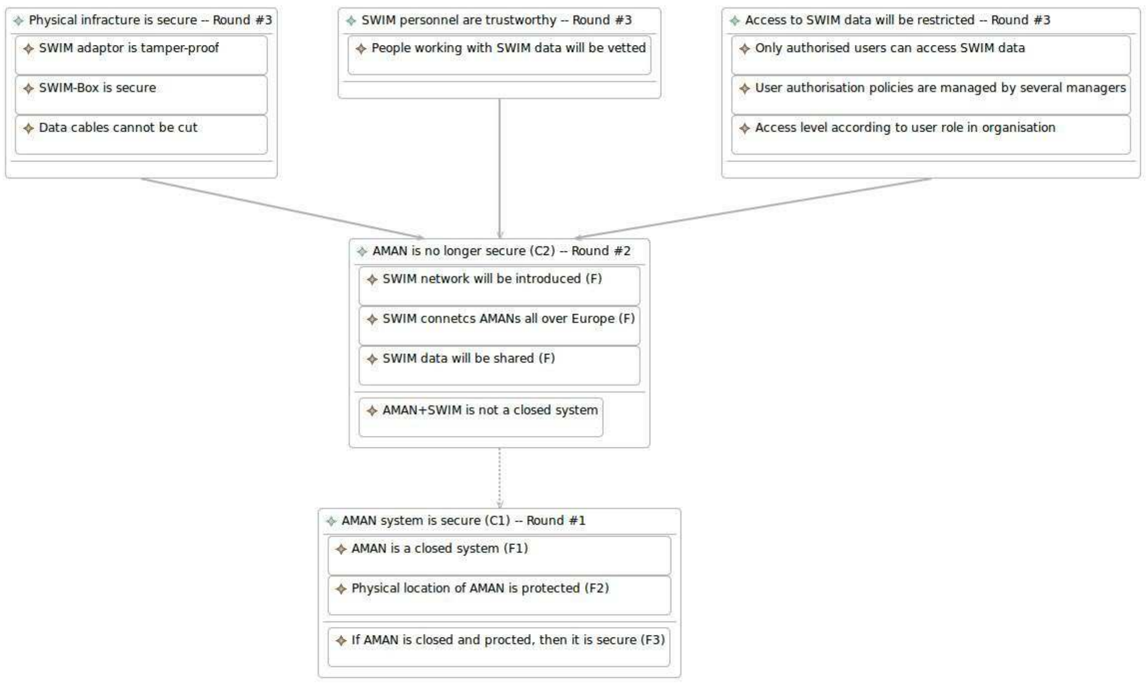

Physical infracture is secure -- Round #3
SWIM adaptor is tamper-proof
SWIM-Box is secure
Data cables cannot be cut
SWIM personnel are trustworthy -- Round #3
People working with SWIM data will be vetted
Access to SWIM data will be restricted -- Round #3
Only authorised users can access SWIM data
User authorisation policies are managed by several managers
Access level according to user role in organisation
AMAN is no longer secure (C2) -- Round #2
SWIM network will be introduced (F)
SWIM connetcs AMANs all over Europe (F)
SWIM data will be shared (F)
AMAN+SWIM is not a closed system
AMAN system is secure (C1) -- Round #1
AMAN is a closed system (F1)
Physical location of AMAN is protected (F2)
If AMAN is closed and procted, then it is secure (F3)

Figure 9. A fragment of an argument model

1.6 Revising argumentation after the effects of change

While argumentation is a powerful framework for early-stage analysis of security properties based on the requirements model, by default it considers a single state of the model. The challenge is in detecting arguments that have potentially been invalidated by changes, and revisiting these arguments to reflect the evolution, while no costly revision process is required for unaffected arguments. The following is a demonstration of the effect of model evolution on arguments.

In the default option of the evolution that is currently reflected by the model, AMAN carries out the SecurityScreening Action. This action means e.g. a regular background check whether personnel who have physical or virtual access to SWIM infrastructure can be suspected to compromise operation or otherwise pose a security threat. This action was used as a fact in the argument for the information access property. A second alternative is also proposed to cut costs for SWIM, where the screening is not performed. After the first alternative has been fleshed out, the second option will also be investigated.

- Switching to the second design alternative removes this Action from the SecMER model. As a consequence, the argument that uses this Action as a fact has now become outdated, and the claim of the argument might have lost its validity. The argument is automatically marked as invalid, which can be used as an alert to notify the argumentation experts.
- The change can be represented within the existing argument as a new round initiated by the rebuttal of the obsolete fact. The argumentation experts can then convene and argue about whether the claim still holds.

2 Tool Realization

In this Section we give an overview of the design choices behind the actual prototype tool and the integration of the academic tools.

2.1 Architectural Overview

A prototype tool is realized as a set of Eclipse plug-ins written in Java (partly generated), and models are represented in the Eclipse Modeling Framework (EMF [1]). The components of the tool currently fall into these categories:

- Eclipse plug-ins of OpenPF (requirements engineering tool by OU [4]), including (a) the implementation of the SecMER conceptual model, (b) the argumentation model and tools, as well as (c) the modeling tools for Problem Frames (only limited synchronization is supported with the other formalisms as of now)
- Si* (requirements engineering tool [3] by UNITN),
- traceability models to represent the relationship between corresponding model elements in different languages, e.g. the SecMER conceptual model and Si*,
- run-time platform components of EMF-IncQuery (incremental EMF model query engine by BME [2]) for change-driven transformations,
- model query plug-ins automatically generated from transformation specification and Evolution Rules by the development-time tools of EMF-IncQuery and VIATRA2 (model transformation framework by BME),
- integration code developed solely for this tool, including User Interface commands and the Java definition of the action parts of Evolution Rules.

The relationship of the most important model management components are depicted on Figure 10, focusing on the Si* and SecMER models in particular, as well as the traceability model established between them. User Interface components are omitted from this diagram. See also Figure 2 for a higher-level overview of what components and modeling formalisms are involved.

All the involved EMF models are accessed through a common EMF ResourceSet and edited solely through the corresponding TransactionalEditingDomain (from the EMF Transaction API). Consequently, all modifications are wrapped into EMF Transactions, including those carried out by manual editing through the User Interface (e.g. the Si* diagram editor or the generic EMF tree editor) as well as changes performed by automated mechanisms such as model transformation. As one of the benefits, concurrent modifications are serialized and therefore conflict-free. Furthermore, the commit process of the transactions provides an opportunity for triggering change-driven actions.

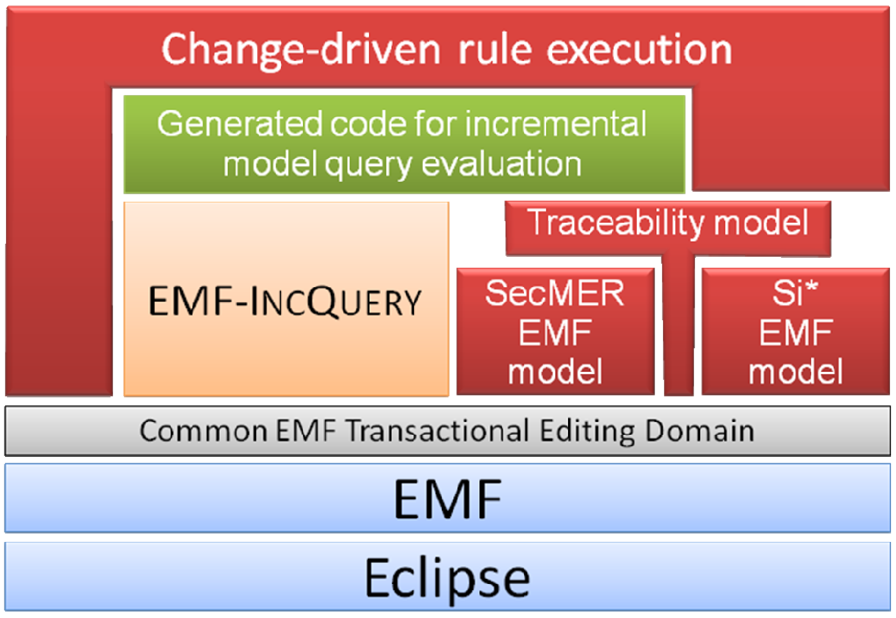

Change-driven rule execution
Generated code for incremental model query evaluation
Traceability model
EMF-INCQUERY
SecMER EMF model
Si* EMF model
Common EMF Transactional Editing Domain
EMF
Eclipse

Figure 10. Architectural overview of mode management components

2.2 Trigger mechanism

The incremental query mechanism provided by EMF-IncQuery plays a key role in the functionality of the tool. Incremental query evaluation code is generated automatically at development time by EMF-IncQuery, from a graph pattern-based declarative description of EMF model queries. Through this incremental evaluation functionality, Evolution Rules can be efficiently triggered by changes captured as graph patterns. The implementation currently supports detecting the presence, appearance and disappearance, but not the entire Graph Change Patterns formalism (see D.3.2).

The core triggering plug-in offers an Eclipse extension point for defining change-driven rules. Multiple constituent plug-ins contribute extensions to active their respective set of rules. The graph pattern-based declarative event/condition feature of the rules is evaluated by the incremental graph pattern matcher plug-ins automatically generated from the declarative description by EMF-IncQuery. At the commit phase of each transaction, the rules that are found to be triggered will be executed to provide their reactions to the preceding changes. These reactions are implemented by arbitrary Java code, and they are allowed to modify the model as well (wrapped in nested transactions) and could therefore be reacted upon.

Currently, there are three groups of change-driven rules contributed to the extension point:

- transformation rules that realize the on-the-fly synchronization between multiple modeling formalisms,

- security-specific evolution rules that detect the appearance of undesired security patterns, provide alerts on these problems and optionally offer candidate solutions.
- rules for marking arguments as invalid when changes are inflicted on their ground facts.

2.3 Transformation specification

The tool maintains a synchronizing transformation between Si* and the SecMER model. The challenge is to provide bi-directional synchronization with changes propagated on the fly. Naturally the two languages have different expressive power, therefore

(a) some concepts are not mapped from one formalism to the other or vice versa,

(b) some model elements may be mapped into multiple (even an unbounded amount of) corresponding model elements in the other formalism, and finally

(c) it is possible that a single model element has multiple possible translations (due to the source formalism being more abstract); one of them is created as a default choice, but the other options are also accepted.

The following mappings define the transformation:

- There is a many-to-one correspondence between Si* Actors of the same name and SecMER Actors.
- There is a many-to-one correspondence between Si* Resources with the same name and SecMER Resources that are provided (thus eligible to be represented in Si*).
- There is a one-to-one correspondence between the original copy of a Si* Resource, owned by an Actor and not received through delegation, and the SecMER Provides Relationship from the corresponding Actor to the corresponding Resource.
- Additional copies of a Si* Resource, owned by an Actor and received by delegation but not delegated further, are mapped into SecMER Consumes Relationships from the corresponding Actor to the corresponding Resource.
- There is a many-to-one correspondence between Si* Tasks with the same name and SecMER Actions that are carried out (thus eligible to be represented in Si*).
- There is a one-to-one correspondence between a copy of a Si* Task owned by an Actor and not delegated further, and the SecMER Carries Out Relationship between the corresponding Actor and Action.
- There is a many-to-one correspondence between Si* Softgoals having the same name and SecMER Security Goals that are wanted (thus eligible to be represented in Si*).
- There is a many-to-one correspondence between Si* Goals of the same name, and SecMER Goals that are wanted (thus eligible to be represented in Si*).and

are *not* Security Goals. A newly created Si* Goal is mapped *by default* into a SecMER Goal of type Goal proper (not any of its subtypes).

- There is a one-to-one correspondence between the orginial copy of a Si* Goal or Softgoal owned by an Actor and not received by delegation, and the SecMER Wants Relationship between the corresponding Actor and Goal.
- There is a one-to-one correspondence between Si* 'AND' Compositions and SecMER And Decompositions between Actions or Goals, if both endpoints are mapped.
- There is a one-to-one correspondence between Si* 'OR' Compositions and SecMER Or Decompositions between Actions or Goals, if both endpoints are mapped.
- There is a one-to-one correspondence between a Si* MeansEnd Relation from a Task to a Goal or Softgoal and the SecMER Fulfils Relationship between the corresponding Action and Goal, if both endpoints are mapped.
- There is a one-to-one correspondence between a Si* Delegation of Permission Relation, pointing from a Resource owned by an Actor to a second Actor, and the SecMER Delegates Relationship with the first Actor as source, the second Actor as target and the Resource as dependum, if all three endpoints are mapped.
- There is a one-to-one correspondence between a Si* Delegation of Execution Relation, pointing from a Task or Goal or Softgoal owned by an Actor to a second Actor, and the SecMER Delegates Relationship with the first Actor as source, the second Actor as target and the Action or Goal as dependum, if all three endpoints are mapped.
- There is a one-to-one correspondence between a Si* Trust of Permission Relation, pointing from a Resource owned by an Actor to a second Actor, and the SecMER Trusts Relationship with the first Actor as source, the second Actor as target and the Resource as dependum, if all three endpoints are mapped.
- There is a one-to-one correspondence between a Si* Trust of Execution Relation, pointing from a Task or Goal or Softgoal owned by an Actor to a second Actor, and the SecMER Trusts Relationship with the first Actor as source, the second Actor as target and the Action or Goal as dependum, if all three endpoints are mapped.

References

[1] The Eclipse Project: *Eclipse Modeling Framework*. http://eclipse.org/emf

[2] G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös, "Incremental Evaluation of Model Queries over EMF Models", *Model Driven Engineering Languages and Systems, 13th International Conference, MODELS'10*: Springer, 10/2010.

[3] F. Massacci, J. Mylopoulos, and N. Zannone, *Computer-aided Support for Secure Tropos.* Automated Software Engineering, 2007. **14**(3): p. 341-364.

[4] T. Tun, Y. Yu, R. Laney, and B. Nuseibeh, "Early Identification of Problem Interactions: A Tool-Supported Approach," in *Requirements Engineering: Foundation for Software Quality*, vol. 5512, Springer Berlin / Heidelberg, 2009, pp. 74-88.

[5] T. Tun, Y. Yu, C. Haley, B. Nuseibeh, "Model-Based Argument Analysis for Evolving Security Requirements," Secure System Integration and Reliability Improvement, pp. 88-97, 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, 2010