D4.4b - INTEGRABILITY OF DESIGN MODELLING SOLUTION

Stéphane Paul, Olivier Delande (Thales Research & Technology)

Document information

Document Number	D4.4b
Document Title	Integrability of design modelling solution
Version	1.0
Status	Final
Work Package	WP 4.4
Deliverable Type	Report
Contractual Date of Delivery	31 January 2012
Actual Date of Delivery	23 January 2012
Responsible Unit	Thales Research & Technology
Contributors	TRT, UIB, UNITN
Keyword List	Security engineering, change management
Dissemination level	PU

Document change record

Version	Date	Status	Author (Unit)	Description
0.01	28/10/11	Working	Stéphane Paul	Outline
0.02	31/10/11	Working	Stéphane Paul	Updated outline after comments by Jan Jürjens
0.03	17/11/11	Working	Stéphane Paul	Chapter 2 (posted on BSCW for UIB).
0.04	05/12/11	Working	Stéphane Paul, Olivier Delande, Frank Innerhofer-Oberperfler, Philipp Kalb	Document close to completion.
0.05	16/12/11	Draft	Stéphane Paul, Olivier Delande, Frank Innerhofer-Oberperfler, Philipp Kalb	Submission for consortium review.
0.06	22/12/11	Draft	Michela Angeli, Stéphane Paul	Quality checked (minor formatting remarks).
0.07	23/12/11	Draft	Olivier Delande	Internal quality check.
0.08	23/12/11	Draft	Michela Angeli, Stéphane Paul	Second round of quality check (minor formatting remarks).
0.09	03/01/11	Draft	Stéphane Paul	Additional note on traceability.
0.10	20/01/11	Draft	Federica Paci, Jan Jürjens	Scientific and quality review comments.
1.0	23/01/11	Final	Stéphane Paul	Final version

Executive summary

How to design large long-lived evolving software-intensive systems, with stringent security concerns, in an industrial context, with rigid existing system engineering processes and widely deployed workbenches?

On a methodological standpoint, the proposed approach is to orchestrate (as opposed to integrate) security and system engineering concerns by two types of relations between engineering processes: (i) vertical relations between successive security-related processes; and (ii) horizontal relations between mainstream system engineering processes and concurrent security-related processes. This approach can be applied to the complete system / software lifecycle, from early security requirement elicitation to runtime configuration and monitoring, via high-level architecting, detailed design, development, integration and design-time testing.

In this deliverable we illustrate the methodological principles of the approach, with a strong focus on change.

On a tooling standpoint, robust tool integration has by far our preference. However, the integration must be both tight and loose. By loose integration we mean that each tool should be useable independently from the complete system engineering workbench (if so needed). By tight integration, we mean that the multiple models created by the different tools of the different system / software stakeholders should be kept consistent at all times, and it should not require overwhelming efforts to maintain this consistency.

In this deliverable, such a tight and loose integration is proposed between two Thales tools: a SOA Modelling Suite v3.1 and a risk assessment tool called Rinforzando v2.5.

We also present an evaluation of languages and tools developed by TUD in WP4 during the course of the project, from an industrial point of view. We discuss the interest of UMLsec and UMLchange in an industrial setting at the design level, and provide feedback on the early prototype of the CARiSMA tool.

Index

1 Introduction

Many large industries, like Thales, have adopted a clear strategy towards model-based system and software engineering. The reasons behind that choice are mainly the capability of such approaches to deal with the increasing complexity of software intensive systems, whilst complying with ever increasing constraints on costs and delays.

Now that the mainstream model-based system and software engineering practices have risen in Technology Readiness Level (TRL), these large industries are starting to examine how to extend their model-based engineering frameworks to cope with non functional properties, such as safety, security and performance. A strong trend is to envisage those specialty-engineering models as viewpoints (according to the ISO/IEC 42010:2011 [4] terminology).

The viewpoint approach allows for collaborative work on models between the mainstream system / software engineering teams and the specialty-engineering teams. It solves issues related to concurrent development and allows for the definition of clear interfaces between the teams, in terms of concepts that are manipulated and in terms of activities to perform by each team.

According the SecureChange technical annex, deliverable **D4.4 - Proof of concept integration of design modelling solution** includes "*an enhanced version of the Thales modelling environment that provides a proof-of-concept integration of the modelling solution produced in the WP. This will support a preliminary assessment of the security design modelling solution and of its integratability into industrial model-driven engineering environments*".

A first release of this document (D4.4a) was delivered at M24. This deliverable discussed and provided a proof of concept tool for the integration of a security risk assessment viewpoint with a UML modelling tool, namely Papyrus[1]. The reason for focusing on the design-time phases of the system / software development lifecycle was that it was assessed by Thales that the most critical decisions (in terms of programme cost and delays) are taken during the early steps of system / software architecting and design. It was thus important to address these steps as a priority. The proposed security risk assessment viewpoint encompassed the overall process of risk identification, risk analysis and risk evaluation[2], as defined by ISO/IEC Guide 73.

The D4.4a document first discussed the issues behind the integration of the security risk assessment viewpoint with the mainstream system / software engineering viewpoint. The integration was addressed through three angles: (i) orchestration of the processes; (ii) reconciliation of the concepts; (iii) integration through the technology. The document then presented the first version of the proof of concept tool that was developed; here, practical considerations about the workbench platform (e.g. UML, Eclipse EMF, etc.) were considered. Since then, the tool has gone through a major

[1] It is to be noted that at the time the UMLsec tool was also running on Papyrus, but using UML 1.4.

2 In fact the scope of the viewpoint extends slightly beyond the classical acceptation of risk assessment. It covers in particular part of the implementation of a risk mitigation strategy, as is normally expected from a risk management approach. The viewpoint, however, does not provide a complete risk management framework.

refactoring, to be redelivered at the end of the project as part of D4.4b. In particular, the human-machine interface was dramatically improved. The document closed on a series of annexes, mainly related to the risk assessment conceptual model and methodology.

The present document (D4.4b) represents the second release of the deliverable. It has been completely restructured and rewritten[3] to reflect the work progress performed during the third year of the project.

One of the major results of this last year (cf. §2.2) relates to integration through orchestration of the processes[4]. Integration through the orchestration of processes is performed by ensuring the consistency of the security engineering and risk assessment activities concurrently with the activities of the mainstream system / software engineering. When the mainstream system / software engineering approach iterates the analysis and design at different abstraction levels (i.e. functional, logical and physical levels), the consistency must be ensured throughout the abstraction levels, vertically between successive activities, and horizontally between concurrent activities (cf. §2.2.1). This work is extensively described herein, but it was also published as an invited paper at ServiceWave'11 [1].

As expressed in D4.4a, change[5] is the artefact that represents the glue between the process activities. Understanding precisely the nature of change is therefore paramount to the definition of a security engineering process and its orchestration with the mainstream system engineering processes. Thus, a thorough typology of changes applicable to security engineering and risk assessment has been defined (cf. §2.2.2). Parts of these changes have then been prototyped (cf. §3.1).

The basic Thales modelling environment used as input for this prototyping is the Thales SOA Modelling Suite (SMS). The enhanced version of the Thales modelling environment resulting from the SecureChange work is SMS tightly and loosely integrated with Rinforzando. The TUD CARiSMA tool is not integrated to the enhanced version of SMS because: (i) the Thales modelling environment is not UML based, and (ii) CARiSMA does not support DSL-based modelling environments. During Year 2 of SecureChange, for D4.4a, Thales worked using Papyrus to ease the integration with the TUD prototype, but that implied that the "modelling environment" was no more the "Thales modelling environment" and that the tool assessments in that environment lost their industrial value. Having an industrial environment was assessed by the Thales SecureChange project manager, and main author of this document, as more important in terms of project impact, than ensuring UML compatibility with the TUD prototype[6]. This is the reason why the CARiSMA tool has assessed separately (see details below), and the necessary model transformations from BPMN to UML were performed manually to ensure the continuity of the security engineering process. Obviously, to be fair with the tool, the evaluation Thales performed did not consider the lack of tool integration as a detrimental factor for CARiSMA.

[3] The content of D4.4a remains valid but, in order to save time, space and paper, it has not been copy-pasted in D4.4b. Interested readers should refer to D4.4a. Still, D4.4b can be read as a standalone document.

[4] As aforementioned, integration through the orchestration of processes was already addressed in D4.4a, but in much more general terms.

[5] Behind "change", one finds "traceability". Traceability is the means through which change can be managed. This deliverable does not focus on traceability, but the state of the art in this domain has been given some attention, cf. [6].

[6] This decision was proved right in terms of exploitation, cf. D8.3.

Compared to D4.4a, the work on processes in D4.4b was carried in much closer cooperation with UIB, work-package 2 leader and main beneficiary of the “Living Security Engineering Process” (defined in D2.1) and the “Integrated SecureChange Process” (defined in D2.2). This cooperation allowed for the modelling of the state diagrams implementing the Thales activity-driven process (of WP4) using the SecureChange artefact-driven process (of WP2). Full details can be found in §2.3.

Another major contribution of D4.4b with respect to D4.4a is the extension of the scope to cover early security engineering verification using CARiSMA, as provided by TUD (cf. §3.2) in October 2011. The assessed version of CARiSMA implemented only[7] a sub-part of UMLsec, which had been ported on UML 2.0. Thus, we provide herein an industrial evaluation of the UMLsec and UMLchange languages, on a conceptual standpoint, as well as an evaluation of CARiSMA, on a practical standpoint, related only to its implemented scope, i.e. a few UMLsec stereotypes.

A glossary completes the main part of this document. Terms used in this document and defined in the glossary are crafted as hyperlinks to their definitions and visually underlined.

This document closes on two annexes providing the Thales risk assessment tool installation manual and an extract of the operator handbook.

[7] The first version of CARiSMA known to implement UMLchange was demonstrated by TUD at the SecureChange GA meeting mid-January 2012. This version could obviously not be assessed by Thales.

2 An industrial instantiation of the SecureChange security engineering process

This chapter describes an industrial security engineering process for lifelong adaptable security-critical systems, orchestrated with a mainstream system engineering process, in a tool-agnostic manner. It is structured as follows:

- Section §2.1 provides an industrial review of the academic work performed in WP2 and positions our industrial process with respect to the processes defined in WP2;
- Section §2.2 is the core section about the industrial instantiation of the SecureChange process, with a focus on change as perceived by the industry.

2.1 A critical analysis of previous project results

In terms of "Integrated Security Process for Lifelong Adaptable Systems" two SecureChange former deliverables are relevant:

- D2.1: An architectural blueprint and a software development process for security-critical lifelong systems;
- D2.2: A configuration management process for lifelong adaptable systems.

Deliverable D2.1 provides a short state of the art of security engineering processes (cf. §2.1) and architectures (cf. §2.2). It then states 5 requirements for the project's new change-driven security engineering process (cf. §3.1.3): (i) support of multiple abstraction layers and of multiple engineering viewpoints / domains ; (ii) support of collaborations amongst stakeholders, based on their respective responsibilities; (iii) support of change propagation between viewpoints; (iv) seamless design-time and runtime operations; and (v) support of information retrieval, and model consistency. Based on these requirements, an artefact-driven "Living Security Engineering Process (LSEP)" is proposed (cf. §3.1.4-5). The defined process is independent of the selected artefacts, but five artefacts[8] were identified together with their dependencies, in order to specify change propagation in the specific case of the engineering of security-critical IT systems (cf. §3.2). A large part of deliverable D2.1 then illustrates the process in an industry-specific context (cf. §3.3) and from an architect-specific viewpoint, based on change patterns (cf. §4-§5), before showing its applicability to a more comprehensive engineering set-up (cf. §6). Deliverable D2.1 concludes on a generic architectural blueprint designed to accommodate a broad set of changes: the Security as a Service (SeAAS) approach (cf. §7).

Even though the "Living Security Engineering Process" defined in D2.1 is largely illustrated on the engineering of security-critical IT systems, its theoretical description is security-domain agnostic. It is in fact an artefact state-driven process to correctly handle change, which can be applied to any domain-specific engineering, e.g. safety

[8] Namely, the system model, the verification model, the risk model, the requirement model, and the test model.

engineering, performance engineering, etc. The process basically handles change on models and model elements, irrespective of what those models represent. This can clearly be seen from the five process requirements (see list above), in which the term “security” never appears. This lack was corrected in D2.2.

After a short state of the art on change management processes (cf. §3.2), deliverable D2.2 refines and instantiates the “Living Security Engineering Process” process for the specific case of engineering security-critical IT systems (cf. §3.3). The instantiated process is called the “Integrated SecureChange Process”. The instantiation is performed in order to cover most[9] of the engineering activities that are within the scope of the SecureChange project, i.e. requirements modelling (WP3), system modelling (WP4), risk assessment (WP5) and testing (WP7). This specialisation allows for a much more detailed process description than what was presented in §3.2 of D2.1. In particular, deliverable D2.2 describes the precise mapping between model concepts[10] and a set of detailed state machines. There is one drawback however to the “Integrated SecureChange Process” with respect to the “Living Security Engineering Process”, which makes it a comparatively light-weight process: instead of operating on different model element types, the basic unit of change of the “Integrated SecureChange Process” is a complete model. In other words, the fine-grained change propagation process has become a coarse-grain propagation process. There is however an advantage of this limitation: the change propagation process can now operate on black box models. Deliverable D2.2 then proceeds by presenting a tool support for the “Integrated SecureChange Process”, called MoVE (cf. §4), whilst the SeAAS approach is refined (cf. §5). The remainder of deliverable D2.2 describes the application of both the “Living Security Engineering Process” and the “Integrated SecureChange Process” on the SecureChange project’s ATM and HOMES case studies (cf. §6-§7).

From the above, it can be seen that deliverables D2.1 and D2.2 have proposed two new processes based on a shift a paradigm, from the legacy activity-centred organisation of the engineering lifecycle to an artefact-driven approach. From an industrial point of view, this paradigm shift raises a number of issues:

- **it wipes the slate clean of all legacy processes**, most of which are activity-driven processes with clearly defined inputs (pre-requisites) and outputs (post-conditions); the integration of an artefact-driven approach with existing legacy processes is not straightforward;
- **it is intrinsically event-driven** (i.e. each change is a new trigger to the process), **which raises questions on the repeatability of the process**[11]; solving this issue is necessary before certifying the engineering process;
- **the state diagrams of the artefacts**, and thereof the propagation of change amongst artefacts, **are currently independent of the contractual / legal status of the models**; in a commercial set-up, with the hypothesis of a multiple-abstraction-layers engineering approach, layers are progressively reviewed and signed by the customer as work progresses; change events need to propagate between artefacts

[9] Compared to D2.1, verification is not covered.

[10] E.g. an “asset” is a concept present both during requirements modelling and risk assessment, but it may not have exactly the same meaning in both models.

[11] In other words, would the engineering product be the same if change C1 occurs before change C2, with respect to C2 occurring before C1?

not only on a simple technical basis (i.e. semantic concept dependencies), but also on a commercial, legal and/or political basis.

From an industrial point of view, the drawbacks of the "Integrated SecureChange Process" presented above can be overcome by combining a legacy activity-centric approach with the SecureChange artefact-driven approach in order to get the best of both worlds:

- the legacy activity-centric approach deals with the domain specifics and the know-how of the organisation in terms of process decomposition and sequencing of activities; it also copes with all the commercial, legal and/or political constraints, as well as the norms and regulations in order to be recognised as a viable security engineering process that can be used to develop certified products;
- the artefact-driven approach deals with the mechanics to implement the legacy activity-centric process; it is practically transparent to the process end-users, i.e. the security engineers, but also all the other mainstream system engineering stakeholders; it ensures that change is propagated across models according to a set of rules that are activated depending on the model and/or model element states.

This fusion of a legacy activity-centric approach with the SecureChange artefact-driven approach is the subject of §2.3. But first, a legacy activity-centric approach to security engineering is presented in §2.2.

2.2 Security engineering process, an industrial view

This section presents Thales' understanding of an industrial security engineering process. It represents a detailed description of the process that was presented at ServiceWave'2011 [1] and rests upon a number of internal Thales Group processes [2][3]. This section is structured in two parts:

- Section §2.2.1 describes a legacy industrial activity-centric approach to security engineering;
- Section §2.2.2 extends §2.2.1 with a particular focus on change as seen in an industrial context.

Then, section §2.3 provides an example of how parts of this legacy industrial activity-centric approach can be implemented using the SecureChange artefact-driven approach.

2.2.1 A legacy activity-centric approach to security engineering

A system / software lifecycle typically has eight phases as illustrated in Figure 1, on page 17: (i) customer need analysis, (ii) specification, (iii) design, (iv) realisation or acquisition, (v) integration and verification, (vi) validation and deployment, (vii) operation and maintenance, and (viii) disposal. In some cases, a system / software may occupy several of these phases at the same time. The process is also iterative.

System / software engineering relates to the design & development of a system / software so that legitimate users can do what they want to do. Security engineering relates to the design & development of a system so that illegitimate users cannot do what they should not do. Thus, mainstream system engineering and security engineering have different overall goals, and this justifies a specific speciality process as defined herein for system / software security engineering.

It is possible to organise the security activities in three categories (each identified by a different background colour in Figure 1):

- analysis-type activities, with in particular, threat, vulnerability and risk assessment activities, which essentially produce security requirements;
- engineering-type activities, with in particular, the design and implementation of security solutions that satisfy the aforementioned security requirements;
- qualification-type activities, with in particular, the collection of evidence and accreditation of the system.

The system / software security engineering process can be conducted regardless of the system lifecycle phase; however the pursued goals may significantly differ depending on the phase. This section provides a coarse-grain description of the main pursued goals during each phase.

2.2.1.1 During the customer need analysis

The need analysis phase is classically performed by the customer strongly supported by industrial parties, using Enterprise Architectural Frameworks (EAF) such as the American Department of Defence Architecture Framework (DoDAF)[12], the English Ministry of Defence Architecture Framework (MODAF)[13], the French AGATE[14], the North Atlantic Treaty Organization Architecture Framework (NAF)[15] or more recently, in the civil world, the European ATM Enterprise Architecture (EAEA) framework. It classically produces an overall understanding of the system (or system of systems), and high-level end-user requirements as documented in one or more call-for-tenders.

In this phase, the main goal of security engineering is to elicit primary asset[16] security needs, mainly in terms of confidentiality, availability and integrity. This **security specification** work may be consolidated by a light-weight **threat source and feared event assessment**.

The elicitation of security needs of primary assets is typically a poor relation in the Enterprise Architectural Frameworks mentioned above. It is important to understand at this stage that we are working on security needs as expressed by the customer and/or end-users, not on security requirements resulting from a formalised security risk assessment process.

Security needs expressed by the customer and/or end-users are based on their own perception of the risks, which strongly depends on:

[12] http://dodcio.defense.gov/sites/dodaf20/.

[13] http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/.

[14] French for « Atelier de Gestion de l'ArchiTEcture des systèmes d'information et de communication », cf. http://www.ixarm.com/-AGATE-.

[15] http://en.wikipedia.org/wiki/NATO_Architecture_Framework.

[16] Even though the identification of primary assets is an important step in any security specification work, we assume here that the identification of primary assets is correctly performed by the mainstream system engineering using an EAF.

- an identification and assessment of threat sources, and in particular, how much they trust their collaborators, equipment and/or environment; this is a “politically” difficult activity to lead, as it is not easy to point to a collaborator as being potentially untrustworthy; however, if we do not consider trust in the architecture, security measures imposed on the system later might be heavier-handed than necessary (i.e. the implementation may introduce pointless protection mechanisms that just hinder operations in a trusted domain), or worst, miss the point (i.e. the implementation of the system may implicitly assume trust relationships, amongst users or between users and the system, that are simply not there);
- an identification and assessment of feared events, i.e. what can go wrong with respect to the primary assets, as triggered by the threat sources, and how bad would the consequences be.

When stating their primary asset security needs, the stakeholders (i.e. customer and/or end-users) may overestimate or underestimate their risks. This will be determined later during the risk assessment process. The important task here is to understand the reasons behind these security needs, and in particular the socio-technical context that supports those security needs.

The following phases, until operation, are mainly performed by the system / software provider.

2.2.1.2 During the specification phase

During the mainstream system / software specification phase, the main goal of the security activities is to gain early assurance that the proposed overall architectural solution is sound with respect to security concerns. This is performed basically through two activities:

- a threat, vulnerability and risk assessment;
- the specification of security requirements.

During this phase it is important to be able to quickly update requirements and models, and bring them in synch. The end-customer might be involved in the loop and must be able to do some form of what-if scenarios. Evidence is collected in order to contribute later to the security assurance argument (cf. §2.2.1.6).

2.2.1.2.1 Threat, vulnerability and risk assessment

The threat, vulnerability and risk assessment process consists in identifying all risks, and defining security objectives to reduce the unacceptable risks to acceptable levels.

The primary assets, their related feared events and security needs, as elicited during the customer need analysis phase (cf. §2.2.1.1) are a key input to this work.

The other key input is the considered system / software baseline, including, if applicable, the planned deployment configurations. The baseline can be represented in a wide variety of forms:

- system / software requirements, some of which may already be explicitly tagged as security requirements; these requirements will typically be stored in a requirements’ database such as DOORS;
- system / software models, as produced by the mainstream engineering activity, some of which may correspond to specified security solutions;

- already existing system / software, some of which may be security solutions;
- any combination of the above.

The threat, vulnerability and risk assessment process will need to ensure the traceability to the baseline, irrespective of the forms under which the baseline is available.

If there are multiple planned deployment configurations in the baseline, the threat, vulnerability and risk assessment process will need to be performed for each deployment configuration, maximising the reuse of the analyses between the deployment configurations.

Example: an Air Traffic Control system must be deployed at the 3 major airports of one country. It is planned to be 1st deployed in shadow-mode at one of the airports, then deployed operationally at that airport, and finally to be deployed operationally at the three airports and interlinked with each other. The threat, vulnerability and risk assessment process is performed for each deployment configuration, each time ensuring the traceability to the subset of the baseline that deals with that configuration.

2.2.1.2.2 Specification of security requirements

The specification of security requirements process consists in transforming the aforementioned security objectives (cf. §2.2.1.2.1) into security requirements, and integrating those requirements into the baseline system / software requirements. Residual risks are identified and all decisions are formally validated, in order to proceed with the design.

2.2.1.3 During the design phase

During the design phase, a breakdown of the system / software is established. The **security design** activity participates to this work by addressing the security related non-functional constraints, as formalised during the specification of security requirements (cf. §2.2.1.2.2). It results in the design of cost-efficient countermeasures for the identified security risks, with minimal impact on the architectural solution.

Depending on the system / software complexity, the threat, vulnerability and risk assessment process performed during the previous phase at a coarse-grain level (cf. 2.2.1.2.1) may be refined for some system / software components, based on the detailed design knowledge elaborated during this phase. If such a refinement is performed, the assessed risk levels[17] at component-level will need to be aggregated and compared to the risk levels established previously at a coarser-grained level (cf. §2.2.1.2.1). This may result in a revision of the overall risk level and therefore on a revision of the security requirements (cf. §2.2.1.2.2).

Evidence of the security design activity is collected in order to contribute later to the security assurance argument (cf. §2.2.1.6).

Depending on the complexity of the security requirements (i.e. the level of expertise required to handle them), the security design activity may be fused with, or kept separate from the mainstream system / software logical and physical design.

[17] In terms of likelihood of the threats and severity of the consequences.

2.2.1.4 During the realisation or acquisition phase

During the system / software realisation or acquisition stage, the main goal of security engineering is to implement or acquire the security solutions as previously designed (cf. §2.2.1.3).

Evidence of the realisation or acquisition of the security solutions is collected in order to contribute later to the security assurance argument (cf. §2.2.1.6).

Depending on the complexity of the security design (i.e. the required level of expertise), the **security solutions realisation or acquisition** activity may be fused with, or kept separate from the mainstream system / software realisation or acquisition.

2.2.1.5 During the integration and verification phase

During the system / software integration and verification stage, the main goal of security engineering is to integrated and verify the security solutions as previously implemented or acquired (cf. §2.2.1.4).

Evidence of the integration and verification is collected in order to contribute later to the security assurance argument (cf. §2.2.1.6).

Depending on the complexity of the security solutions (i.e. the required level of expertise to integrate and verify them), the **security solutions integration and verification** activity may be fused with, or kept separate from the mainstream system / software integration and verification.

2.2.1.6 During the validation and deployment phase

During the validation and deployment phase, the main goal of security engineering is the **security qualification** of the system / software.

The qualification results must be fed back to the **threat, vulnerability and risk assessment** and compared to the security objectives. Residual risks will need to be identified and reviewed anew.

Accreditation is an official authorisation by management for the operation of a system / software, and acceptance by that management of the associated residual risks [5]. It establishes the extent to which the security controls are implemented correctly, operating as intended, and producing the desired outcome with respect to meeting the system / software security requirements. Accreditation is usually based on the certification process as well as other management considerations.

2.2.1.7 During the operation and maintenance phase

During system / software operation, the main goal of security engineering is to **maintain the system / software under secure conditions**. This is basically performed by continuously monitoring the effectiveness of the countermeasures to determine the extent to which the controls are operating as intended, and producing the desired outcome with respect to meeting the security requirements for the system / software. The different **threat, vulnerability and risk assessment** steps should also be regularly revisited in order to cope with environmental and internal changes during the system / software's lifetime.

2.2.1.8 During the disposal phase

There is no bright line that separates "secure" from "insecure" forms of disposal. What is appropriate in a particular situation depends on the sensitivity of the information at issue, and the perceived threats to it. Therefore, one of the main security engineering activities during the system / software disposal phase will be a dedicated security **threat, vulnerability and risk assessment**. Similarly to the security threat, vulnerability and risk assessment performed for the building of the system / software, this activity will produce security requirements. These requirements will be handled by the **secure disposal** process, including, but not limited to, shredding of paper media, appropriate cleaning of electronic (i.e. magnetic or optical) media, and secure recycling of valuable equipment.

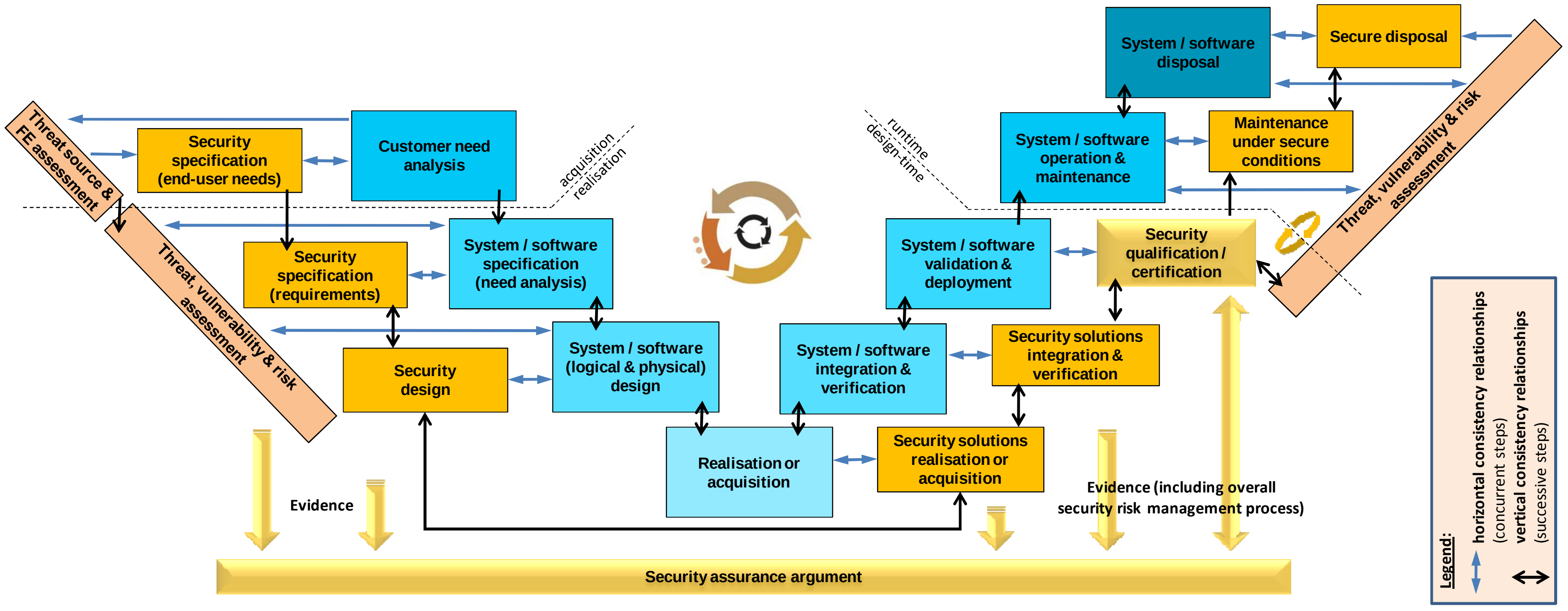

Security specification (end-user needs)
Customer need analysis
Threat source & FE assessment
Security specification (requirements)
System / software specification (need analysis)
Threat, vulnerability & risk assessment
Security design
System / software (logical & physical) design
acquisition
realisation
Realisation or acquisition
Security solutions realisation or acquisition
System / software integration & verification
Security solutions integration & verification
System / software validation & deployment
Security qualification / certification
runtime
design-time
System / software operation & maintenance
Maintenance under secure conditions
System / software disposal
Secure disposal
Threat, vulnerability & risk assessment
Evidence
Evidence (including overall security risk management process)
Security assurance argument
Legend:
horizontal consistency relationships (concurrent steps)
vertical consistency relationships (successive steps)

Figure 1: A legacy activity-centric approach to security engineering orchestrated with a mainstream system / software engineering V-cycle

2.2.2 An industrial typology of change applied to security engineering

2.2.2.1 Introduction

The overall security engineering process presented in §2.2.1 runs through the complete system / software lifecycle, and therefore encompasses a large number of changes which allow going from "*I have an idea for a new system / software*" to "*the system has been disposed of*", via the development of the said system and years of secure and successful operations. This section provides an industrial typology of change[18] viewed from the threat, vulnerability and risk assessment point of view when it is performed in parallel to the following mainstream system engineering activities (cf. Figure 1 and §2.2.1.2.1):

- system / software specification (need analysis),
- system / software (logical & physical) design,
- system / software validation.

This scope covers the complete design-time activities under the responsibility of the industry, which are performed in parallel to the threat, vulnerability and risk assessment activity. The threat, vulnerability and risk assessment activity relies on the following assumptions:

1. the existence of an initial set of system / software requirements, called "baseline" (cf. Figure 2), which describes the system / software under study; the baseline contains both functional and non-functional requirements, some of which may be security requirements;
2. security needs (cf. §2.2.1.1) mainly but not exclusively expressed in terms of confidentiality, availability and integrity, and potentially supported by:
   a. a list of threat sources, with their related likelihood of action, and
   b. a list of feared events, with their related severity, as expressed by the customer and/or end-users.

Depending upon the item under study, the number of security requirements to consider may range:

- from a few tens, for a not too security-demanding software,
- to a few thousands, for highly security-critical and complex industrial systems.

It is of course in the latter cases that change support is required, and therefore, a compulsory requirement of the change support is related to its scalability.

This section proposes four types of changes:

- Change type n°1: minor specification and/or design change;
- Change type n°2: risk treatment;

[18] This industrial typology of change is presented in a tool-agnostic manner. For an illustration based on the D4.4b prototype, please refer to §3.

- Change type n°3: publication of qualification results;
- Change type n°4: change of configuration.

2.2.2.2 Change type n°1: minor specification and/or design change

The threat, vulnerability and risk assessment models encompass a number of concepts that make direct reference to mainstream engineering concepts, in particular:

- the primary asset concept, which makes reference to anything that is important[19] to the end-user; a typical example is an operational or business process;
- the supporting asset concept, which makes reference to anything that supports primary assets; a typical example is a data store;
- the security requirement concept, which makes reference to a security requirement included in the baseline;
- the security solution concept, which makes reference to an element already designed, implemented and qualified in the system / software under study.

Any change in the mainstream engineering objects that are, or should, be referenced by threat, vulnerability and risk assessment objects obviously have an impact on the assessment. Thus, the 1st type of change selected in this typology is represented by minor specification and/or design changes (cf. Figure 2). These changes should be handled as transparently as possible for the security expert. Please refer to §3.1.2 to see how this type of change was handled in the D4.4b prototype.

2.2.2.3 Change type n°2: risk treatment

The main output of threat, vulnerability and risk assessment is represented by security objectives. A security objective is an expression of the decision to treat a risk according to prescribed methods[20], i.e. a risk treatment plan. The security specification activity (cf. §2.2.1.2.2) transforms these security objectives into security requirements, which are then included into the baseline. This is the 1st step to effective risk treatment.

For the threat, vulnerability and risk assessment activity, this change is rather easy to cope with, because it is somehow self-triggered[21]. It mainly consists in being able to distinguish:

- the initial risk level, in terms of likelihood and consequences,
- the residual risk level, assuming that the security objective will be complied with; if the security objectives are not complied with, then the residual risk levels must be revised.

2.2.2.4 Change type n°3: publication of qualification results

As explained in §2.2.2.3, the specification of security objectives subsumes a positive impact on the risk levels, from unacceptable levels to acceptable levels. The likelihood

[19] I.e. has value, and therefore needs protection.

[20] We distinguish in particular risk reduction, risk transfer, risk avoidance and the risk retention.

[21] The only exception could be that a security objective is not transformed into a security requirement.

assumptions[22] are checked during the security qualification / certification activity (cf. §2.2.1.6). Each likelihood assumption may be:

- confirmed; in that case, the related risk level is confirmed at its acceptable residual level, and the risk should visually[23] disappear;
- or infirmed; here, the change is subtle, as it does not concern the system (or software) itself, but the expectations[24] that we have from it, i.e. the residual risk level is proved wrong; therefore the change must be handled by a capability of recording the new qualified threat (or risk) likelihood, without loosing the history if the initial likelihood assessment; new or alternate security controls will need to be put in place.

2.2.2.5 Change type n°4: change of configuration

A system configuration is a specific definition of the elements that define and/or prescribe what a system is composed of at a given time. Configuration control is the process of regulating major and long-term planned changes to hardware, firmware, software and documentation throughout the development and operational life of a system.

As stated in §2.2.1.2.1, when a threat, vulnerability and risk assessment is started, multiple deployment configurations might already be planned. Quite obviously, the threat, vulnerability and risk assessment must be able to deal with these different configurations, either by proceeding with the configurations in turns, or by running the assessments in parallel.

The basic capability to deal with these major and long-term planned changes is to be able to deal with the presence or absence of some configuration items (e.g. primary assets, supporting assets, threats, etc.), within a given configuration.

More subtly, the threat, vulnerability and risk assessment must also be able to deal with the changes of the values of some attributes of some configuration items that are present in multiple configurations.

Example: Let us suppose that a primary asset (e.g. a service), its supporting asset (e.g. a computer) and some threat scenario (e.g. theft) on the supporting asset are present in two configurations. Let us also suppose that the environmental conditions are different between configuration 1 (e.g. the supporting asset is in the factory) and 2 (e.g. the supporting asset is on site), and that these differences have an impact on the likelihood of the threat (e.g. theft is likely in the factory, negligible on site because of the highly secured premises). Here, the threat belongs to both configurations, but the threat likelihood attribute has a different value for each configuration. This in turn will impact the risk level, so the associated risk will also have different attribute values for the different configurations. Finally, if we suppose that the impact on the risk is significant, then the risk may be acceptable in one configuration, and unacceptable in the other. This is turn will trigger different values for the security objectives and security requirements associated with the risk within the different configurations.

[22] The qualification applies only to the likelihood, not to the consequences (i.e. severity).

[23] The history of risk treatment must be recorded, but means must be given to filter out acceptable residual risk in order to allow for the attention of the security expert to be focused on the remaining unacceptable risks, until all risks have become acceptable.

[24] On Figure 2, this is shown by an arrow stemming from and pointing to the risk assessment activities.

As can be seen from the trivial example above, the management of the major and long-term planned changes (i.e. configurations) of a system can quickly become highly complex when the system is large and when there are many (e.g. 5 or more) foreseen configurations.

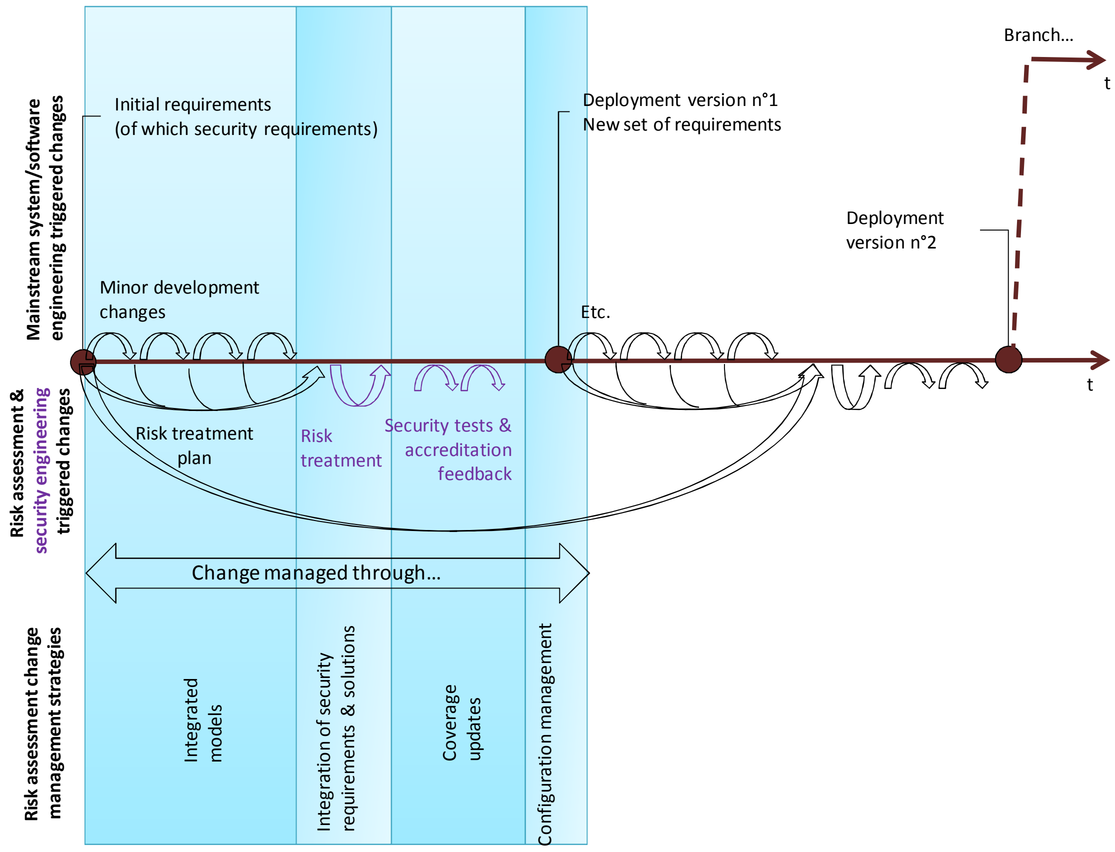

Mainstream system/software engineering triggered changes
Risk assessment & security engineering triggered changes
Risk assessment change management strategies
Initial requirements (of which security requirements)
Minor development changes
Deployment version n°1 New set of requirements
Etc.
Deployment version n°2
Branch...
t
t
Risk treatment plan
Risk treatment
Security tests & accreditation feedback
Change managed through...
Integrated models
Integration of security requirements & solutions
Coverage updates
Configuration management

Figure 2: An industrial typology of change

2.2.3 An industrial feedback on an academic proposal for change representation at the design level

In the context of the project, TUD provided with UMLchange[25] a means of representing change in UML models, typically produced at design level. UMLchange is a UML profile dedicated to the explicit representation of changes in a UML model. It evolved from the UMLseCh profile that was divided into its security-related component UMLsec and its change-related component UMLchange with the move to UML 2.

Consider a UML model subject to changes. UMLchange works by representing all the possible states of the model in a single model. That is, all the model elements present in one of the states are part of the model. Then, UMLchange stereotypes are applied to these elements to denote changes and clarify which element belongs to which state.

[25] The assessment of UMLchange is performed only at conceptual level, in chapter 2, because TUD did not provide a prototype implementing UMLchange until January 2012. Chapter 3 below is dedicated to prototype assessments.

UMLchange contains stereotypes for representing atomic changes, which are the addition of new model elements («add»), the deletion of model elements («delete»), and the substitution of model elements with others («subst»). Complex changes are described by aggregating changes («change set»). In addition, constraints can be applied to changes (through the {constraint} tag) to specify under which condition is the change to be applied. In particular it is possible to express that a change shall only be applied when another specific change has also been applied, or has not been applied, or under any Boolean combination of such conditions.

This framework is obviously an extremely generic way of expressing change in a UML model, as it is intended to be. The various tags associated with the stereotypes are of type String, which emphasizes the flexibility of the notation as anything might be encoded into the notation. In particular, {constraint} is of type String. The syntax and semantics of the language used to express them are not fixed. It is assumed that those should be provided by the analysis tools manipulating UMLchange stereotypes.

From the industrial point of view, UMLchange shall be seen as a building block for representing change in general, not as an operational solution. Indeed, the constraints of an industrial environment in this respect are many:

- A representation of change is meaningless if it is not supported by analyses. This, in turn, requires semantics to be attached to the change objects, and therefore to restrict the expressivity of the language. For example the {constraint} tag might be used to refer to other changes in the same model, or to changes in other models, or to external object representing, for instance, the decision of a board to allow a certain change or not.
- Changes must be traceable and justified; in particular it may be necessary to provide a link to the requirement(s) that cannot be met unless the change is implemented, or a link to the analysis (for example, a risk analysis) which determined the necessity for the change.
- UMLchange explicitly represents change as a first-order citizen in the model, which puts on the same level two distinct activities: the manipulation of model elements and the management of change. Industrial processes impose a strict separation of these activities. The consequence is that the modelling tool must take this point into account by providing a restricted access to the underlying modelling facilities supported by the model.

2.3 A focus on the design phase security modelling and change management using the UMLsec & UMLchange languages

The UMLsec language is a UML profile consisting of a collection of stereotypes that are meant to convey security-related information of the following kinds:

- Security characteristics of the physical and logical levels of the system. They can be seen as assumptions (say, of the known security characteristics of a readily available component) or specifications (guiding the choice of a component matching these security requirements).
- High-level security policies expressing a constraint on the model.

The combined use of those stereotypes enables security analyses to be performed on a model. Typically a stereotype expressing a security policy is applied on a container (e.g. a UML package) and the associated constraint involves the security characteristics expressed by the UMLsec stereotypes attached to the sub-elements. We refer to each such constraint and its verification as an analysis. In this section, we discuss the analyses that are most relevant to industrial use in general. The ATM use case is used as a concrete example to illustrate the use of the selected analyses, but our choice of analyses is based on industrial needs that go beyond the project's use case.

2.3.1 ATM use case: TCC physical arrival

In the following sections we identify several UMLsec analyses that are relevant in an industrial setting. We give a more concrete flavour by illustrating each check with an example from the ATM use case, namely the process followed by a TCC when physically arriving in a control room.

We developed an activity diagram[26] (see Figure 3) describing the process of the arrival of a TCC in a control room. It was developed with Papyrus UML. Notes above the diagram indicate actors and activities are aligned with the note of their actor. Next, we performed security analyses with UMLsec.

Upon arrival, the TCC goes through a security check involving the identification and the deposit of unauthorized items, before being allowed to proceed to the control room. Several security properties of the process may be expressed in terms of UMLsec analyses.

- Identification: ensure that the TCC is properly identified.
- Auditing: ensure that some specific actions in the process can be traced.
- We illustrate the use of selected UMLsec analyses by showing how they can help verify that the process has those properties.

[26] In D4.4b, we keep the discussion on a purely technical level, as our aim is to introduce UMLsec's features that seem relevant. An ATM business-oriented presentation of the same process can be found in D1.3, where the same UMLsec analyses are mentioned and put in the context of the use case. The reader is encouraged to refer to that deliverable for more information.

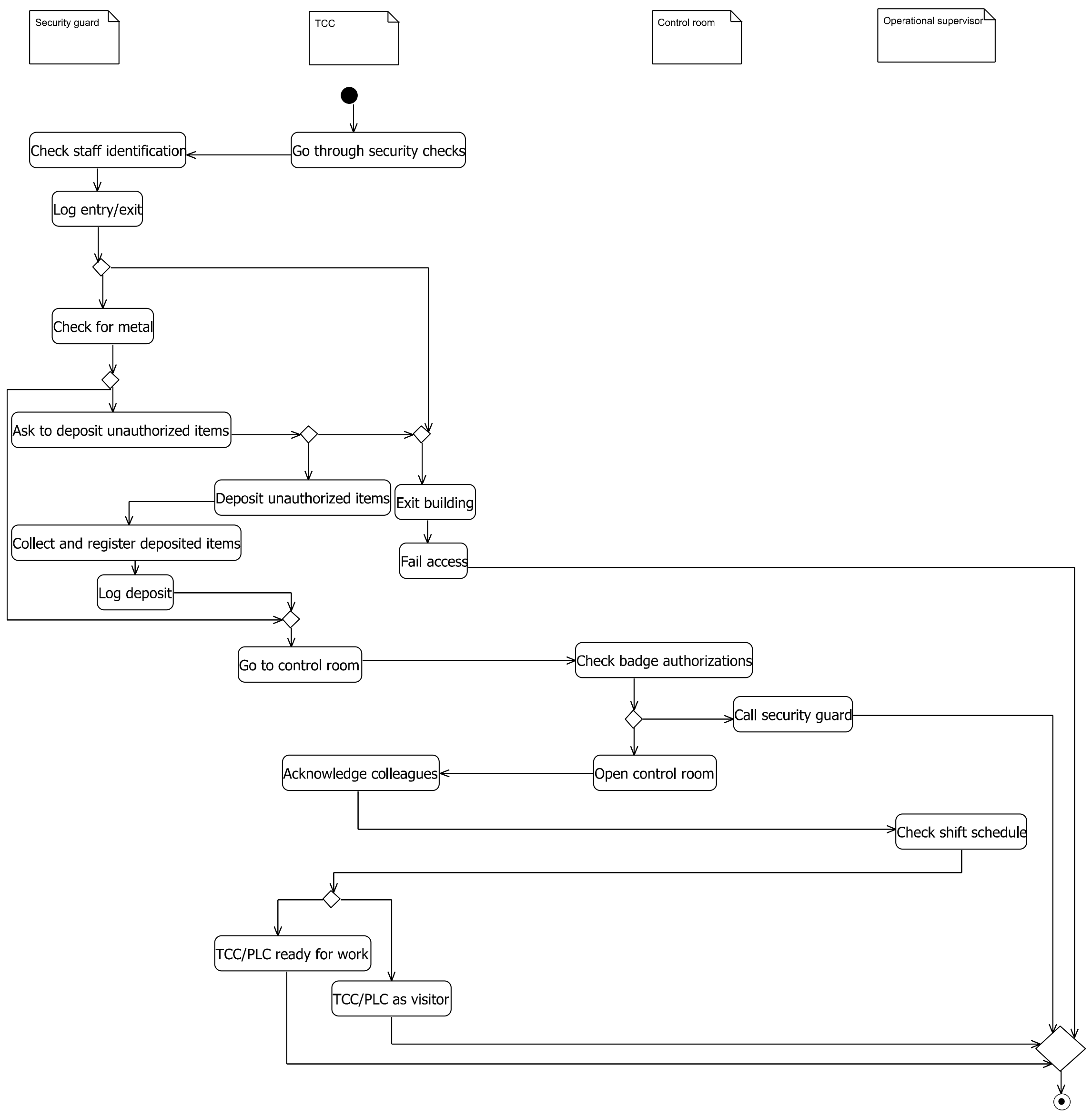


Figure 3 - Activity diagram for the physical arrival of the tactical controller (TCC)

2.3.2 The fair exchange analysis

The fair exchange analysis applies to an activity diagram, typically the one given above in Figure 3. It consists in verifying a certain liveliness property; its interest lies in its generality.

The analysis is enabled by applying the «fair exchange» stereotype to a package containing an activity diagram. This stereotype has associated tags {start} and {stop} whose values are lists of actions from the activity. The associated constraint can be stated as follows: every path in the activity diagram going through an action in {start} must subsequently go through an action in {stop}. The name “fair exchange” refers to a possible use of this analysis. For a transaction protocol to be considered secure, one must have the assurance that any payment will be followed by either the delivery of the purchased good or a refund.

Some of the aforementioned properties can be expressed as fair exchange problems.

- Identification: ensure that the bearer of an unauthorized badge is not allowed to proceed. This constraint can be expressed by taking {start = Check badge authorizations} and {stop = Open control room, Call security guard}.
- Auditing: ensure that the process enforces the traceability of identification checks. This constraint can be expressed by taking {start = Check identification} and {stop = Log entry/exit}.
- Auditing: ensure that the process enforces the traceability of item deposits. This constraint can be expressed by taking {start = Deposit unauthorized items} and {stop = Log deposit}.

Fair exchange is useful as a general purpose liveliness check. In model checking, liveliness properties are those of the form “X will eventually happen”; together with safety properties, of the form “X will never happen”, they form the basic building blocks for expressing high-lever properties.

One important characteristic of the fair exchange analysis is that each instance of the stereotype supports one analysis. If several fair exchange constraints must be verified simultaneously by the same model (such as in the ATM example), then the modelling tool must support attaching several instances of a stereotype to the same model element. This restriction is important to bear in mind.

2.3.3 Provable

The provable analysis applies to an activity diagram as well. It can be used in a similar way as fair exchange, but in the opposite direction; it requires that an action be executed before another.

The analysis is enabled by applying the «provable» stereotype to a package containing an activity diagram. This stereotype has associated tags {action} and {cert} whose values are names of actions from the activity diagram. The associated constraint can be stated as follows: every path in the activity diagram going through the action named by {cert} must have previously gone through the action named by {action}. The name “provable” refers to a possible use of this analysis. {cert} serves as a certificate (or proof) that the action named by {action} was performed.

Some of the aforementioned properties can be expressed as provable problems.

- Unauthorised items: the TCC may only go to the control room after having been checked for metal. This constraint can be expressed by taking {action = Check for metal} and {cert = Go to control room}.
- Identification: the control room may only be opened after proper authorisations have been checked. This constraint can be expressed by taking {action = Check badge authorizations} and {cert = Open control room}.
- Auditing integrity: logs are only written to when needed. This constraint can be expressed by taking {action = Check staff identification} and {cert = Log entry/exit}, or {action = Collect and register deposited items} and {cert = Log deposit}.
- Identification: the TCC must be acknowledged by their colleagues before being allowed to work. This constraint can be expressed by taking {action = Acknowledge colleagues} and {cert = TCC/PLC ready for work}.

Provable is a versatile check extremely useful to a range of security problems. This is clearly understandable; the solution to many security problems consist in requiring that a security check be passed before a sensible action is performed.

The provable analysis suffers from the same limitation as the one detailed for fair exchange.

2.3.4 Role-based access control

The classic role-based access control paradigm is implemented in UMLsec with the help of a dedicated stereotype, «rbac». This stereotype is applied to a package containing an activity diagram with actors and swim-lanes and allows controlling which actors are allowed to perform which actions in the diagram.

There are three associated tags, {protected}, {role}, and {right}. {protected} is a list of actions that need protection, {role} is a string describing a list of pairs (actor, role) assigning roles to actors, and {right} is a string describing a list of pairs (role, action) specifying which roles are allowed to perform which actions.

The associated constraint is the following: each action in the swim-lane of an actor must be allowed for some role assigned to the actor.

The process for the ATM use case presented above involves four actors and could benefit from the use of RBAC to protect some of the most sensible activities. For example, the following properties could be enforced by UMLsec:

- Auditing integrity: logs should only be performed by accredited personnel. This can be enforced with $\{\text{Log entry/exit, Log deposit}\} \subseteq \{\text{protected}\}$, $\{(\text{Log entry/exit, Logger}), (\text{Log deposit, Logger})\} \subseteq \{\text{right}\}$, and $\{(\text{Security guard, Logger})\} \subseteq \{\text{role}\}$.
- Identification: the TCC must be allowed to work by their supervisor. This can be enforced with $\{\text{Check shift schedule}\} \subseteq \{\text{protected}\}$, $\{(\text{Check shift schedule, Shift manager})\} \subseteq \{\text{right}\}$, and $\{(\text{Shift manager, Operational supervisor})\} \subseteq \{\text{role}\}$.

In contrast to fair exchange and provable, the rbac analysis can support all the above constraints with a single stereotype application.

2.3.5 Concluding remarks

UMLsec supports a catalogue of security analyses. We presented above those that seemed the most appropriate to a generic treatment of security in industrial applications, and illustrated it through the ATM use case.

As far as the ATM use case is concerned, an important point to consider is that the various properties are not supported out-of-the box, but must be encoded in terms of the various available UMLsec analyses. The mapping is not always obvious, but that is to be expected when using generic analyses (e.g. fair exchange, provable…) as opposed to analyses tailored to a specific problem (e.g. identification, logging, etc.)

In particular, the fact that a high-level property has to be encoded into a conjunction of lower-level constraints, thus requiring the application of several instances of a stereotype to the same model element, should be seen as a risk considering the limitations of some of the current modelling tools in that respect.

2.4 Instantiation using the “Integrated SecureChange Process”

A collaboration was established between THA and UIB to implement parts of the security engineering process, as described in §2.2, using the results of work-package 2. In this collaboration, THA described the change management process between its risk assessment tool (called Rinforzando v2.5) and its mainstream system engineering tool (called SMS v3.1) whilst UIB created the corresponding state machines that both describe the model states in each tool, and ensure the change impact propagation between the tools (cf. §2.4.2). This work was performed through a number of teleconferences and email exchanges. It required a significant amount of iterations, which were beneficial for both parties in terms of clarification and formalisation of the processes and mechanisms underlying the models (cf. §2.4.1).

2.4.1 Short introduction to MoVE modelling

In general, a process is mapped to MoVE[27] by writing the appropriate state machines using the W3C State Chart XML (SCXML) standard.

To handle to interactions between processes, the standard state charts are insufficient. UIB has extended the state charts with actions (cf. §2.4.1.2) and events (cf. §2.4.1.2).

2.4.1.1 Actions

An “action” allows a state machine to trigger a transition in one or more different state machines. Actions are defined on the entry[28] of a state, using the onEntry tag of the SCXML standard. Actions are optional. Example:

```
<state id="registered">
<onentry>
<ocl:adaptation actions="Action[query=let result: Course = self.for_subject
in result event=checkRegistrations]" />
</onentry>
<transition event="schedule" target="scheduled"/>
</state>
```

In the above example, the SCXML engine triggers its action listener every time the state registered is entered. If no action is defined, nothing is triggered.

An action tries to trigger a transition from the current state of an object to the desired state of the object. It does not switch to the desired state if the switch is not supported by an existing transition, whose transition conditions are fulfilled. In case no transition exists, an exception is raised.

On a modelling point of view, UIB uses UML representations of the state machines to represent the SCXML state charts. This approach lacks accuracy. Alternatives are being studied.

[27] The MoVE tool stands for Model Versioning and Evolution and is a tailor-made model repository to support Living Models and change-driven processes. For details, please refer to D2.3.

[28] An action could also be defined upon exit of a state (onExit tag) but the need for such a behaviour has not yet occurred.

2.4.1.2 Events

For some processes even the action-extended state machines are not expressive enough to cover all rules.

To overcome problems with the limited expressiveness of state machines UIB has implemented an event based mechanism that "cooperates" with the state machines. The basic idea of the events is that MoVE analyses changes in the model and generates messages such as "*the name attribute of element with ID 1234 changed from Risk-1 to R-1*". MoVE plugins can register for certain type of events and certain models and respectively receive all corresponding messages.

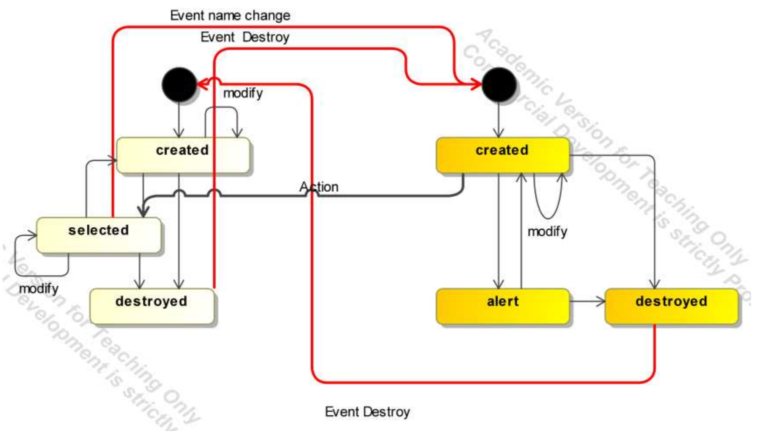


Figure 4: Modelling interactions between processes using MoVE

In the figure the red arrows show events that are sent via MoVE to plugins listing to the corresponding events. They point to the initial notes simply to denote the semantics of the event.

2.4.2 Instantiation

As described in §2.1, the process modelling work performed in WP2 allows for the creation of two versions of the state-machines:

- one which supports a fine-grained process based on the state of single model elements;
- one which supports a coarse-grained process which is based on the state of the entire SMS and Rinforzando models.

In our case of the coarse-grained model does not allow for the capture of the interactions as developed by Thales between SMS and Rinforzando. This section therefore presents a set of fine-grained state machines (cf. Figure 5) that captures some of the interactions between SMS and Rinforzando as detailed in §2.2.1, §2.2.2 and §3.1. To capture correctly the interactions, it was needed to model the state-machines of three model elements:

- mainstream system engineering model elements (shown on the left side in Figure 5);

- the proxies of the mainstream system engineering model elements within the risk assessment tool (shown in the centre, in Figure 5);
- the assets identified and created in the risk assessment study (shown on the right side in Figure 5).

To better understand the logic of the state machines, the reader is also invited to read about the software architecture, in §4.3.

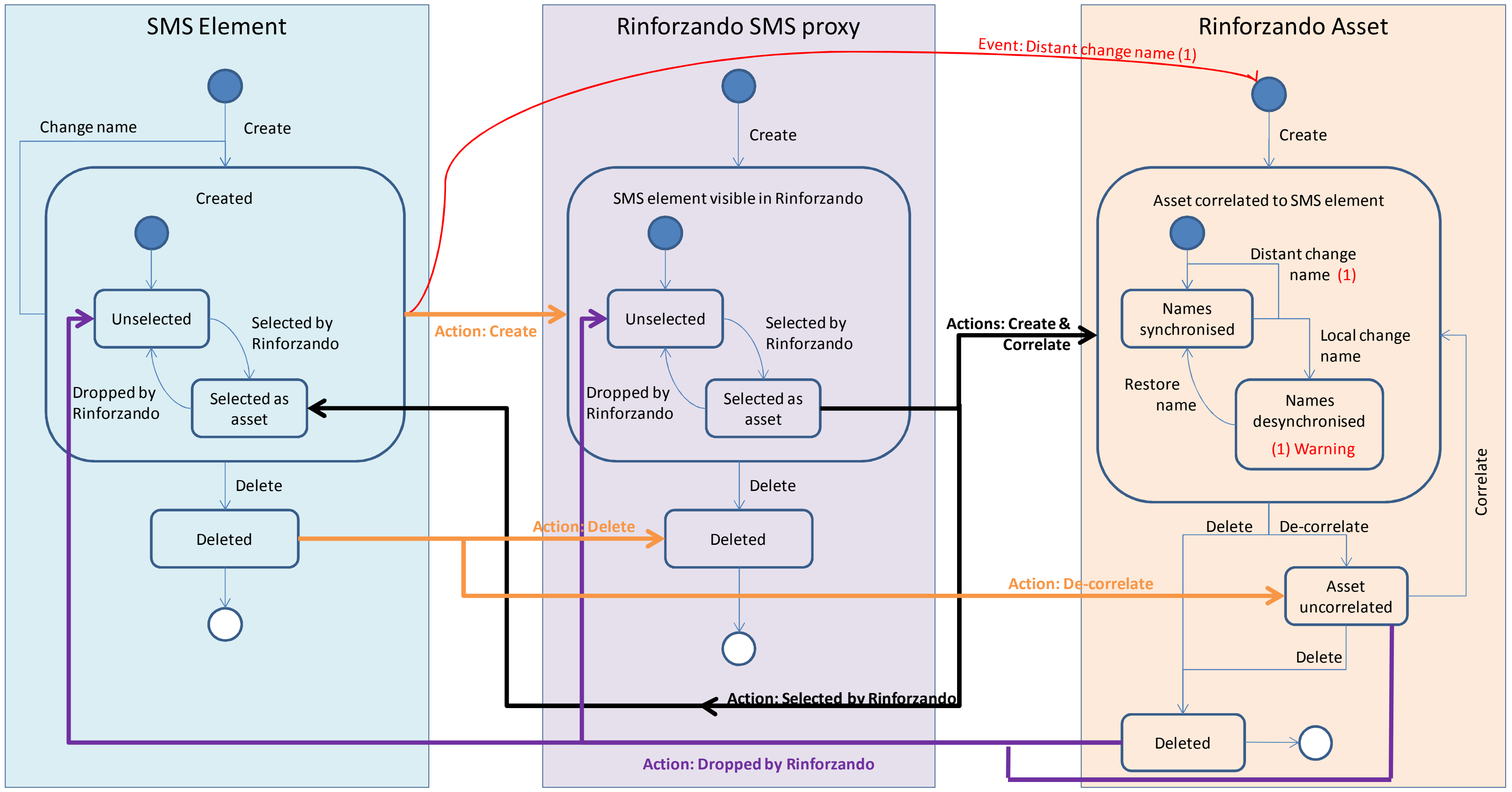


Figure 5: Fine-grained process state machine

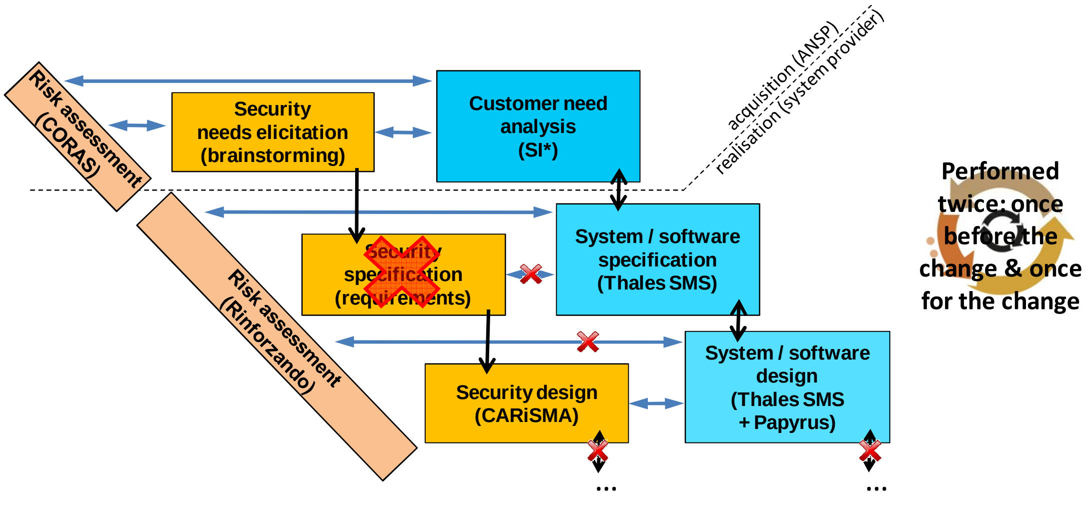
It is interesting to note that both sub-states (i.e. “Unselected” and “Selected as asset”) of the “Created” state of an SMS element have inbound actions, and no outbound actions. This is because these states do not exist as absolute states of SMS elements; these states exist only in relation to Rinforzando, i.e. a tool external to SMS. In other words, someone who would have modelled the SMS element state-machines, taking SMS as a standalone tool, would never have modelled those states. The SMS element state-machine is therefore completely specific to the integration with Rinforzando.

To close the description, two short explanations about:

- The proxy state machine, which triggers the creation of a Rinforzando asset element. This is because a selection window of Rinforzando shows the list of existing SMS elements. Each element in that list of the Rinforzando tool is an SMS proxy. When the user selects an element in the list, a Rinforzando asset is created on the diagram.
- The “Asset uncorrelated” state, which is reached when the correlated SMS element sends an action that it was deleted and which sends back an action message to the already deleted element that the correlated element is now uncorrelated. In fact, no message is to be sent back if the origin of the state is triggered by a deletion. Indeed, the Rinforzando asset can also be decorrelated manually within Rinforzando. This action is not supported by the tool HMI, but it can be done using the Properties window of Eclipse. This was semantically expressed by the label “Dropped by Rinforzando”.

3 Illustration on the prototypes

This chapter focuses on a small part of the overall security engineering process as described in §2.2. Figure 6 shows the scope of the ATM running example, as extensively described in D1.3, where the tools used for each step are clearly specified. This chapter enters into the practical and technical details of some of the activities performed in the D1.3 running example.



Risk assessment (CORAS)
Security needs elicitation (brainstorming)
Customer need analysis (SI*)
acquisition (ANSP)
realisation (system provider)
Risk assessment (Rinforzando)
Security specification (requirements)
System / software specification (Thales SMS)
System / software design (Thales SMS + Papyrus)
Security design (CARiSMA)
Performed twice: once before the change & once for the change
...
...

Figure 6: Scope of the ATM running example for the D4.4b prototype illustration

In section §3.1, the focus is on the change management between the risk assessment performed using Rinforzando, and the system / software specification using the Thales SOA Modelling Suite (SMS). Section §3.2 examines the next phase, i.e. change management during security design, using CARiSMA.

3.1 Change management between security risk assessment and system / software specification using the Thales prototype

According to the ATM risk assessment performed by SINTEF with CORAS[29], the "Unauthorised execution of APP services" stands out as the most severe unwanted incident (i.e. feared event), with respect to the "APP service provisioning" goal (i.e. primary asset), with an overall severity rated "High", and the respective severities on safety, business and reputation being high, medium and high. There are two threat scenarios leading to this unwanted incident: "External unauthorized actor gets access to APP control room" and "External unauthorized actor operates CWP in APP control room". The likelihood of those scenarios has been rated respectively "possible" and "rare". At this phase of the development lifecycle, the system is not yet designed, so

[29] Cf. D1.3 for more details.

the likelihood values are to be understood as security requirements to make the risks acceptable, given the severity of the unwanted incidents, i.e.:

- The likelihood of an external unauthorized actor getting access to APP control room ***shall*** be at most “possible”.
- The likelihood of external unauthorized actor operating the CWP in APP control room ***shall*** be at most “rare”.

These 2 requirements represent the main input for the system specification phase performed herein.

3.1.1 Step 1: System specification using SMS

The two aforementioned security requirements were considered in designing a staff physical arrival process using the Thales SOA Modelling Suite (SMS).

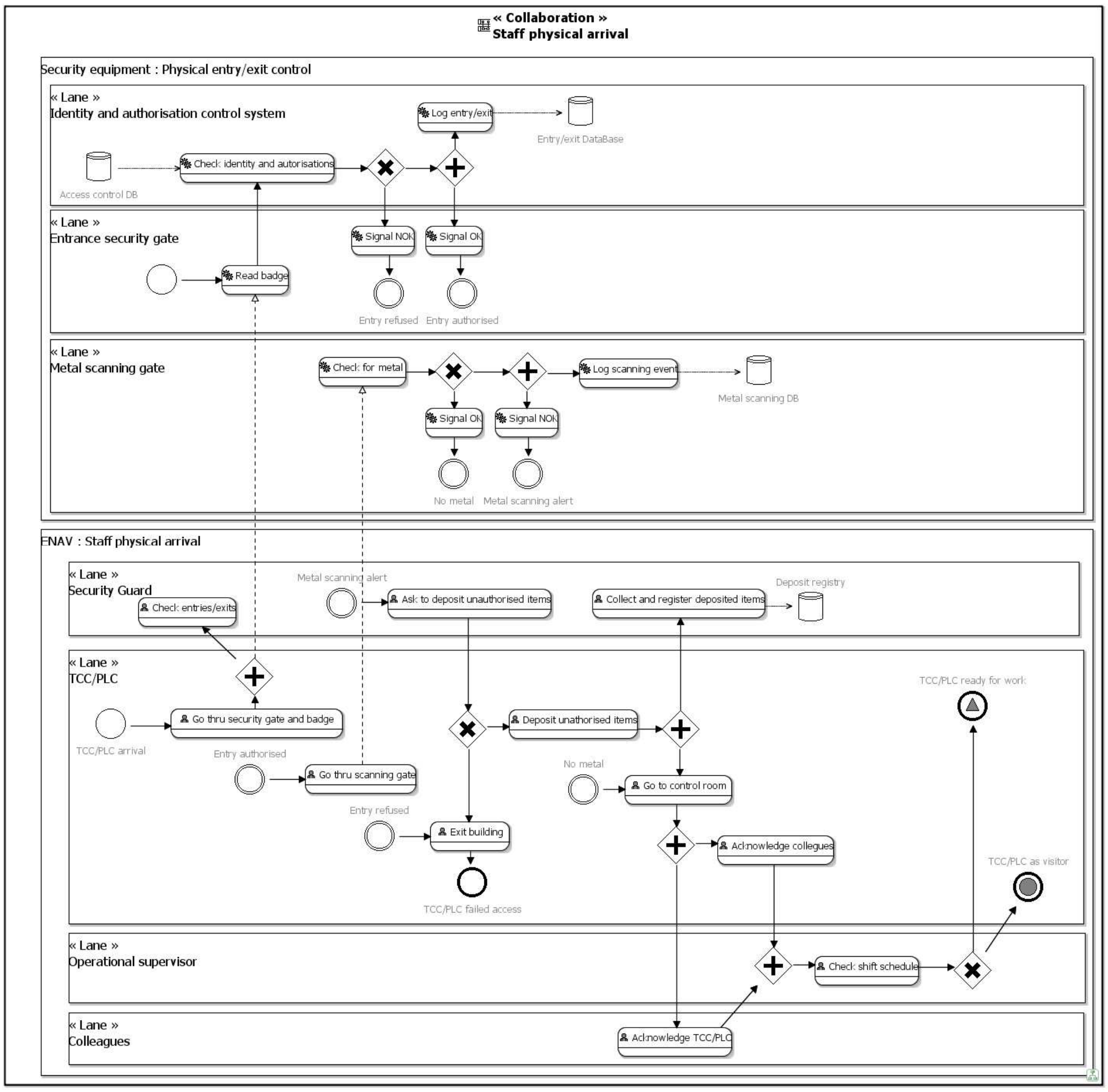


Figure 7: Staff physical arrival (collaboration)

Figure 7 shows a collaboration, according to the standard Business Process Modelling Notation (BPMN) 2.0 notation. It shows how tactical or planner controllers must go through a number of identity, authenticity and access control checks before being able to start their work in the APP control room. The process also includes a number of logging activities in order to support auditing.

Once the design activity is started, the security risk assessment activity can begin. The latter is described in §3.1.2.

3.1.2 Step 2: Security risk assessment using Rinforzando

The first activity[30] in risk assessment is to identify primary assets (called "essential elements" in the tool) within the SMS models. This is done using the "Essential Element(s)" tool in the "Import" palette (cf. Figure 42 on page 62). The complete list of objects that can be selected is shown in a window (cf. Figure 8).

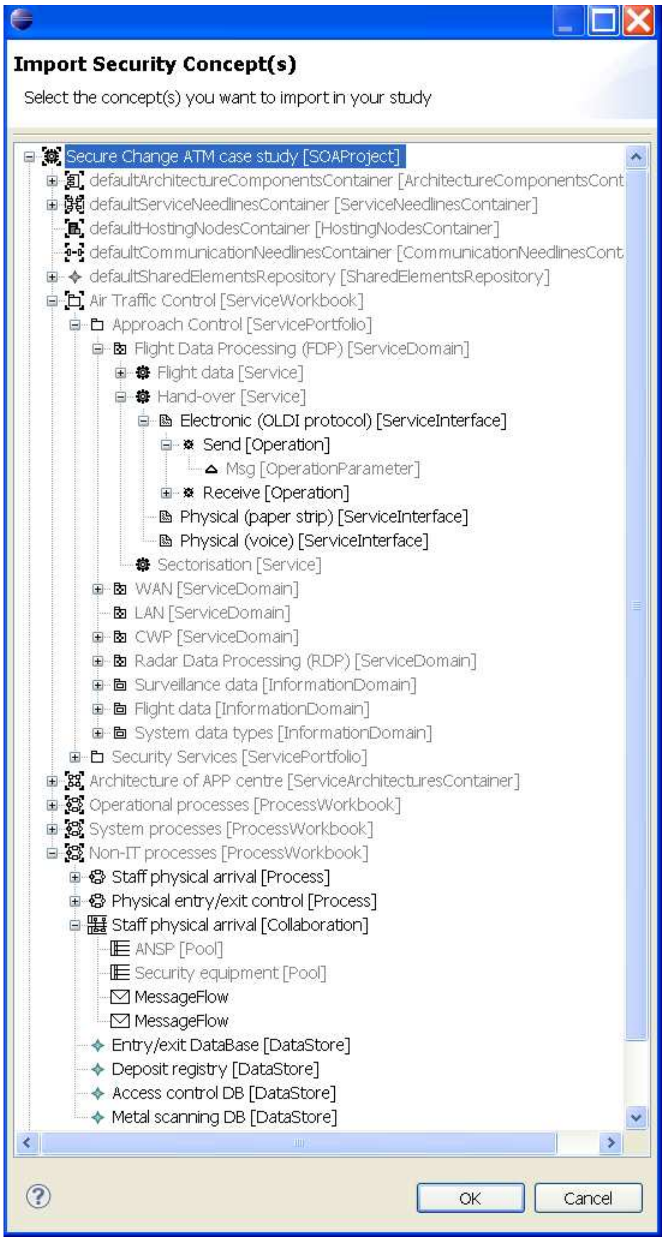
This window provides the security experts with a fine grained insight on the ongoing design work.

As can be seen in black writing on Figure 8, comprehensive artefacts such as the "electronic hand-over service" interface, the "staff physical arrival" collaboration or the "physical entry/exit control" process can be selected, as well as fine-grained artefacts, such as the "send" operation by the "hand-over" service or a "deposit registry" data store within the "staff physical arrival" collaboration.

The scope of artefact types that can be selected as primary assets has been configured offline in order to give a comprehensive choice without overwhelming the security experts with irrelevant artefacts, such as messages or pools.

Irrelevant artefacts are shown in grey so as to picture the overall structure of the system design. Indeed, there may be tens of operations called "send" within the complete design, but only one of them, e.g. the "send" operation of the "hand-over" service, might need to be selected as a primary asset. The grey elements allow for an understanding of the overall architecture and provide useful support in selecting the right modelling artefacts.

[30] A prerequisite to perform a risk assessment is that a new study is created and linked to the SMS models. Full details on this operation are given in §4.2.1.



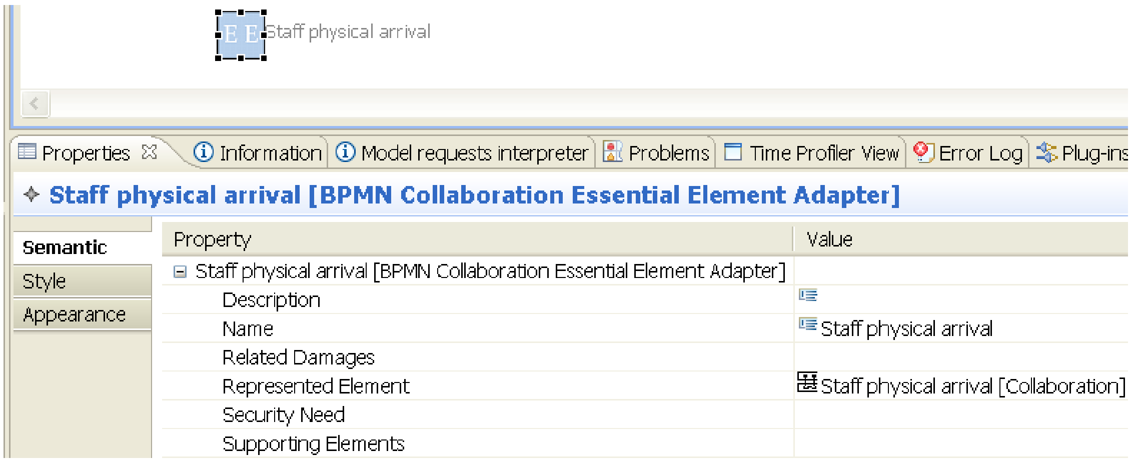
Import Security Concept(s)

Select the concept(s) you want to import in your study

```
Secure Change ATM case study [SOAProject]
  defaultArchitectureComponentsContainer [ArchitectureComponentsCont
  defaultServiceNeedlinesContainer [ServiceNeedlinesContainer]
  defaultHostingNodesContainer [HostingNodesContainer]
  defaultCommunicationNeedlinesContainer [CommunicationNeedlinesCont
  defaultSharedElementsRepository [SharedElementsRepository]
  Air Traffic Control [ServiceWorkbook]
    Approach Control [ServicePortfolio]
      Flight Data Processing (FDP) [ServiceDomain]
        Flight data [Service]
        Hand-over [Service]
          Electronic (OLDI protocol) [ServiceInterface]
            Send [Operation]
              Msg [OperationParameter]
            Receive [Operation]
          Physical (paper strip) [ServiceInterface]
          Physical (voice) [ServiceInterface]
        Sectorisation [Service]
      WAN [ServiceDomain]
      LAN [ServiceDomain]
      CWP [ServiceDomain]
      Radar Data Processing (RDP) [ServiceDomain]
      Surveillance data [InformationDomain]
      Flight data [InformationDomain]
      System data types [InformationDomain]
    Security Services [ServicePortfolio]
  Architecture of APP centre [ServiceArchitecturesContainer]
  Operational processes [ProcessWorkbook]
  System processes [ProcessWorkbook]
  Non-IT processes [ProcessWorkbook]
    Staff physical arrival [Process]
    Physical entry/exit control [Process]
    Staff physical arrival [Collaboration]
      ANSP [Pool]
      Security equipment [Pool]
      MessageFlow
      MessageFlow
    Entry/exit DataBase [DataStore]
    Deposit registry [DataStore]
    Access control DB [DataStore]
    Metal scanning DB [DataStore]
```

Figure 8: Importing essential elements from SMS (1/2)

Once the “staff physical arrival” collaboration is selected in the window, the primary asset associated with that collaboration appears, under the same name, within the risk assessment study, as shown in Figure 9. The link of the primary asset artefact to the collaboration artefact in the SMS study is visible in the “Properties” window.



Staff physical arrival

Properties | Information | Model requests interpreter | Problems | Time Profiler View | Error Log | Plug-ins

Staff physical arrival [BPMN Collaboration Essential Element Adapter]

Semantic | Style | Appearance

Property	Value
Staff physical arrival [BPMN Collaboration Essential Element Adapter]	
Description	
Name	Staff physical arrival
Related Damages	
Represented Element	Staff physical arrival [Collaboration]
Security Need	
Supporting Elements	

Figure 9: Creation of a primary asset within the risk assessment study

The selection and creation of this primary asset has multiple side-effects:

- Within SMS, it is now possible to see that the "staff physical arrival" collaboration has been selected as a primary asset in the risk assessment study; to this end, a "layers" menu was added in SMS during the installation of the risk assessment viewpoint. It allows for the selection of the desired annotations, as picture in Figure 10; once the desired layer has been selected, the artefact is discretely annotated, here in the upper left corner of the collaboration box. If these annotations are considered cumbersome, it is of course possible to hide all annotations by unselecting all layers.

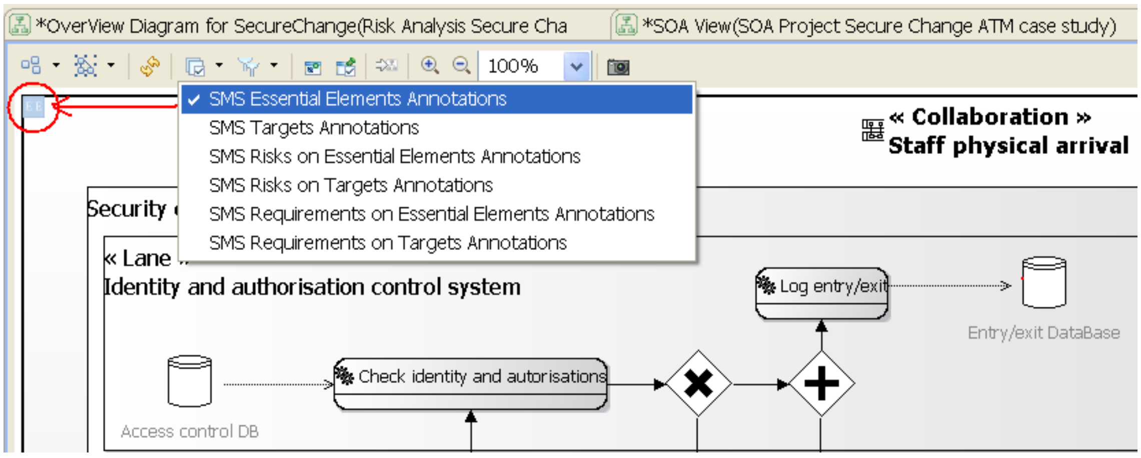

*OverView Diagram for SecureChange(Risk Analysis Secure Cha
*SOA View(SOA Project Secure Change ATM case study)
100%
✓ SMS Essential Elements Annotations
SMS Targets Annotations
SMS Risks on Essential Elements Annotations
SMS Risks on Targets Annotations
SMS Requirements on Essential Elements Annotations
SMS Requirements on Targets Annotations
« Collaboration »
Staff physical arrival
Security
« Lane
Identity and authorisation control system
Log entry/exit
Entry/exit DataBase
Check identity and autorisations
Access control DB

Figure 10: Adding risk assessment annotations in the mainstream system engineering design

- Within the risk assessment study, a new call to the "Essential Element(s)" tool in the "Import" palette (cf. Figure 42 on page 62) will show a list of objects that can be selected as primary asset within the SMS study, in which the already selected item appears in blue. In our case here, the staff physical arrival" collaboration is now written in blue, cf. Figure 11.

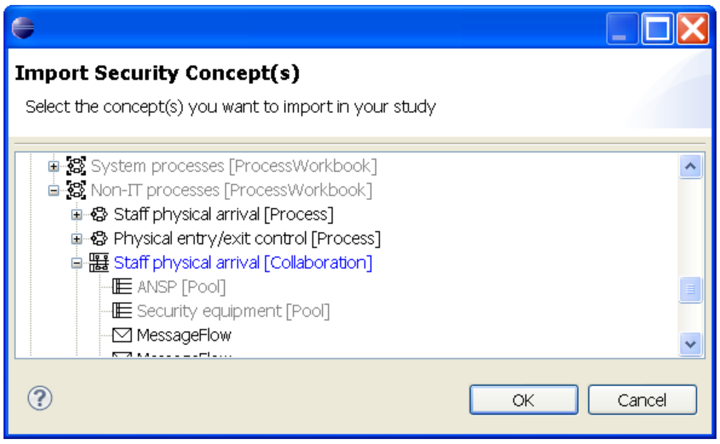

Import Security Concept(s)
Select the concept(s) you want to import in your study
System processes [ProcessWorkbook]
Non-IT processes [ProcessWorkbook]
Staff physical arrival [Process]
Physical entry/exit control [Process]
Staff physical arrival [Collaboration]
ANSP [Pool]
Security equipment [Pool]
MessageFlow
OK
Cancel

Figure 11: Importing essential elements from SMS (2/2)

The same process can be repeated for the supporting assets, as shown in Figure 12. The colour coding may seem trivial, but it is of high utility. Under the hypothesis that the mainstream system engineering activity is ongoing in parallel to the risk assessment, it is highly likely that new elements will be created in SMS after the initial analysis performed by the security expert. In that case, a quick scan through the elements in the window will show in black all the elements which have not yet been selected, thus saving a tedious comparison between the two models.

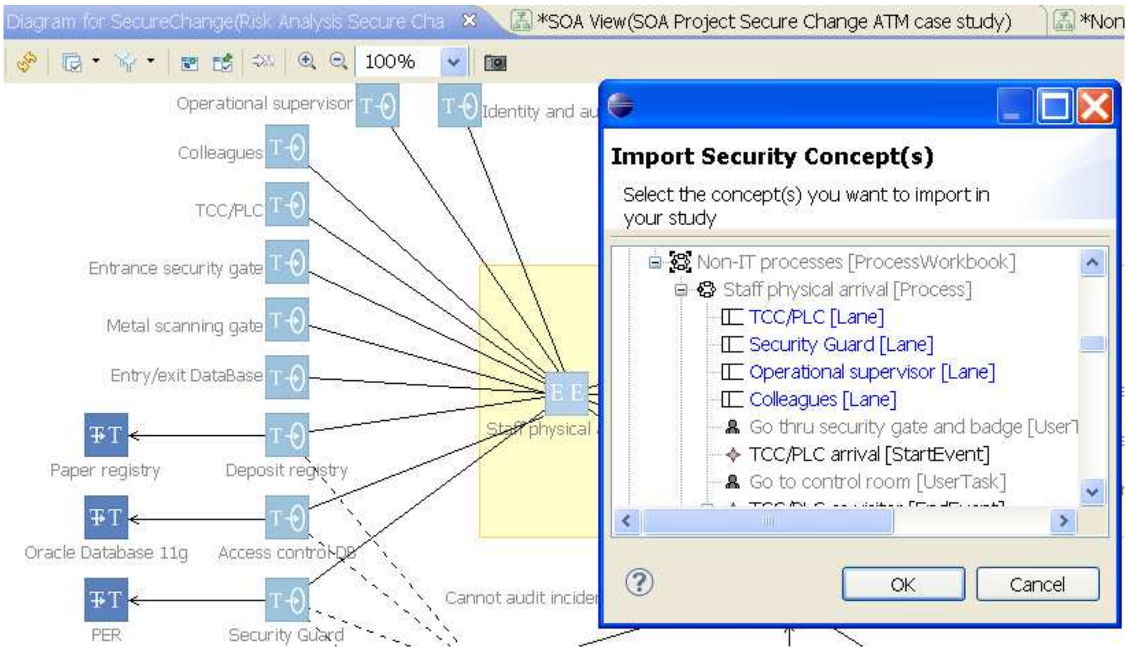


Diagram for SecureChange(Risk Analysis Secure Cha
*SOA View(SOA Project Secure Change ATM case study)
*Non
100%
Operational supervisor
Identity and au
Colleagues
TCC/PLC
Entrance security gate
Metal scanning gate
Entry/exit DataBase
Paper registry
Deposit registry
Oracle Database 11g
Access control DB
PER
Security Guard
Staff physical
Cannot audit incide
Import Security Concept(s)
Select the concept(s) you want to import in your study
Non-IT processes [ProcessWorkbook]
Staff physical arrival [Process]
TCC/PLC [Lane]
Security Guard [Lane]
Operational supervisor [Lane]
Colleagues [Lane]
Go thru security gate and badge [User
TCC/PLC arrival [StartEvent]
Go to control room [UserTask]
OK
Cancel

Figure 12: Importing targets from SMS

Change management support to the risk assessment is also provided in case existing and already selected elements are changed in SMS. Let is for example focus on the "security guard" that has been selected as a supporting asset to the "staff physical arrival" collaboration, as pictured in Figure 12. To stress the fact that the security guard position will not be sub-contracted, the design chooses to rename the “Security Guard” lane into “ANSP[31] In-House Security Guard”, as shown in Figure 13.

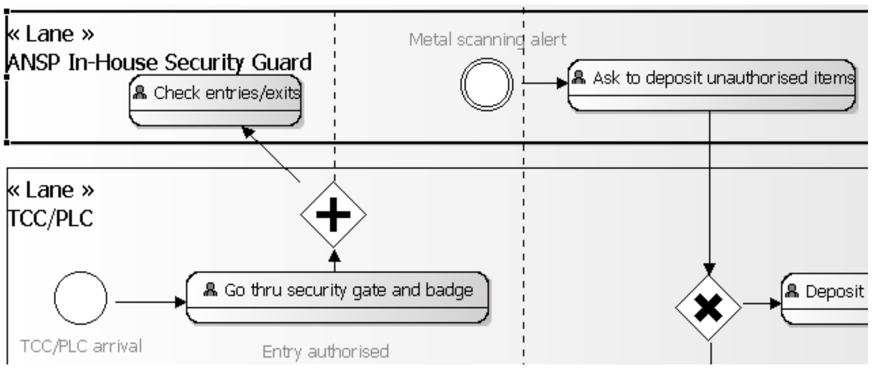


Figure 13: Renaming the “Security Guard” lane into “ANSP In-House Security Guard”

As soon as the BPMN collaboration diagram is saved, the change is automatically and transparently reflected in the risk assessment study, as shown in Figure 14.

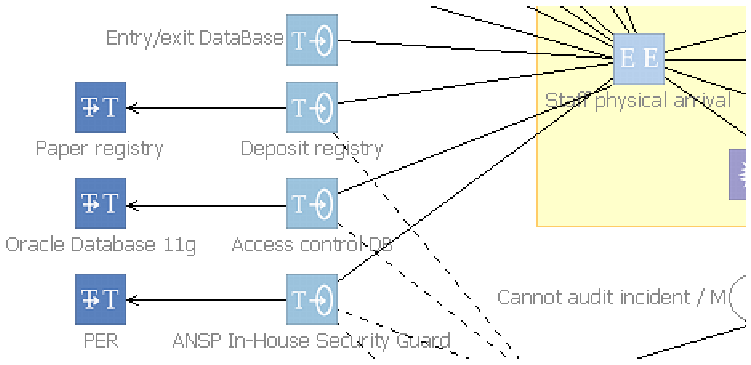


Figure 14: Automatic and transparent renaming from SMS to Rinforzando

It is to be noted that the change is not automatically and transparently reflected in the risk assessment study if the name was originally changed in the risk assessment study. Let us assume that the question about the sub-contracting of the security guard position was first discussed by the security experts with the customer, and it was confirmed that the security guard would be a member of the ANSP staff. To raise the ambiguity, the name of the supporting asset was changed in the risk assessment study, as shown in Figure 15. This change has no effect on the SMS models.

[31] Air Navigation Service Provider.

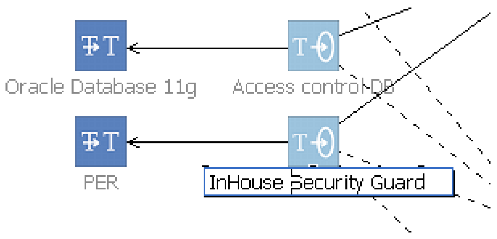
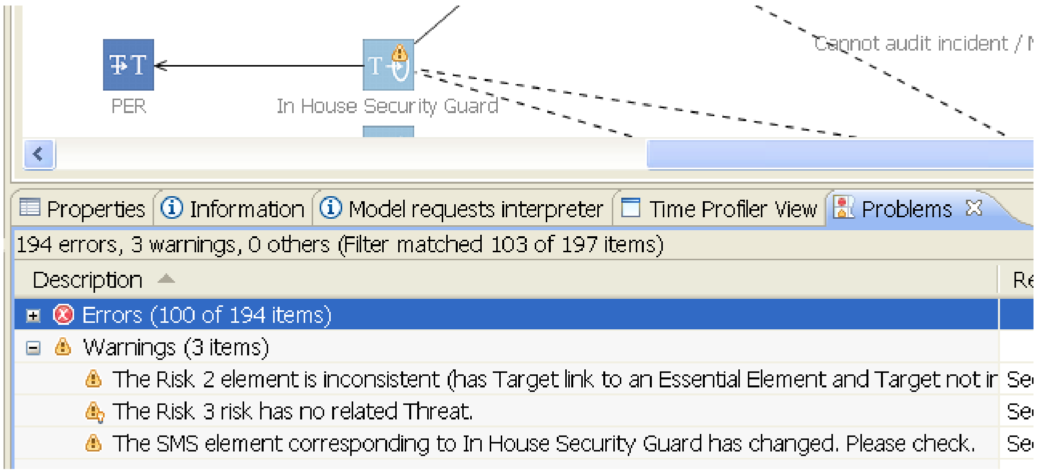


Figure 15: Changing the name of a supporting asset in the risk assessment study

If the designers now change the name, as shown in Figure 13, there will be no change in the risk assessment study, as this would mean the loss of some work performed by the security expert. However, this change should not go unnoticed by the security expert. This is implemented through a warning which is raised during a diagram validation, as shown in Figure 16.



```
194 errors, 3 warnings, 0 others (Filter matched 103 of 197 items)
Description                                                                                 Re
Errors (100 of 194 items)
Warnings (3 items)
    The Risk 2 element is inconsistent (has Target link to an Essential Element and Target not ir   Se
    The Risk 3 risk has no related Threat.                                                       Se
    The SMS element corresponding to In House Security Guard has changed. Please check.          Se
```

Figure 16: Diagram validation and warning (1/3)

A similar change management support is provided by Rinforzando when an existing and already selected element is deleted in SMS. Lets us suppose here that for cost reasons, the logging function is suppressed from the “identity and authorisation control system” during a design review, as pictured in Figure 17 and Figure 18.

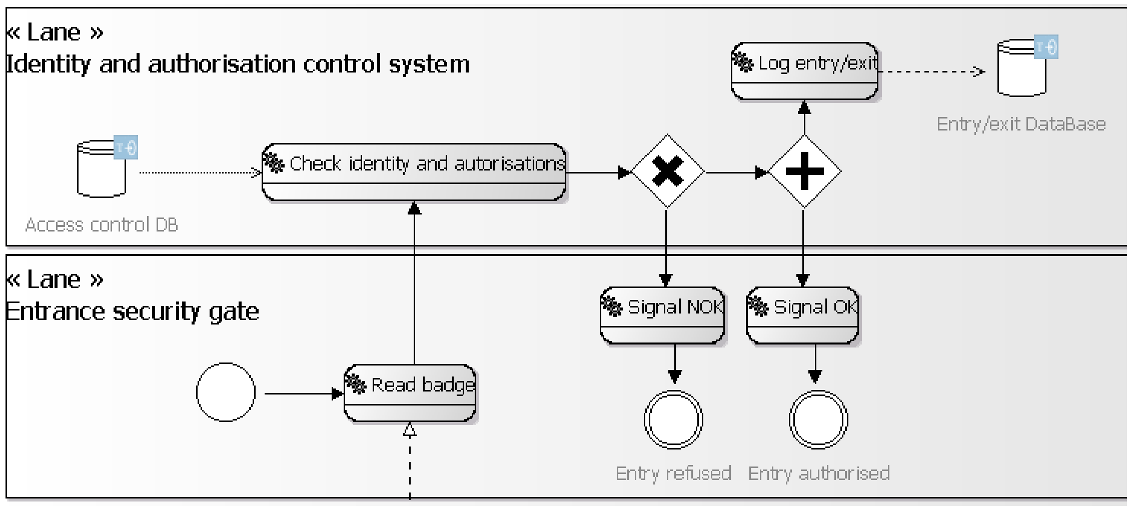


Figure 17: Identity and authorisation control system, before cost reduction, and its interactions with the entrance security gate (with risk assessment annotations)

During this cost reduction operation, the "entry/exit database" is also deleted from the SMS model, not withstanding the fact that it had been identified as a supporting asset during the risk assessment (cf. Figure 11). It is important to note that the fact that it had been selected as a supporting asset was also visible from the SMS side, through the target annotation on the database (cf. Figure 17).

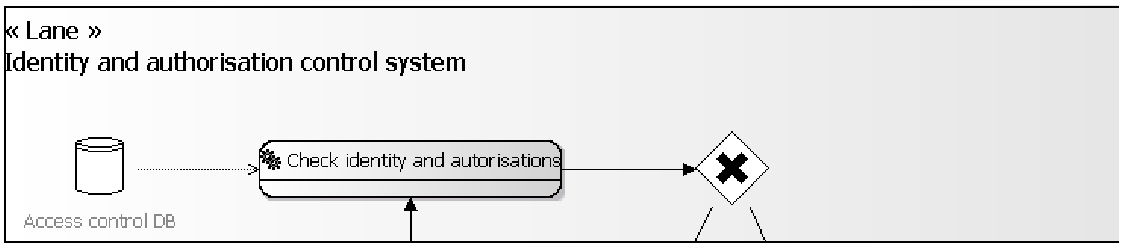
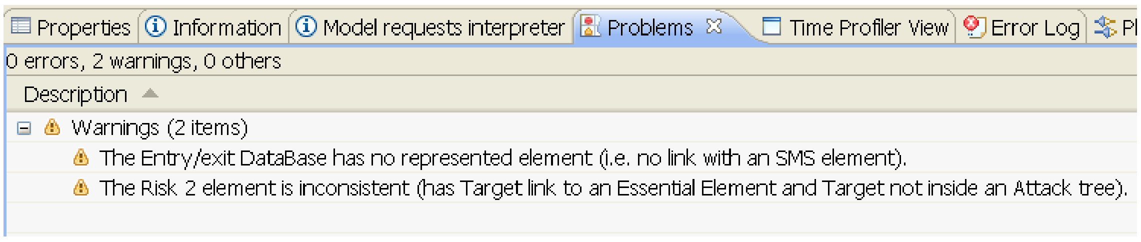

« Lane »
Identity and authorisation control system
Access control DB
Check identity and autorisations

Figure 18: Identity and authorisation control system, after cost reduction (without risk assessment annotations)

Again, there will be no automatic or transparent change in the risk assessment study following this deletion, as this would mean the loss of some work performed by the security expert and because the impact on the risk assessment is extremely complex, in terms of threat likelihood, risk level and/or compromised security goals. The change notification is once again implemented through a warning which is raised during a diagram validation, as shown in the "Problems" window (cf. Figure 19) and graphically directly on the related item (cf. Figure 20). The security expert will then be in a position to process the change manually.



0 errors, 2 warnings, 0 others

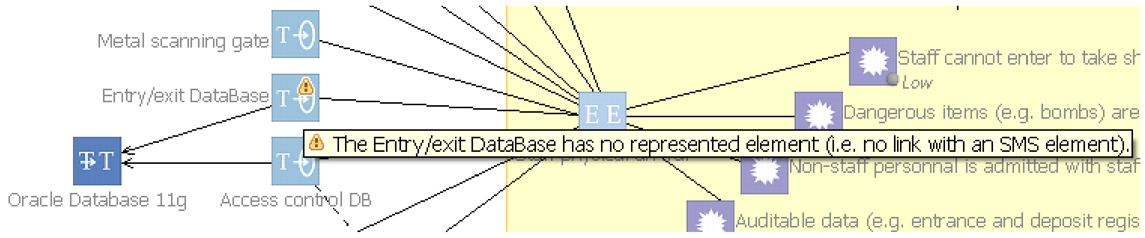
Description

Warnings (2 items)

The Entry/exit DataBase has no represented element (i.e. no link with an SMS element).

The Risk 2 element is inconsistent (has Target link to an Essential Element and Target not inside an Attack tree).

Figure 19: Diagram validation and warning (2/3)


Metal scanning gate
Entry/exit DataBase
Oracle Database 11g
Access control DB
E E
Staff cannot enter to take sh
Low
Dangerous items (e.g. bombs) are
Non-staff personnal is admitted with staf
Auditable data (e.g. entrance and deposit regis

The Entry/exit DataBase has no represented element (i.e. no link with an SMS element).

Figure 20: Diagram validation and warning (3/3)

3.2 Evaluation of change management between early validation of security properties and system / software design using UML and CARiSMA

CARiSMA is a tool developed by TUD during the second and third years of SecureChange supporting analyses for the UMLsec and UMLchange UML 2 profiles. Those profiles are dedicated to, respectively, security and change modelling. UMLsec's stereotypes and tags annotate the elements of a UML 2 model with security-related information that is then used by CARiSMA to carry out security analyses. UMLchange's stereotypes and tags represent changes (additions, deletions and substitutions) and can be applied to a UML 2 model to denote the evolution of a model. Some of CARiSMA's security analyses have evolution-aware variants supporting UMLchange annotations that are used to verify that the described change does not invalidate the security property.

CARiSMA is the successor to the legacy UMLsec tool, which performed analyses for the legacy UML 1.x UMLsec profile, but had no support for evolution.

This section presents the results of Thales' evaluation of the tool. CARiSMA consists of a set of plug-ins for the Eclipse platform distributed as an update site. Once installed, the tool integrates into Eclipse's UML2 framework by:

- making the UMLsec and UMLchange profiles available so that they can be applied to UML 2 models;
- allowing analyses to be performed on profiled UML2 models.

We used Papyrus UML as our main modelling tool. Papyrus unfortunately uses a specific mechanism to register profiles that CARiSMA does not make use of, preventing a profile to be applied directly from Papyrus. This is easily worked around, however: the user can simply apply the profile(s) in Eclipse's basic UML editor, and then stereotypes can be added to individual model elements directly from Papyrus.

CARiSMA's user interface is simple and intuitive, and includes a help section integrated into Eclipse's help system. It is quite easy for a user to set up and run an analysis. A dedicated view displays the synthetic results of analyses, and more details can be obtained by generating text reports.

We identified several areas of possible improvement.

1. CARiSMA's implementation of the checks is not specified; the best documentation source is the UMLsec book, which details the semantics of UMLsec for UML 1.x. The following information would be useful:
   a. the exact impact on each UMLsec constraints caused by the move to UML2;
   b. any deviation from these updated semantics in the implementation in CARiSMA.
2. In terms of user interface, the integration with Eclipse could be pushed further, in particular with the EMF Validation framework; one of the benefits would be the use of the standard Problems view for reporting analysis results instead of a specific one.
3. The implementation of all the checks we tested seemed to be sequential, even when independent sub-checks could be treated in parallel (e.g. separate path in fair exchange). As a matter of fact, the checks seem to be run in Eclipse's UI thread, with the consequence that the UI is completely frozen during long computations and no progress feedback is given. This is incompatible with any non-trivial check.
4. There is no alternative to CARiSMA's point-and-click interface; in particular no public API is provided to call analyses programmatically. This is a major issue if the tool is to be integrated in a larger industrial environment.

3.2.1.1 Structural checks

CARiSMA supports UMLsec's "secure dependency" and "secure links" analyses. Those are static analyses in the sense that their associated constraint expresses a structural constraint on the model, such as those that are typically implemented in OCL or Java.

Those analyses have a limited use because they are highly specific. We do not provide examples for their use since they were not relevant to our use cases. However, many similar constraints usually complement meta-models and are part of any industrial speciality engineering. Our main remark is that we could not find any reason to use CARiSMA for such checks, when Eclipse makes it easy to add equivalent OCL or Java constraints with the added benefit of a full integration with the EMF Validation framework, including the Problems view, tooltips, and quick fixes.

3.2.1.2 Benchmarks for the fair exchange check

At the time of this writing, the only general-purpose dynamic check supported by CARiSMA is fair exchange. As described in section 2.3.1, the fair exchange check can express generic liveliness properties. No evolution aware variant is available; it would have been useful to evaluate the benefit of an explicit representation of change for a computationally extensive check such as fair exchange.

Liveliness is one of the key properties analysed by model checkers. A model checker explores a description of a system (usually in the form of a graph structure of states and transitions) exhaustively and checks that all possible executions of the system verify a given property. The major issue model checkers face is the combinatorial explosion of the number of states / execution paths to consider. A number of techniques have been developed in the last 20 years to alleviate this problem.

We ran three distinct benchmarks to estimate how CARiSMA's implementation of fair exchange copes with this combinatorial explosion. The hardware used for the benchmark was an Intel Core 2 Duo T9600 CPU clocked at 2.80 GHz, and 3.48 GB of RAM.

Long parallel paths

We developed an activity diagram from an industrial process in which two actors perform a few actions independently before synchronising. In doing so, we ran into performance issues with CARiSMA. We isolated the part of the diagram responsible for these and obtained the simpler diagram illustrated in Figure 21.

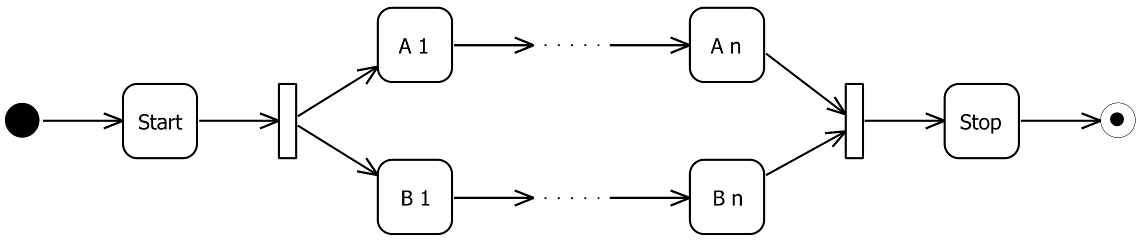


Figure 21 - Activity diagram for the long parallel paths benchmark

The path goes from the initial node to an initial action Start marking the first synchronisation point of the two actors, to two parallel paths of n actions each representing the actions of each actor on its own path, to a final action Stop marking the second synchronisation point of the two actors, to the final node. We ran the check with {start = Start} and {stop = Stop} for various values of n and measured the time the check took to complete; the results are given in Table 1.

Value of n	**Value of $\binom{2n}{n}$**	**Time**	**Paths / s**
6	924	2 s	462
7	3432	24 s	143
8	12870	6 min 17 s	34
9	48620	1 h 40 min 30 s	8

Table 1 - Timing results for the long parallel paths benchmark

These results show that the execution time of CARiSMA's implementation of fair exchange grows extremely quickly with the length of the paths. This could be caused by the interleaving semantics of activity diagrams in UMLsec. UMLsec considers sequential execution paths, and all the possible interleavings of parallel paths must be considered in the verification. In our example, if $n = 2$, the 6 possible interleavings are:

A1, A2, B1, B2	B1, A1, A2, B2
A1, B1, A2, B2	B1, A1, B2, A2
A1, B1, B2, A2	B1, B2, A1, A2

In other words, an interleaving of the two parallel paths is essentially a permutation of the $2n$ actions such that the order of the actions in each parallel path is respected. The number of interleavings (and, consequently, of paths to check) is therefore the binomial coefficient $\binom{2n}{n}$. We conjecture this to be the cause for the high running time. However the running time grows significantly quicker than $\binom{2n}{n}$; the increasing length of the paths seems unlikely to account for that effect, which we suspect might be attributed to memory constraints (e.g. swapping).

In any case, this benchmark shows that the check can only accommodate small numbers of actions in parallel paths. It does not support the verification of processes involving independent paths with a large number (i.e. around 10) of actions at any point. This is unfortunately not uncommon in an industrial environment.

Parallelisation

Following the issues raised in the previous benchmark, we developed another benchmark to evaluate the check's sensitivity to parallelism. We considered the family of activity diagrams illustrated in Figure 22.

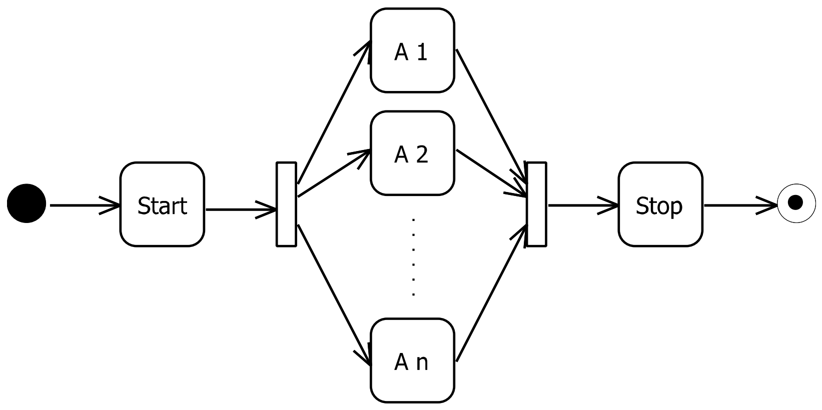


Figure 22 - Activity diagrams for the parallelisation benchmark

A diagram consists of a path from an initial node to an initial action Start, to a group of n actions executed in parallel, to a final action Stop, to a final node. We ran the fair exchange check with {start = Start} and {stop = Stop} for several values of n and measured the time the check took to complete. Table 2 presents the results.

Value of n	Value of $n!$	Time	Paths / s
5	120	1 s	-
6	720	2 s	360
7	5040	34 s	148
8	40320	39 min	17

Table 2 - Timing results for the parallelisation benchmark

The timing results show once again that the running time of the check grows extremely quickly, confirming our hypothesis as to the sensitivity of the check to parallel paths. In our example, an interleaving of the n parallel paths is essentially a permutation of the n actions. There are therefore $n!$ paths to check. Again, the running time grows much quicker than the number of paths.

The check does not support the verification of processes involving a large number (i.e. around 10) of actors performing concurrent actions at any point. Although one does not always handle such cases, this is an important limitation to be aware of in an industrial environment.

More generally, the tool can clearly only accommodate a small amount of parallelism, being limited to handling a few short parallel paths.

Independent choices

In a third benchmark, we explored another source of combinatorial explosion in the number of paths unrelated to parallelism. This time we used independent decision / merge patterns chained sequentially as illustrated in Figure 23.

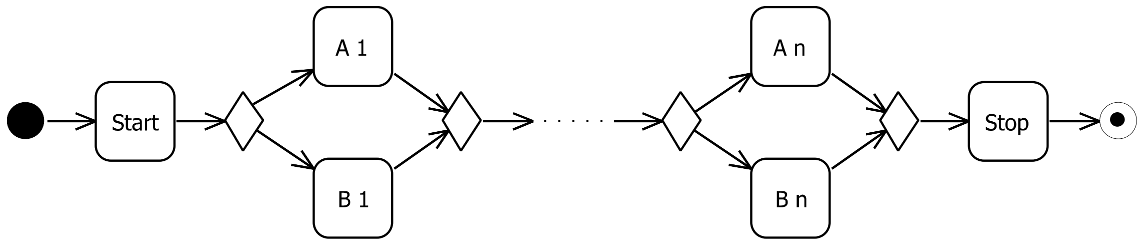


Figure 23 - Activity diagrams for the independent choices benchmark

A diagram consists of a path from an initial node to an initial action Start, to a chain of n patterns, each of which consisting of two disjunctive branches containing each one action, to a final action Stop, to a final node. We ran the fair exchange check with {start = Start} and {stop = Stop} for several values of n and measured the time the check took to complete. Table 3 presents the results.

Value of n	**Value of 2^n**	**Time**	**Paths / s**
10	1024	3 s	341
11	2048	10 s	205
12	4096	36 s	114
13	8192	3 min 40 s	37
14	16384	11 min	25
15	32768	51 min	11

Table 3 - Timing results for the independent choices benchmark

The timing results show that the running time of the check grows quickly, certainly quicker than the number of paths 2^n, but seemingly roughly exponentially.

The check does not support the verification of processes involving a quite large number (i.e. around 15) of independent choices. This is a limitation to be aware of, but the typical models manipulated in an industrial environment at a high level of description (process) do not contain that many choices. Besides, this limitation is more likely to be expected from a verification tool than the limitation on parallelism.

Concluding remarks

Although the fair exchange analysis seems to be useful as a generic check, its implementation in CARiSMA suffers from the classic state space explosion problem. The benchmarks we provided demonstrated the problem, and such cases are not uncommon in industrial models. For example in the ATM domain modelling the detailed work of the ATCs in a control tower involves 5 or 6 parallel paths (one per controller) with sequences of about 10 actions. The mere presence in an activity diagram of patterns such as the ones exposed in the benchmarks causes extremely high running times of the check, even if the fair exchange problem only involves parts of the diagram in which the patterns do not occur.

This issue, combined with the fact that any long computation freezes the user interface and requires the user to terminate the Eclipse instance losing any unsaved work, makes it unsuitable for industrial use in its current state.

Evolution aware checks currently implemented are structural only. However, the real benefit of an explicit representation of change, would be the support of computationally intensive behavioural checks such as fair exchange.

3.2.1.3 SWOT analysis

The following SWOT analysis is written from the point of view of an industrial organisation considering the adoption of CARiSMA.

Strengths

- The UMLsec language has received considerable attention over the years and is a reference in UML-based security verification.
- UMLsec addresses a fragment of UML for which formal semantics are given. Recent work on the language allowed it to be ported to UML 2.
- CARiSMA is integrated in the modern Eclipse EMF platform.

Weaknesses

- The formal semantics are defined in the UMLsec book but have not been explicitly ported to UML 2. The relationship with other semantics (such as the fUML standard) is not specified, but they are likely distinct.
- A few of the UMLsec checks have been ported and implemented in CARiSMA so far, but the tool is currently immature. A documentation of deviations from the semantics described in the book, if any, would be useful.
- The integration with Eclipse is partial. While Eclipse's UML 2 framework is supported, there is no integration with EMF Validation.
- There is no clear advantage in using CARiSMA's structural checks over OCL-like constraints supported by EMF Validation. CARiSMA's only behavioural check, i.e. fair exchange, suffers from performance limitations.

Opportunities

- UMLsec's analyses are helpful in performing early security validation on a system under development.

Threats

- While UML is a standard for modelling, the same is not true of UMLsec and UMLchange.
- There is no clear public roadmap for the future development of UMLsec and CARiSMA in terms of direction or amount of involved resources.
- There are no noticeable large scale industrial deployments of UMLsec, UMLchange, and CARiSMA.

Glossary and acronyms

Asset Definition

An asset is any resource that has value for the organism and which is necessary for the accomplishment of its goals. It is possible to distinguish primary assets and supporting assets.

CARiSMA Definition

CARiSMA is a tool framework for compliance, risk, and security analysis of software models. It is a reimplemented variant of the former UMLsec tool (www.umlsec.de) and has a plug-in architecture that makes it extensible for different modelling languages (including but not limited to UML2 and BPMN) and a broad range of analyses. CARiSMA is integrated into Eclipse and can thus be integrated into many modelling tools, such as TOPCASED or Papyrus MDT.

Within SecureChange the tool has been extended with evolution support. Possible evolutions of a software model can be expressed using the UMLchange notation and based thereon security properties can be checked on all evolution paths.

Essential Element (EE) Definition

Rinforzando's name for primary asset.

Primary Asset Definition

A primary asset is information or a process deemed important by the organisation. The system security risk manager assesses its security needs, but not its vulnerabilities.

Rinforzando Definition

Rinforzando is a Thales security risk assessment modelling tool that is both tightly and loosely integrated with the Thales SOA Modelling Suite (SMS), a tool suite which is representative, in terms of technical foundations, of the Thales corporate mainstream system engineering workbench. It is loosely integrated because Rinforzando can easily be used as a standalone risk assessment tool. It is tightly integrated because it can directly reference artefacts within the mainstream system engineering models and ensure the consistency of the risk assessment model with the mainstream system engineering models, irrespectively of the phase. In SecureChange, Rinforzando was used during the design phase, but it can also be used during user need analysis, detailed design, deployment, operation & maintenance or disposal.

SOA Modelling Suite (SMS) Definition

The purpose of the Thales SOA Modelling Suite is to capture the different concerns related to service oriented architecture specifications and implementations. SMS is a

Thales Research & Technology in-house prototype on which work is still on-going. Ultimately, it shall provide its users with domain specific languages (DSLs), as well as their corresponding graphical representations, that allow specifying efficiently SOA concerns. Such DSLs are designed by capturing the concepts associated with SOA standards, technologies and Thales engineers' specific requirements. The Thales SOA Modelling Suite already comprises some code-level WS-security policy generation capabilities.

Within the SecureChange project, the Thales Risk Assessment Domain Specific Modelling Language, implemented in the Rinforzando tool, has been integrated with SMS. The integrated result has been used to model part of the ATM case-study (WP1).

Supporting Asset Definition

A supporting asset is an asset supporting primary assets, e.g. information systems, organisations, premises. The system security risk manager assesses its vulnerabilities, but not its security needs.

Target Definition

Rinforzando's name for supporting asset.

References

[1] Fabio Massacci, Fabrice Bouquet, Elizabeta Fourneret, Jan Jurjens, Mass S. Lund, Sébastien Madelénat, Jan Tobias Muehlberg, Federica Paci, Stéphane Paul, Frank Piessens, Bjørnar Solhaug, Sven Wenzel, *Orchestrating Security and System Engineering for Evolving Systems*, Security, Privacy and Trust Scientific Track, 4th European Conference ServiceWave'11, Poznan, Poland, 27 October 2011.

[2] *Arcadia method for architecting & design in system / software definition V4.0* – Reference Manual, J.-L. Voirin, X. Le Roux, ref. DAE001966-00, 1 September 2008 [Thales confidential in industry].

[3] *System-EM handbook*, The Thales system engineering methodology, version 2.0, ref. 87100098/E/C/09, December 2009 [Thales confidential in industry].

[4] *Systems and software engineering -- Architecture description,* ISO/IEC/IEEE 42010:2011,

[5] *Harmonized Threat and Risk Assessment (TRA) Methodology*, Communications Security Establishment (CSE) Canada, October 23, 2007.

[6] Vikash Katta, Tor Stalhåne, *A conceptual model of traceability for safety systems,* Poster session, 2nd International Conference on Complex Systems Design & Management (CSDM), December 7-9, 2011, Paris, France.

4 Appendices

4.1 Thales prototype installation manual

This section deals with the installation of the Risk Assessment viewpoint components.

The Risk Assessment viewpoint works on the Thales SMS 3.1 environment. The first step is to launch this environment. Once the environment has been launched:

- Click Help > Install New Software...

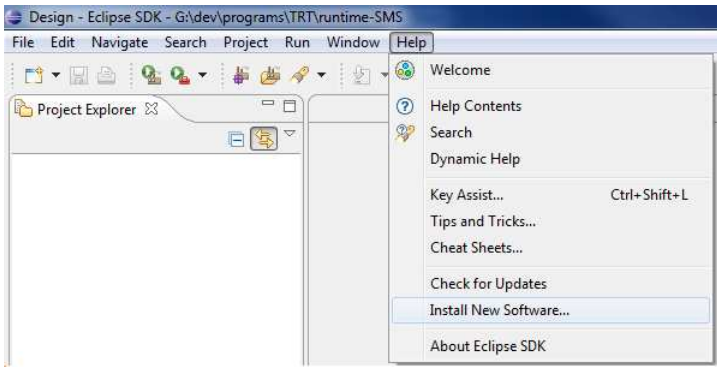
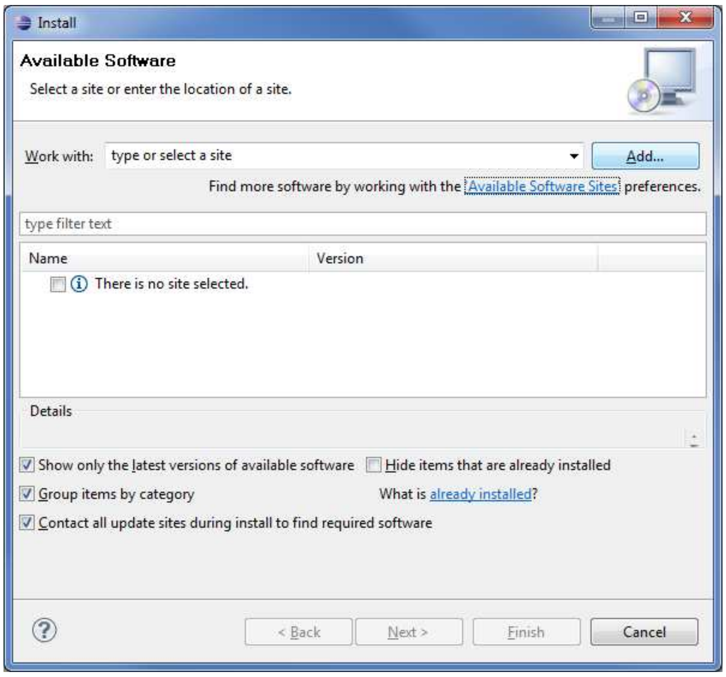


Figure 24: Help menu

- A “Install” window opens. Click on the *Add* button.



Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site | Add...

Find more software by working with the Available Software Sites preferences.

type filter text

Name	Version
☐ There is no site selected.	

Details

☑ Show only the latest versions of available software ☐ Hide items that are already installed

☑ Group items by category What is already installed?

☑ Contact all update sites during install to find required software

< Back | Next > | Finish | Cancel

Figure 25: Install window (1/3)

- An “Add Site” window opens. Select the archive file that contains the Risk Assessment viewpoint, by clicking on *Archive* button. Then enter a name for that repository.

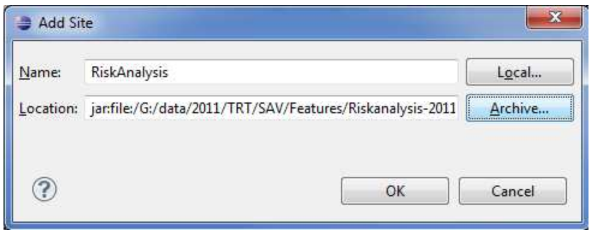

Add Site
Name: RiskAnalysis
Location: jar:file:/G:/data/2011/TRT/SAV/Features/Riskanalysis-2011
Local...
Archive...
OK
Cancel

Figure 26: Add Site window

- The “Install” window shows the feature contained in the archive file. Check Risk analysis f*eature*, and unchecked *Group items by category* option. Click *Next >*.

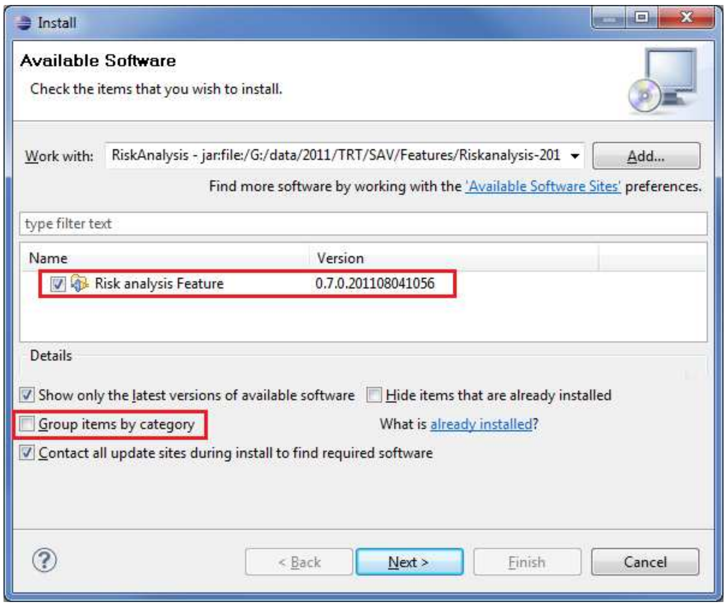
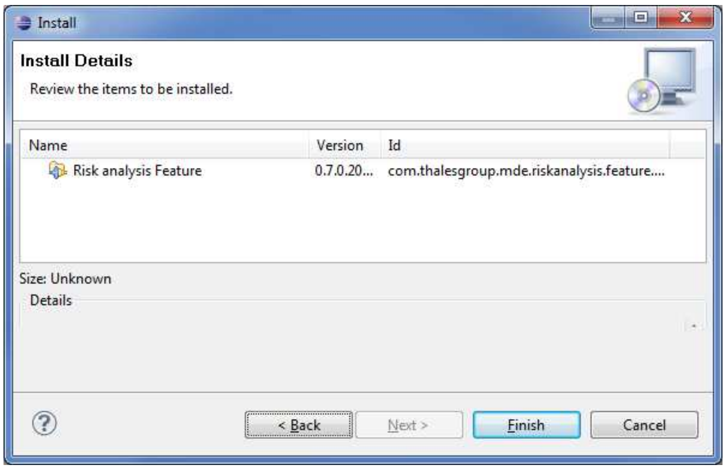

Install
Available Software
Check the items that you wish to install.
Work with: RiskAnalysis - jar:file:/G:/data/2011/TRT/SAV/Features/Riskanalysis-201
Add...
Find more software by working with the 'Available Software Sites' preferences.
type filter text
Name
Version
Risk analysis Feature
0.7.0.201108041056
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel

Figure 27: Install window (2/3)

- Click Finish and follow the instructions. Restart the environment after completing the installation.


Install

Install Details

Review the items to be installed.

Name	Version	Id
Risk analysis Feature	0.7.0.20...	com.thalesgroup.mde.riskanalysis.feature....

Size: Unknown

Details

Figure 28: Install window (3/3)

The Risk Assessment viewpoint is now installed on the Thales SMS 3.1 environment.

4.2 Thales prototype operator handbook (extract)

The Thales risk assessment tool is a complex and complete tool. The objective here is not to provide the complete operator handbook to support a risk assessment study.

The Thales risk assessment tool is designed to be used in conjunction with the Thales SOA Modelling Suite (SMS), but it can also be used as a stand-alone tool.

In line with the scope of this deliverable, this section only describes the parts of the RA tool that are specific to its use in conjunction with SMS. However, a flavour of the complete tool is given in Figure 29.

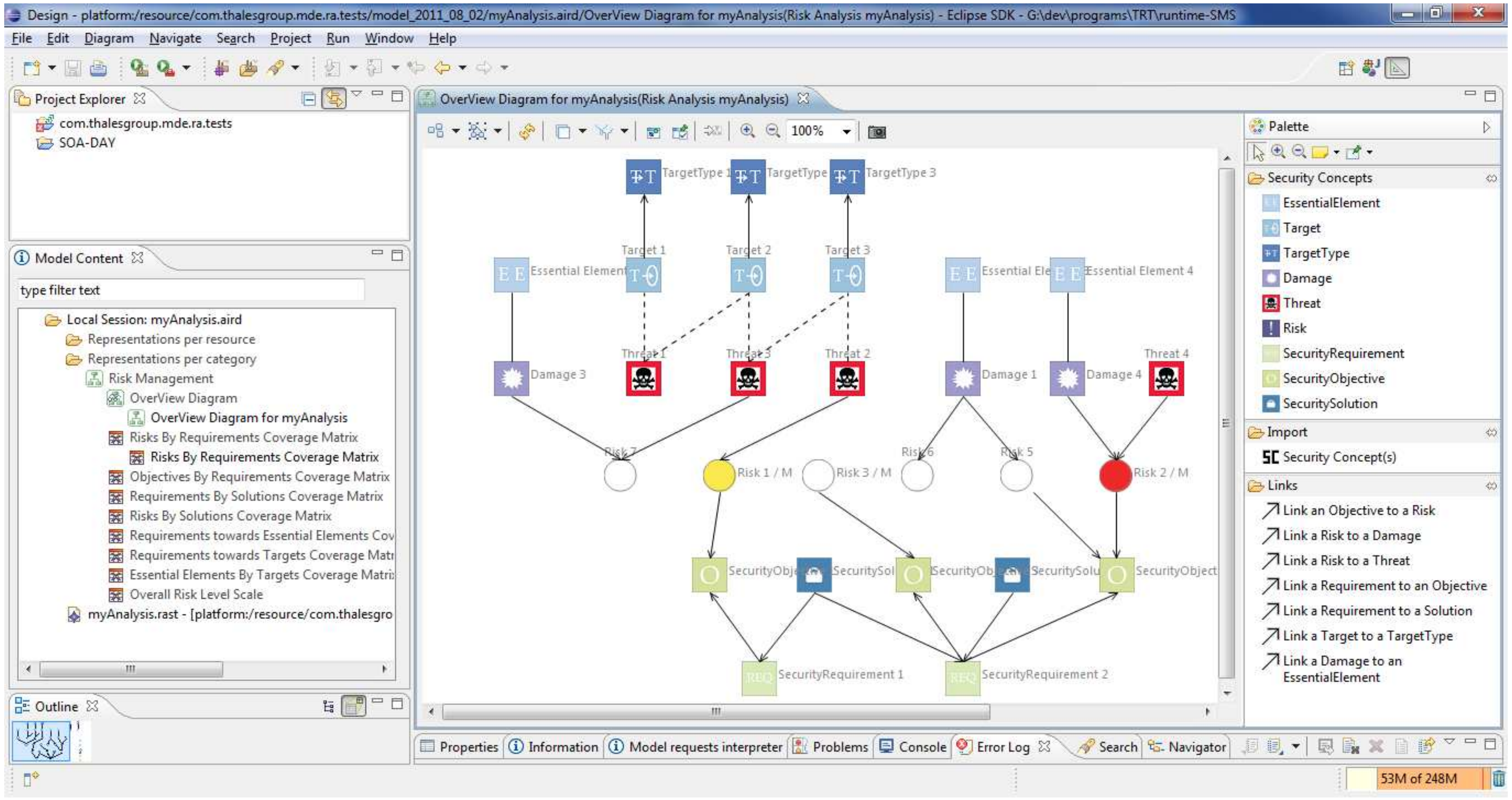

Design - platform:/resource/com.thalesgroup.mde.ra.tests/model_2011_08_02/myAnalysis.aird/OverView Diagram for myAnalysis(Risk Analysis myAnalysis) - Eclipse SDK - G:\dev\programs\TRT\runtime-SMS
File Edit Diagram Navigate Search Project Run Window Help
Project Explorer
com.thalesgroup.mde.ra.tests
SOA-DAY
Model Content
type filter text
Local Session: myAnalysis.aird
Representations per resource
Representations per category
Risk Management
OverView Diagram
OverView Diagram for myAnalysis
Risks By Requirements Coverage Matrix
Risks By Requirements Coverage Matrix
Objectives By Requirements Coverage Matrix
Requirements By Solutions Coverage Matrix
Risks By Solutions Coverage Matrix
Requirements towards Essential Elements Cov
Requirements towards Targets Coverage Matr
Essential Elements By Targets Coverage Matri
Overall Risk Level Scale
myAnalysis.rast - [platform:/resource/com.thalesgro
Outline
OverView Diagram for myAnalysis(Risk Analysis myAnalysis)
100%
TargetType 1
TargetType
TargetType 3
Target 1
Target 2
Target 3
Essential Element
Essential Ele
Essential Element 4
Threat 1
Threat 3
Threat 2
Threat 4
Damage 3
Damage 1
Damage 4
Risk 7
Risk 1 / M
Risk 3 / M
Risk 6
Risk 5
Risk 2 / M
SecurityObj
SecuritySol
SecurityOb
SecuritySolu
SecurityObject
SecurityRequirement 1
SecurityRequirement 2
Palette
Security Concepts
EssentialElement
Target
TargetType
Damage
Threat
Risk
SecurityRequirement
SecurityObjective
SecuritySolution
Import
Security Concept(s)
Links
Link an Objective to a Risk
Link a Risk to a Damage
Link a Risk to a Threat
Link a Requirement to an Objective
Link a Requirement to a Solution
Link a Target to a TargetType
Link a Damage to an EssentialElement
Properties
Information
Model requests interpreter
Problems
Console
Error Log
Search
Navigator
53M of 248M

Figure 29: The risk assessment overview diagram

4.2.1 Risk assessment study initialisation

This section explains how to create a new risk assessment study and how to link it to an existing SMS study.

The screenshot below shows an existing SOA study, called SOA-DAY, with an ".aird" associated file.

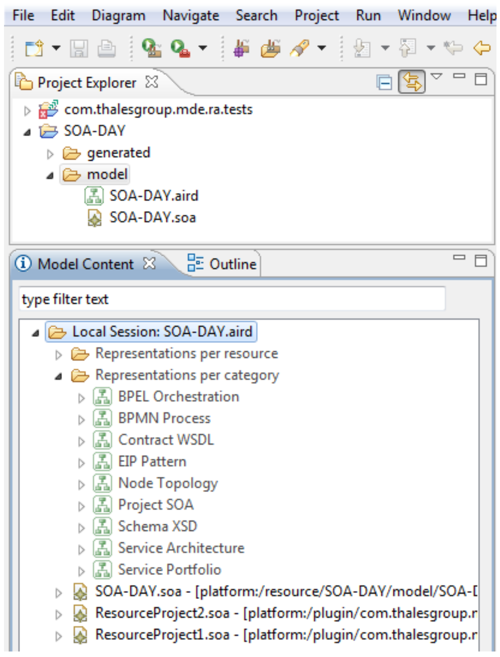

File Edit Diagram Navigate Search Project Run Window Help
Project Explorer
com.thalesgroup.mde.ra.tests
SOA-DAY
generated
model
SOA-DAY.aird
SOA-DAY.soa
Model Content
Outline
type filter text
Local Session: SOA-DAY.aird
Representations per resource
Representations per category
BPEL Orchestration
BPMN Process
Contract WSDL
EIP Pattern
Node Topology
Project SOA
Schema XSD
Service Architecture
Service Portfolio
SOA-DAY.soa - [platform:/resource/SOA-DAY/model/SOA-[
ResourceProject2.soa - [platform:/plugin/com.thalesgroup.r
ResourceProject1.soa - [platform:/plugin/com.thalesgroup.r

Figure 30: SMS project with its local session

The first step is to create the Risk Assessment model. To do that, click *File* > *New* > *Other...*

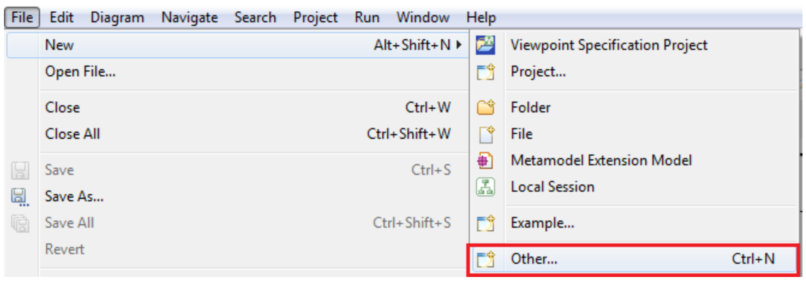
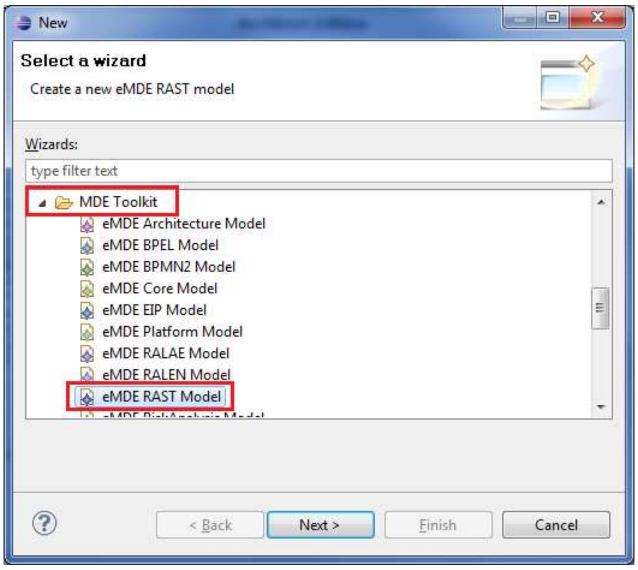


Figure 31: File > New menu

Select *eMDE RAST Model* in *MDE Toolkit* section. Click *Next >*.


New

Select a wizard

Create a new eMDE RAST model

Wizards:

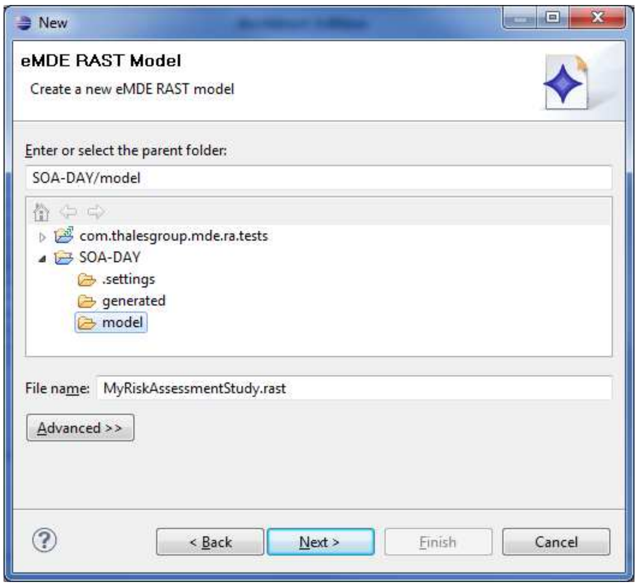
type filter text

- MDE Toolkit
  - eMDE Architecture Model
  - eMDE BPEL Model
  - eMDE BPMN2 Model
  - eMDE Core Model
  - eMDE EIP Model
  - eMDE Platform Model
  - eMDE RALAE Model
  - eMDE RALEN Model
  - eMDE RAST Model

< Back | Next > | Finish | Cancel

Figure 32: New window (1/3)

Select the parent folder where the new RAST model will be stored. Then, enter a file name for the study. Click *Next >*.


New

eMDE RAST Model

Create a new eMDE RAST model

Enter or select the parent folder:

SOA-DAY/model

- com.thalesgroup.mde.ra.tests
- SOA-DAY
  - .settings
  - generated
  - model

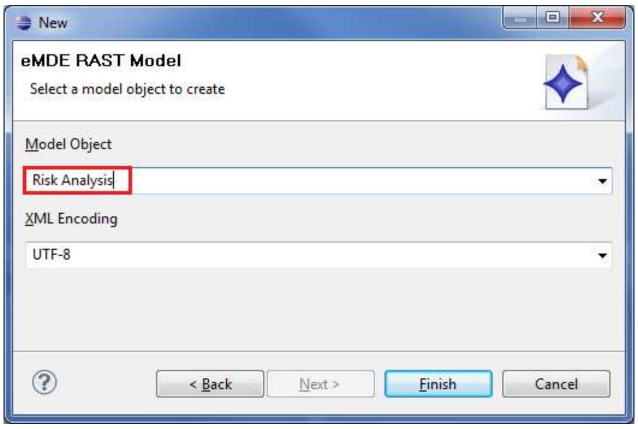
File name: MyRiskAssessmentStudy.rast

Advanced >>

< Back | Next > | Finish | Cancel

Figure 33: New window (2/3)

Select the model object to create (Risk Analysis in our case). Click *Finish*.


New

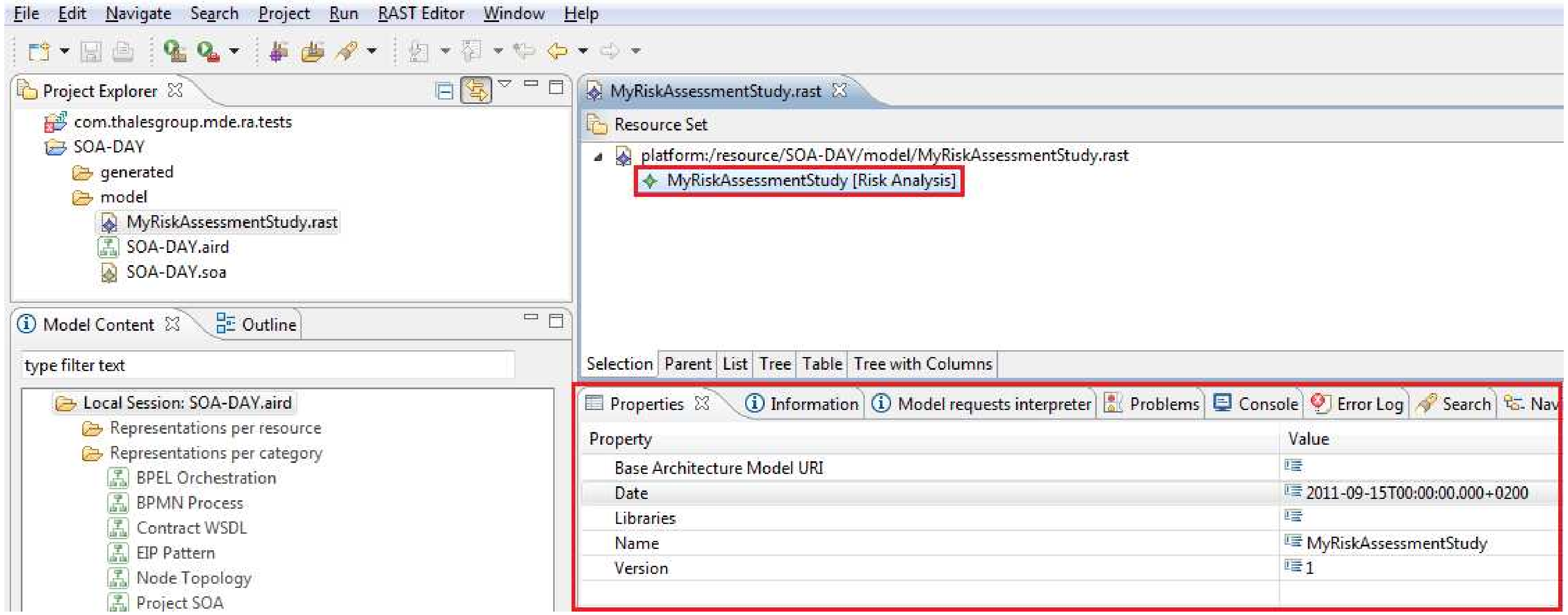
eMDE RAST Model

Select a model object to create

Model Object: Risk Analysis

XML Encoding: UTF-8

Figure 34: New window (3/3)


File Edit Navigate Search Project Run RAST Editor Window Help
Project Explorer
com.thalesgroup.mde.ra.tests
SOA-DAY
generated
model
MyRiskAssessmentStudy.rast
SOA-DAY.aird
SOA-DAY.soa
Model Content
Outline
type filter text
Local Session: SOA-DAY.aird
Representations per resource
Representations per category
BPEL Orchestration
BPMN Process
Contract WSDL
EIP Pattern
Node Topology
Project SOA
MyRiskAssessmentStudy.rast
Resource Set
platform:/resource/SOA-DAY/model/MyRiskAssessmentStudy.rast
MyRiskAssessmentStudy [Risk Analysis]
Selection Parent List Tree Table Tree with Columns
Properties Information Model requests interpreter Problems Console Error Log Search Nav

Property	Value
Base Architecture Model URI	
Date	2011-09-15T00:00:00.000+0200
Libraries	
Name	MyRiskAssessmentStudy
Version	1

Figure 35: The risk assessment study (RAST model)

The RAST model is now created. Fill the date, name and version *properties* of the model.

It is still needed to link the newly created RAST model with the SOA study. To do that, right click on the *Local Session* folder in the *Model Content* View. Then click *Add Semantic Resource*.

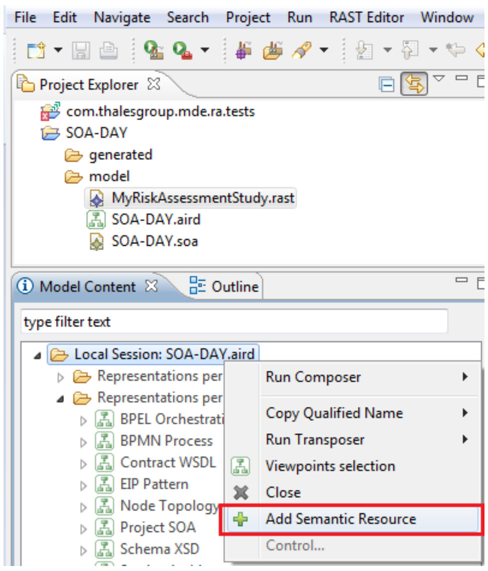
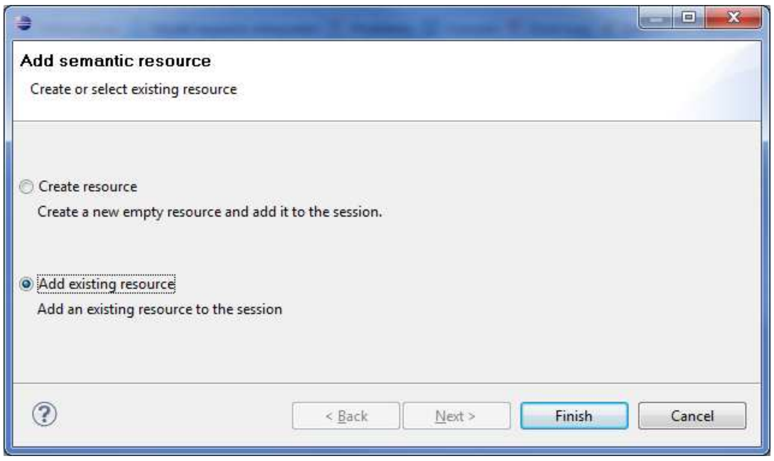


Figure 36: Add Semantic Resource menu

Select *Add existing resource*. Click *Finish*.


Add semantic resource

Create or select existing resource

Create resource

Create a new empty resource and add it to the session.

[✓ Add existing resource]

Add an existing resource to the session

Figure 37: Add Semantic Resource window

Select the resource to add by clicking on *Browse workspace...*

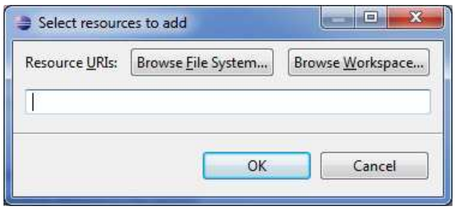


Figure 38: Select resources to add window

Select the created RAST model. Click *OK*.

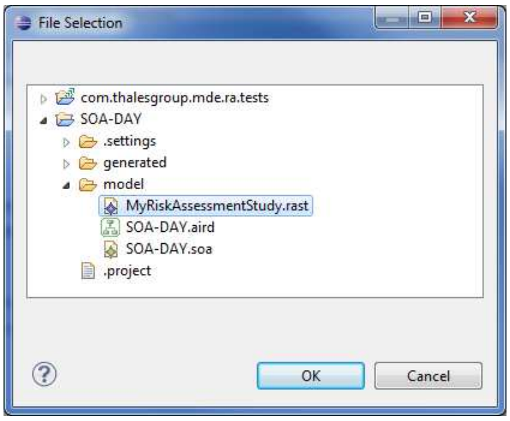


Figure 39: File selection window

Click *OK* on the *Resource to add* Window to finish.

The last step is to select viewpoints. The new resource (RAST model) allows selecting the *Risk Assessment* and the *Risk Assessment for SMS* viewpoints[32]. Select both and click Finish.

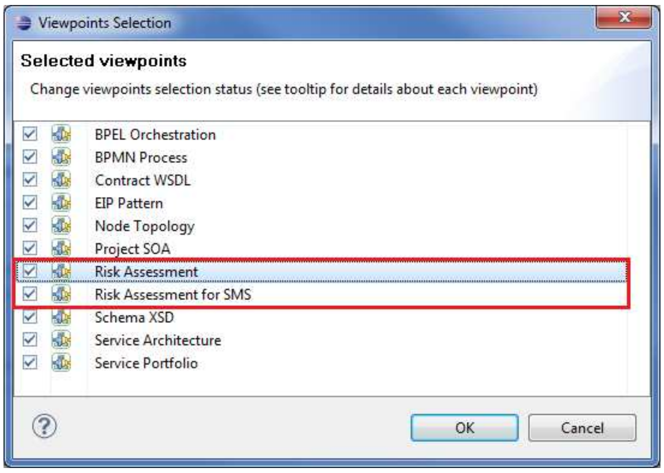


Figure 40: Viewpoints selection window

[32] The others are SMS viewpoints, and should already be selected.

The risk assessment viewpoint is the main viewpoint: it provides full support for the risk assessment activities. It is not explained herein.

The *Risk Assessment for SMS* viewpoint adds the capability to link elements from the SMS study to the Risk Assessment study and to maintain the consistency between both models. This viewpoint is detailed in §4.2.2.

4.2.2 The “Risk Assessment for SMS” viewpoint

The risk assessment tool offers multiple types of diagrams, the main one being the “RA Overview”. To use the additional functions provided by the *Risk Assessment for SMS* viewpoint, it is needed to activate the *SMS Layer* in the layers selection list of the RA Overview.

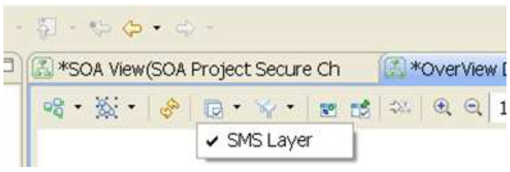


Figure 41: Selection of the SMS layer

The selection of the SMS layer provides new tools in the palette, cf. Figure 42. In particular:

- the *Import* section of the tool palette is extended with the two new tools to import existing primary and supporting assets[33] from the SMS study;
- primary and supporting assets can be created with specific types, like data store, ESB, firewall or other SMS concepts, so that they can latter be associated with SMS elements.

[33] Note that primary and supporting assets are called “essential element” and “target” in the tool.

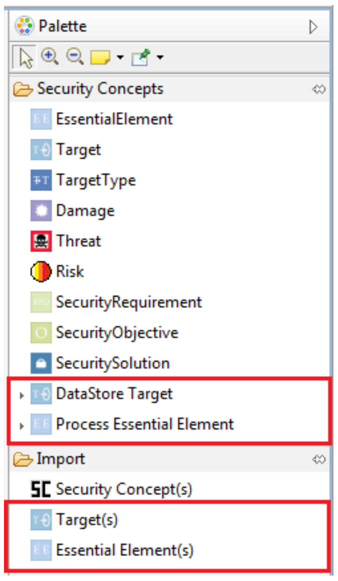


Figure 42: Tool palette

To import a primary or supporting asset from SMS, select the adequate tool from the palette and click on the RA overview background. An *Import Security Concept(s)* window opens.

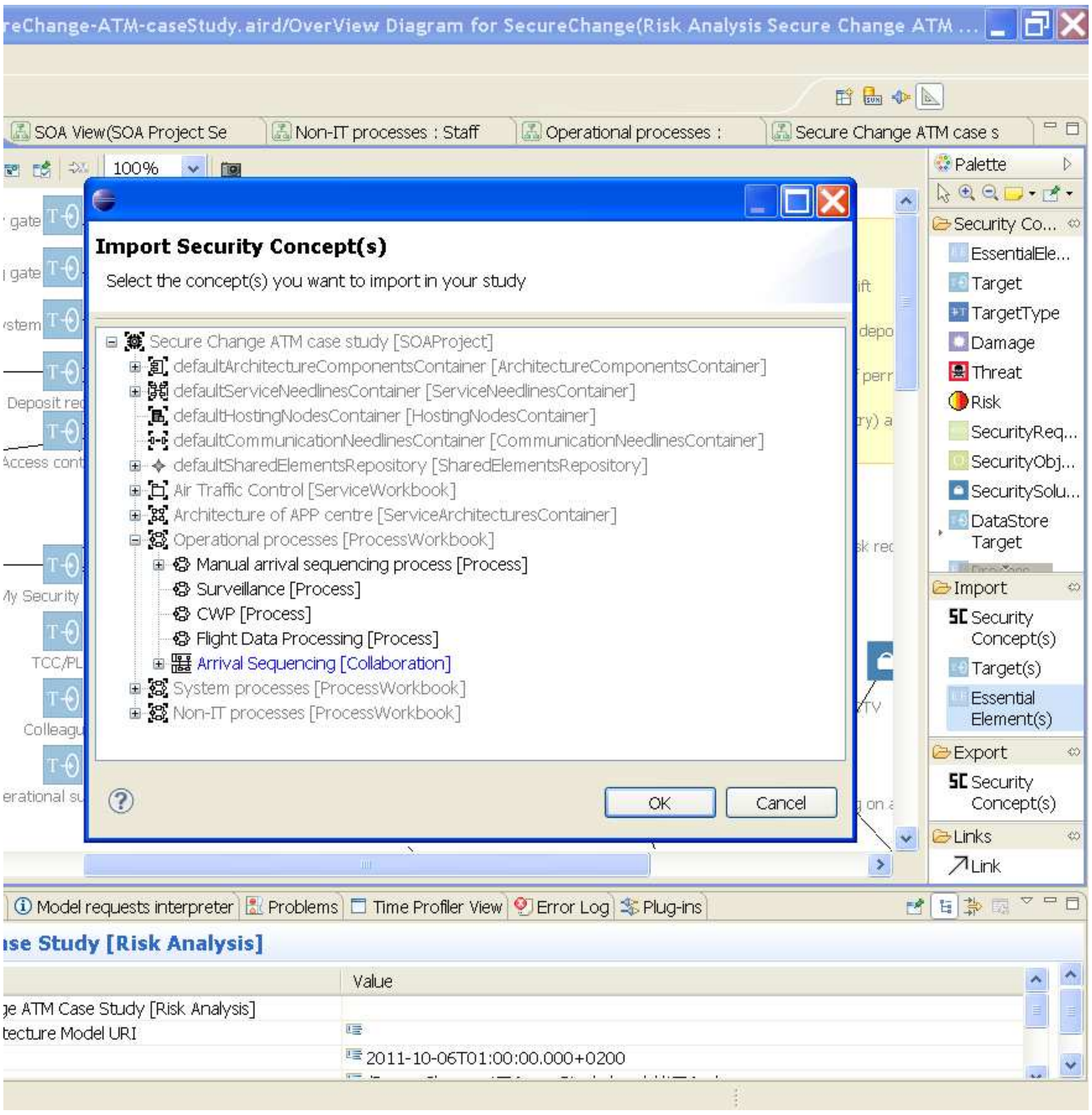
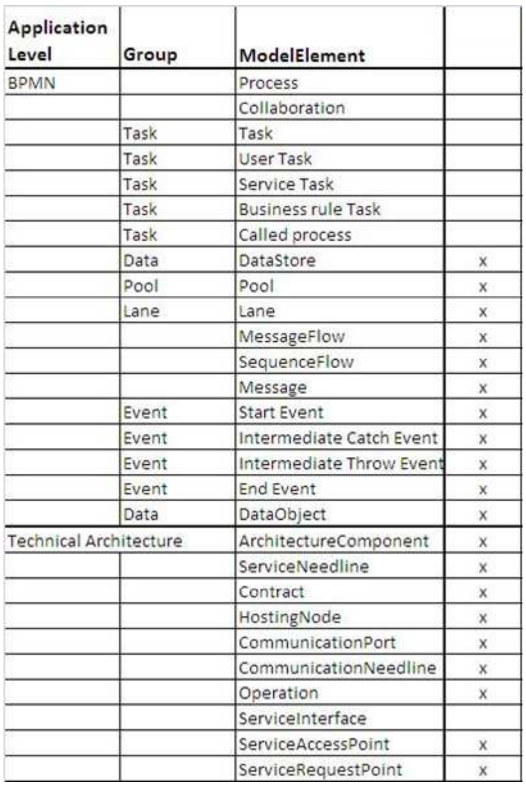

Import Security Concept(s)
Select the concept(s) you want to import in your study
Secure Change ATM case study [SOAProject]
defaultArchitectureComponentsContainer [ArchitectureComponentsContainer]
defaultServiceNeedlinesContainer [ServiceNeedlinesContainer]
defaultHostingNodesContainer [HostingNodesContainer]
defaultCommunicationNeedlinesContainer [CommunicationNeedlinesContainer]
defaultSharedElementsRepository [SharedElementsRepository]
Air Traffic Control [ServiceWorkbook]
Architecture of APP centre [ServiceArchitecturesContainer]
Operational processes [ProcessWorkbook]
Manual arrival sequencing process [Process]
Surveillance [Process]
CWP [Process]
Flight Data Processing [Process]
Arrival Sequencing [Collaboration]
System processes [ProcessWorkbook]
Non-IT processes [ProcessWorkbook]
OK
Cancel
Palette
Security Co...
EssentialEle...
Target
TargetType
Damage
Threat
Risk
SecurityReq...
SecurityObj...
SecuritySolu...
DataStore Target
Import
Security Concept(s)
Target(s)
Essential Element(s)
Export
Security Concept(s)
Links
Link

Figure 43: Import security concept(s) window

This window shows, organised as a tree, all the existing SMS elements:

- the elements that cannot be selected, because inadequate for the function, are shown in grey;
- the elements that have already been selected are shown in blue;
- the elements that can be usefully selected are shown in black.

The elements shown in this table are directly extracted from the SMS study. They are always up-to-date, and colour filtered according to precisely defined criteria (cf. Figure 44).



Application Level	Group	ModelElement	
BPMN		Process	
		Collaboration	
	Task	Task	
	Task	User Task	
	Task	Service Task	
	Task	Business rule Task	
	Task	Called process	
	Data	DataStore	x
	Pool	Pool	x
	Lane	Lane	x
		MessageFlow	x
		SequenceFlow	x
		Message	x
	Event	Start Event	x
	Event	Intermediate Catch Event	x
	Event	Intermediate Throw Event	x
	Event	End Event	x
	Data	DataObject	x
Technical Architecture		ArchitectureComponent	x
		ServiceNeedline	x
		Contract	x
		HostingNode	x
		CommunicationPort	x
		CommunicationNeedline	x
		Operation	x
		ServiceInterface	
		ServiceAccessPoint	x
		ServiceRequestPoint	x

Figure 44: Example of mapping of supporting assets with SMS elements

When a SMS element is imported into the risk assessment model, a proxy is created with the same name.

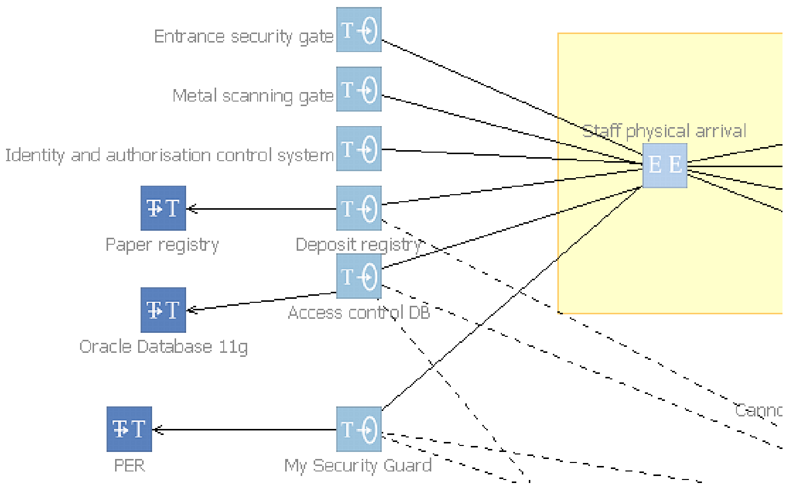


Figure 45: Examples of elements imported from SMS

In SMS, it is possible to visualise which elements have been selected by the security expert during the risk assessment. For this, it is necessary to activate the relevant layers (cf. Figure 46).

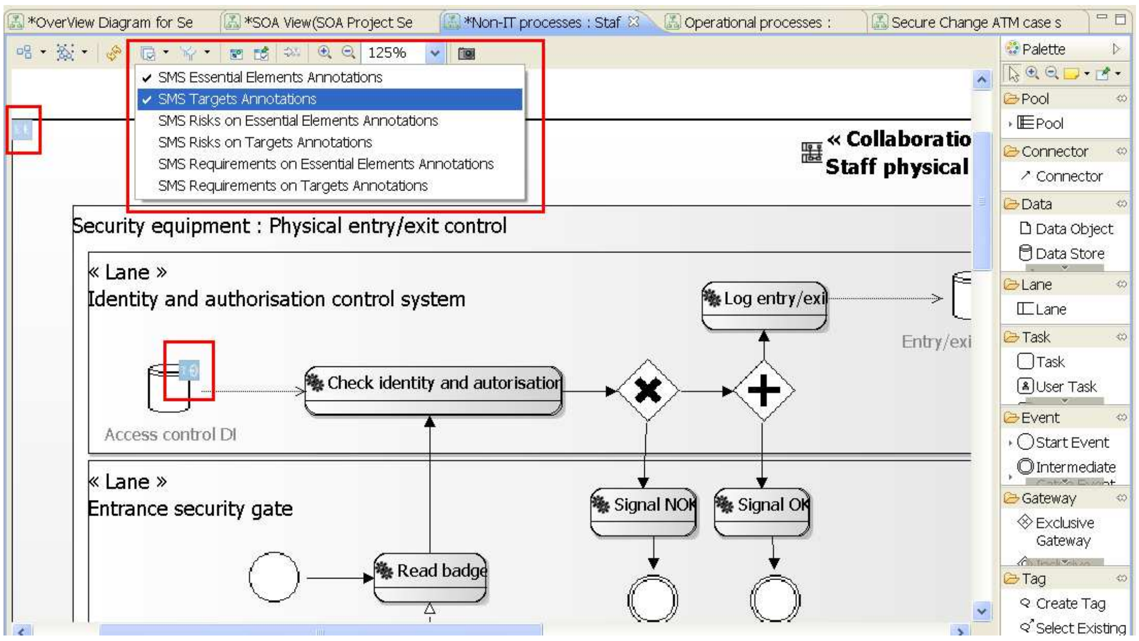


Figure 46: SMS annotations related to the selection of element in the risk assessment

Here, it can be seen that the "Staff physical arrival" process has been selected as a primary asset, and that the "Access control DB" data store has been selected as a supporting asset.

4.3 Quick insight on the prototype architecture

The architecture chosen for the risk assessment viewpoint meets a flexibility requirement: the tool can be run as standalone, or can be connected to a system engineering modelling tool to take advantage of the there-defined system models.

The current prototype is based on Eclipse and was developed under Obeo Designer.

The connexion with a system engineering modelling tool based on Eclipse is taken in charge by EMF itself. The figure below shows the proxy architecture implemented in this case.

The green tool is the standalone risk assessment viewpoint. It is possible to define assets or elements of a system (square green boxes) as well as self-borne security concepts. The risk assessment viewpoint is accessible to the user through its own set of diagrams.

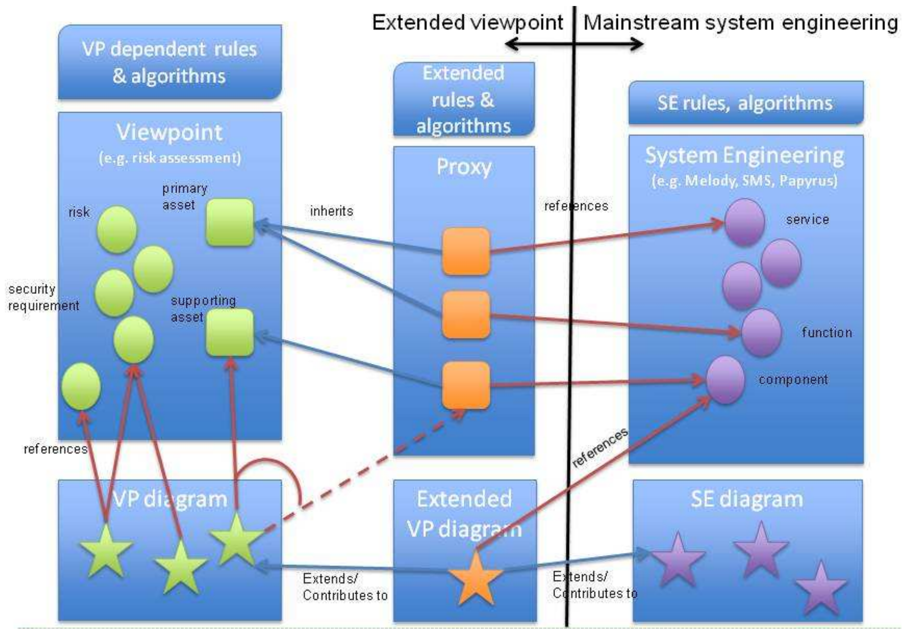


Figure 47 : Proxy architecture used in the case of an Eclipse based System modelling tool

The purple tool is a system engineering modelling tool such as SMS or Papyrus.

The connectivity between the risk assessment viewpoint and the system engineering modelling tool is handled simply through EMF references by means of the orange proxies implementing extended rules and algorithms, as well as additional layers annotating the system modelling tool diagrams (the purple diagrams).