A Tool for Managing Evolving Security Requirements*

Gábor Bergmann[1], Fabio Massacci[2], Federica Paci[2],
Thein Tun[3], Dániel Varró[1], and Yijun Yu[3]

[1] DMIS - Budapest University of Technology and Economics,
{bergmann,varro}@mit.bme.hu
[2] DISI - University of Trento,
{fabio.massacci,federica.paci}@unitn.it
[3] DC - The Open University
{t.t.tun,y.yu}@open.ac.uk

Abstract. Requirements evolution management is a daunting process. Requirements change continuously making the traceability of requirements hard and the monitoring of requirements unreliable. Moreover, changing requirements might have an impact on the security properties a system design should satisfy: certain security properties that are satisfied before evolution might no longer be valid or new security properties need to be satisfied. This paper presents SeCMER, a tool for requirements evolution management developed in the context of the SecureChange project. The tool supports automatic detection of requirement changes and violation of security properties using change-driven transformations. The tool also supports argumentation analysis to check security properties are preserved by evolution and to identify new security properties that should be taken into account.

Keywords: security requirements engineering, secure i*, security argumentation, change impact analysis, security patterns

1 Introduction

Modern software systems are increasingly complex and the environment where they operate is increasingly dynamic. The number and needs of stakeholders is also changing constantly as they need to adjust to the changing environment. A consequence of this trend is that the requirements for a software system increases and changes continually. To deal with evolution, we need analysis techniques that assess the impact of system evolution on the satisfaction of requirements such as security of the system which is very sensitive to evolution: security properties satisfied before evolution might no longer hold or new security properties need to be satisfied as result of the evolution.

Another important aspect is the change management process itself which is a major problem in practice. Changes make the traceability of requirements

* Work parly supported by the project EU-FP7-ICT-FET-IP-SecureChange.

hard and the monitoring of requirements unreliable: requirements management is difficult, time-consuming and error-prone when done manually. Thus, a semi-automated requirements evolution management environment, supported by a tool, will improve requirement management with respect to keeping requirements traceability consistent, realizing reliable requirements monitoring, improving the quality of the documentation, and reducing the manual effort.

In this paper we present SeCMER[4], a tool developed in the context of the SecureChange European project[5]. The tool supports the different steps of SeCMER methodology for evolutionary requirements [4]. The methodology allows to model requirement evolution in different state of the art requirement languages such as SI* [6], Problem Frames (PF) [9] and SeCMER that is a requirement language that includes concepts belonging to SI*, PF and security such as asset. The methodology also supports the automatic detection of requirement changes and violation of security properties and argumentation analysis [9] to check security properties are preserved by evolution and to identify new security properties that should be taken into account. Change driven transformations based on *evolution rules* [3] are leveraged to check argument validity, to automatically detect violations or fulfilment of security properties, and to issue alerts prompting human intervention, a manual analysis or argumentation process, or trigger automated reactions in certain cases.

In the next section (§2) we describe the tool architecture, then we illustrate the tool features based on an industrial example of evolution taken from the air traffic management domain (§3) and finally conclude the paper (§4).

2 SeCMER Tool architecture

SeCMER is an Eclipse-based heterogeneous modeling environment for managing evolving requirements models. It has the following features (See also Fig. 1):

- **Modeling of Evolving Requirements**. Requirement models can be drawn in SI*, Problem Frames or SeCMER. Traceability and bidirectional synchronization is supported between SeCMER and SI* requirements models.
- **Change detection based on evolution rules**. Violations of formally defined static security properties expressed as patterns can be automatically identified. Detection of formal or informal arguments that has been invalidated by changes affecting model elements that contributed to the argument as evidence is also supported.
- **Argumentation-based security analysis**. Reasoning about security properties satisfaction and identification of new security properties is supported.

These capabilities of the tool are provided by means of the integration of a set of EMF-based [7] Eclipse plug-ins written in Java, relying on standard EMF technologies such as GMF, Xtext and EMF Transaction.

[4] A detailed description of the tool implementation is reported in [5]

[5] www.securechange.eu

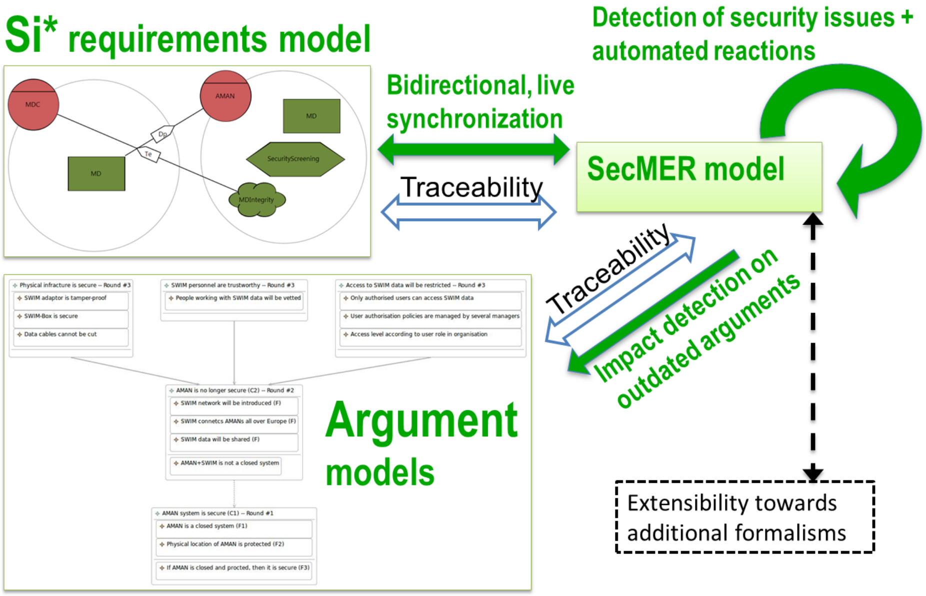

Si* requirements model
MDC
AMAN
MD
Dp
Te
MD
SecurityScreening
MDIntegrity
Bidirectional, live synchronization
Traceability
Detection of security issues + automated reactions
SecMER model
Traceability
Impact detection on outdated arguments
Extensibility towards additional formalisms
Physical infracture 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)
Argument models

Fig. 1. Models and features in the SeCMER tool

SeCMER integrates Si* [6] as a graphical modeling framework for security requirements, with OpenPF [8] that supports formal and informal manual argumentation of security properties. Change detection for security patterns and evolution rules, as well as the detection of invalidated arguments are performed using EMF-IncQuery [2].

The core trigger engine plug-in offers an Eclipse extension point for defining change-driven rules. Multiple constituent plug-ins contribute extensions to register their respective set of rules. The graph pattern-based declarative event/condition feature of the rules is evaluated efficiently (see measurements in [2]) by the incremental graph pattern matcher plug-ins automatically generated from the declarative description by EMF-IncQuery. At the commit phase of each EMF 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.

So far, there are three groups of change-driven rules as extension points:

- 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, raise alerts and optionally offer candidate solutions.
- rules for invalidating arguments when their ground facts change.

A major feature is the a bi-directional synchronizing transformation between Si* and the SeCMER model with changes propagated on the fly, interactively. Since the languages have different expressive power, the following challenges arise:

1. some concepts are not mapped from one formalism to the other or vice versa,
2. some model elements may be mapped into multiple (even an unbounded amount of) corresponding model elements in the other formalism, and finally
3. 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 it can later be changed to the other options, which are also tolerated by the transformation system.

3 Demo Scenario

We are going to illustrate the features supported by our prototype using the ongoing evolution of ATM systems as planned by the ATM 2000+ Strategic Agenda [1] and the SESAR Initiative.

Part of ATM system's evolution process is the introduction of the Arrival Manager (AMAN), which is an aircraft arrival sequencing tool to help manage and better organize the air traffic flow in the approach phase. The introduction of the AMAN requires new operational procedures and functions 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 to data that belong to different domain such as flight, surveillance, meteo, etc. The introduction of the SWIM requires suitable security properties to be satisfied: we will show how to protect information access on meteo data and how to ensure integrity of meteo data.

1. **Requirements evolution**. We show how SeCMER supports the representation of the evolution of the requirement model as effect of the introduction of the SWIM.
2. **Change detection based on evolution rules.**
   a *Detection of a security property violation based on security patterns.* We show how the tool detects that the integrity security property of the resource MD "Meteo Data" is violated due to the lack of a trusted path.
   b *Automatically providing corrective actions based on evolution rules.* We show how evolution rules may suggest corrective actions for the detected violation of the integrity security property.
3. **Argumentation-based security analysis**. We show how argumentation analysis [9] can be carried to provide evidence that the information access property applied to the meteo data is satisfied after evolution.

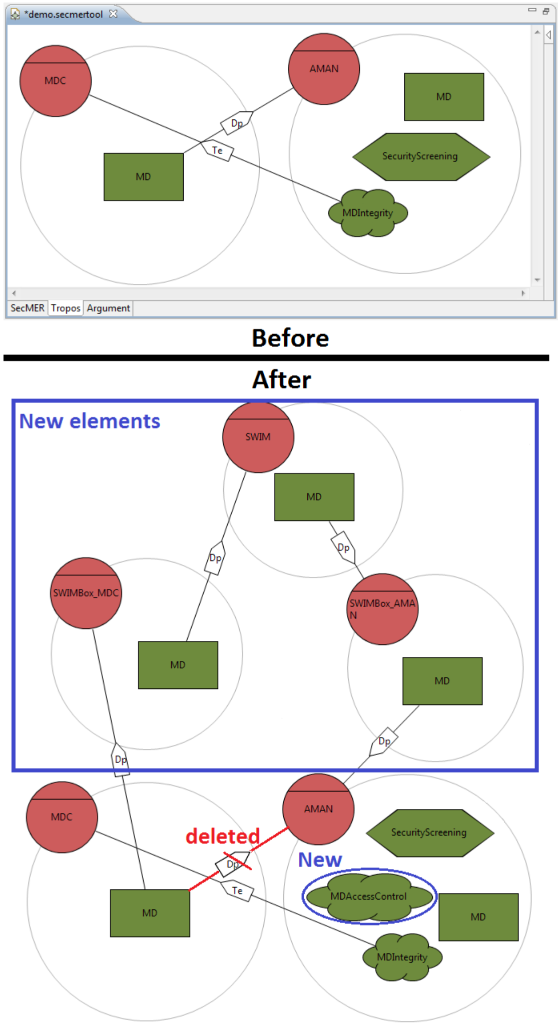


Fig. 2. Annotated screenshot fragments showing requirements evolution

The steps of the demo were chosen such that they follow a typical requirements evolution workflow, also featuring the contributions of WP3.

Requirements evolution. Fig. 2 shows the evolution of the model. The before model includes two actors the *AMAN* and *MDC*: MDC provides the asset Meteo Data (*MD*) to the AMAN. The AMAN has an integrity security goal *MDIntegrity* for MD, and MDC is entrusted with this goal. AMAN also performs an Action, *SecurityScreening*, to regularly conduct a background check on its employees to ensure that they do not expose to risk the information generated by the AMAN.

Listing 1 Pattern to capture violations of the trusted path property

```
shareable pattern
noTrustedPath(ConcernedActor,SecGoal,Asset,UntrustedActor)={
  Actor.wants(ConcernedActor,SecGoal);
  SecurityGoal(SecGoal);
  SecurityGoal.protects(SecGoal, Asset);
  Actor.provides(ProviderActor,Asset);
  find
transitiveDelegation(ProviderActor,UntrustedActor,Asset);
  neg Actor.trust*(ConcernedActor,UntrustedActor);
  neg find
trustedFulfillment(ConcernedActor,AnyActor,AnyTask,SecGoal); }
```

As the communication between the AMAN and MDC is mediated by the SWIM, the before model evolves as follows:

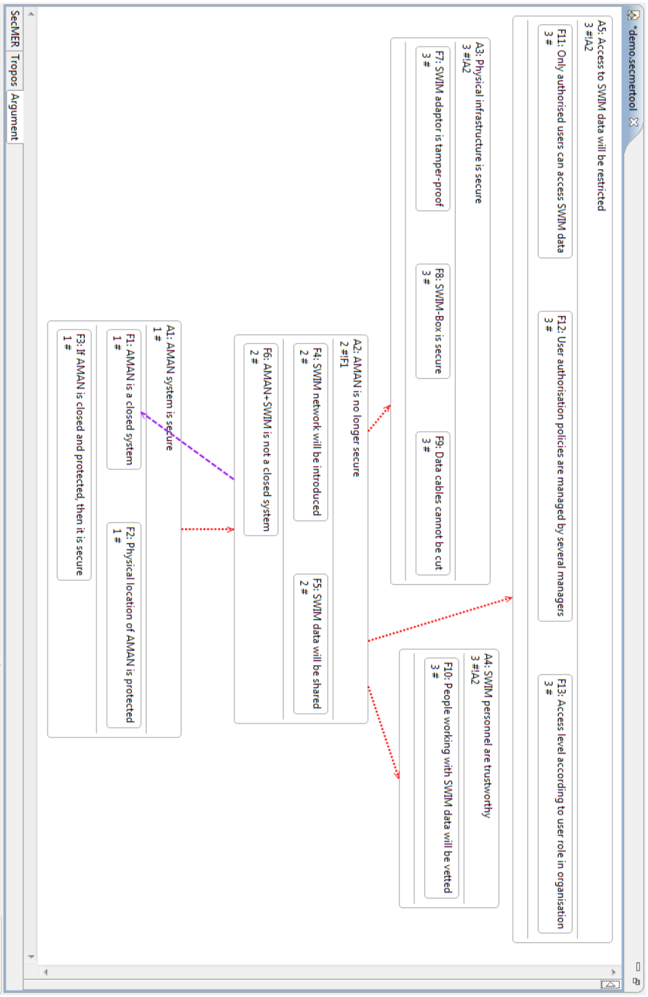
- The Actors *SWIM*, *SWIMBox_MDC* and *SWIMBox_AMAN* are introduced in the SI* model
- As the meteo data is no longer directly provided by MDC to AMAN, the delegation relation between the two is removed.
- Delegation relationships are established between the Actors MDC, SWIMBox_MDC, SWIM, SWIMBox_AMAN, AMAN.
- As the SWIM network can be accessed by multiple parties, the AMAN has a new security goal *MDAccessControl* protecting MD resource.

Detecting violations of security properties based on security patterns. SeCMER includes facilities that allow for the declarative definition of security patterns that express situations that leads to the violation of a security property. For example, if a concerned actor wants a security goal that expresses that a resource must be protected, then each actor that the resource is delegated to must be trusted (possibly transitively) by the concerned actor. An exception is made if a trusted actor performs an action to explicitly fulfill the security goal, e.g. digital signature makes the trusted path unneccessary in case of an integrity goal. See Lst. 1 for the definition of the pattern using the declarative model query language of EMF-IncQuery [2].

According to this pattern the integrity property for MD is violated because AMAN entrusts MDC with the integrity security goal, but the communitation intermediary actors SWIMBox_MDC, and SWIMBox_AMAN are not.

Automatic corrective actions based on evolution rules. The security pattern in Lst. 1 can be used to define evolution rules that define automated corrective actions to be applied to the model in order to re-establish the integrity security property. Possibile examples of corrective actions are:

- Add a trust relationship between MDC and SWIM Network having the integrity security goal as dependum.
- Alternatively, an Action such as MD is digitally signed can be created to protect the integrity of MD even when handled by untrusted actors.



A5: Access to SWIM data will be restricted
3 #A2

F11: Only authorised users can access SWIM data
3 #

F12: User authorisation policies are managed by several managers
3 #

F13: Access level according to user role in organisation
3 #

A3: Physical infrastructure is secure
3 #A2

F7: SWIM adaptor is tamper-proof
3 #

F8: SWIM-Box is secure
3 #

F9: Data cables cannot be cut
3 #

A4: SWIM personnel are trustworthy
3 #A2

F10: People working with SWIM data will be vetted
3 #

A2: AMAN is no longer secure
2 #F1

F4: SWIM network will be introduced
2 #

F5: SWIM data will be shared
2 #

F6: AMAN+SWIM is not a closed system
2 #

A1: AMAN system is secure
1 #

F1: AMAN is a closed system
1 #

F2: Physical location of AMAN is protected
1 #

F3: If AMAN is closed and protected, then it is secure
1 #

Fig. 3. Screenshot fragment showing the argumentation model

Argumentation for the information access property. Fig. 3 shows the different rounds of the argumentation analysis that is carried out for the infomation access security property applied to MD resource.

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.

4 Conclusions

The paper presented SeCMER, a tool for managing evolving requirements. As shown by the ATM-based demo scenario, the tool supports visual modeling of security requirements. Additionally, argument models can be constructed manually to investigate the satisfaction of security properties; the tool detects invalidated arguments if the requirements model evolves. Finally, the tool performs continuous and automatic pattern-based security properties violation detection, with "quick fix" corrective actions specified by evolution rules.

We are planning to extend the tool in order to support other set of security patterns and evolution rules to automate the detection and handling of security violations in a wider range of application scenarios. We will also realize a tighter integration with additional modeling formalisms (Problem Frames) and industrial tools e.g DOORS-TREK. The usability and the features of the tool are going to be evaluated through a study involving ATM-domain experts.

References

1. EUROCONTROL ATM Strategy for the Years 2000+ Executive Summary (2003)
2. Bergmann, G., et al.: Incremental evaluation of model queries over EMF models. In: Model Driven Engineering Languages and Systems, MODELS'10. Springer (2010)
3. Bergmann, G., et al.: Change-Driven Model Transformations. Change (in) the Rule to Rule the Change. Software and System Modeling (2011), to appear.
4. Bergmann et al.: D3.2 Methodology for Evolutionary Requirements, `http://www.securechange.eu/sites/default/files/deliverables/D3.2-%20Methodology%20for%20Evolutionary%20Requirements_v3.pdf`
5. Bergmann et al.: D3.4 Proof of Concept Case Tool, `http://www.securechange.eu/sites/default/files/deliverables/D3.4%20Proof-of-Concept%20CASE%20Tool%20for%20early%20requirements.pdf`
6. Massacci, F., Mylopoulos, J., Zannone, N.: Computer-aided support for secure tropos. Automated Software Engg. 14, 341–364 (September 2007)
7. The Eclipse Project: Eclipse Modeling Framework, `http://www.eclipse.org/emf`
8. Tun, T., et al.: Early identification of problem interactions: A tool-supported approach. In: Glinz, M., Heymans, P. (eds.) Requirements Engineering: Foundation for Software Quality, 15th International Working Conference, pp. 74–88. No. 5512 in Lecture Notes in Computer Science, Springer (2009)
9. Tun, T.T., et al.: Model-based argument analysis for evolving security requirements. In: Proceedings of the 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement. pp. 88–97. SSIRI '10, IEEE Computer Society, Washington, DC, USA (2010)