Report on the Industrial Validation of SecureChange Solutions

M. Angeli (UNITN), G. Bergmann (BME), P. Capelastegui (TID),
B. Chetali (GTO), O. Delande (TRT), M. Felici (DBL),
F. Innerhofer-Oberperfler (UIB), V. Meduri (DBL), F. Paci (UNITN),
S. Paul (TRT), B. Solhaug (SIN), A. Tedeschi (DBL)

Document information

Document Number	D1.3
Document Title	Report on the Industrial Validation of SecureChange Solutions
Version	4.3
Status	Final
Work Package	WP 1
Deliverable Type	Report
Contractual Date of Delivery	30/01/2012
Actual Date of Delivery	
Responsible Unit	DBL
Contributors	BME, DBL, GTO, TID, TRT, UNITN
Keyword List	Validation, Case Studies
Dissemination level	PU

Document change record

Version	Date	Status	Author (Unit)	Description
0.1	23/06/2011	Draft	M. Felici, V. Meduri, A. Tedeschi (DBL)	Initial document structure
0.2	02/08/2011	Draft	M. Felici, V. Meduri, A. Tedeschi (DBL)	Initial information about the validation methodology
0.3	07/10/2011	Draft	M. Felici, V. Meduri, A. Tedeschi (DBL)	Initial structure of the sections for the ATM Case Study Validation and related WPs
0.4	23/11/2011	Draft	M. Felici, V. Meduri (DBL)	ATM WP2 Evaluation Results
0.5	08/12/2011	Draft	M. Felici, V. Meduri (DBL)	ATM WP3 Evaluation Results
0.6	15/12/2011	Draft	M. Felici, V. Meduri (DBL) S. Paul, O. Delande (TRT)	ATM WP4 Evaluation Results
0.7	28/12/2011	Draft	M. Felici, V. Meduri (DBL)	ATM WP5 Evaluation Results
1.0	07/01/2012	Draft	P. Capelastegui (TID)	HOMES Evaluation Results
2.0	11/01/2012	Draft	B. Chetali (GTO)	POPS Evaluation Results
3.0	11/01/2011	Draft	M. Felici (DBL)	Structured executive summary
3.1	12/01/2012	Draft	M. Felici (DBL)	Reviw of section Validation Objectives
3.2	16/01/2012	Draft	M. Felici (DBL)	Review of Appendixes
3.3	17/01/2012	Draft	M. Felici (DBL)	Review of Conclusions and Executive Summary
3.3	23/01/2012	Draft	M. Angeli (UNITN)	First quality check completed – minor remarks
4.0	25/01/2012	Draft	M. Felici (DBL)	Addressed first quality check remarks
4.1	27/01/2012	Draft	M. Felici (DBL)	Addressed comments by WP partners
4.1	30/01/2012	Draft	M. Angeli (UNITN)	Second quality check completed – minor remarks
4.3	30/01/2012	Final	M. Felici (DBL)	Addressed second quality check remarks

EXECUTIVE SUMMARY

This deliverable consists of the results of the validation including the evaluation of the applicability in realistic industrial contexts, together with recommendations for the future improvements and refinements of the project results.

The deliverable identifies and defines the validation strategy for the SecureChange final results, with the identification of the validation objectives (Section 1 Validation Objectives) for each WP and of the validation exercises and analyses of the related outcomes. The validation analyses the final and consolidated project results. It demonstrates that SecureChange artefacts can work efficiently in real life environments, while addressing the problem for which they have been developed:

> *SecureChange's objective is to develop techniques and tools that ensure "lifelong" compliance to evolving security, privacy and dependability requirements for a long-running evolving software system.*

The validation has taken place in the final year of SecureChange and it has delivered and influenced the research work in the last phase of the project lifecycle. The validation has provided insights for future further improvements and refinements of the SecureChange results. Particular attention has been given to the usability, of the project results, in real industrial contexts captured by the three different case studies: ATM (Section 2 ATM CASE STUDY), HOMES (Section 3 HOMES CASE STUDY) and POPS (Section 4 POPS CASE STUDY). Moreover, the validation criteria also include the applicability in real life and specific validation exercises designed to provide industrial feedback about essential aspects of the project results.

The SecureChange validation identifies the validation objectives (Section 1 Validation Objectives) with respect to the project outcomes (for each WP) and the way the validation activities have been organised and carried out (in the final part of the project) in order to address these objectives. Due to the complexity of validating diverse project outcomes, the validation strategy has taken into account changes and subsequent contributions. As natural consequence of the complexity of the SecureChange approach, tools and solutions that will be the outcomes of each SecureChange work package can be significantly different. Therefore, each work package has contributed to this document by designing, planning and performing different validation activities, compliant with the characteristics and scopes of the work package itself. The validation involves subsequent validation activities that have been planned for each case study and for each WP (Appendixes B, F, and H show the validation plans for the ATM, HOMES and POPS case studies, respectively). The validation activities combined together highlight validation strategies and processes tailored to the specific validation objectives and case studies.

Each work package has provided inputs for the final validation under the support and coordination of DBL, as leader of T1.3 and responsible of the overall Validation of the SecureChange results.

SecureChange Overview

The main objective of SecureChange is to develop techniques and tools that ensure "lifelong" compliance to security, privacy and dependability requirements for an evolving software system. This is particularly challenging because these requirements are not necessarily preserved by system evolution. The practical relevance of this research has been validated against three challenging complex industrial domains, i.e. smart-cards, digital homes and Air Traffic Management, which offer most research challenges and greatest long-term business opportunities.

The complexity and the innovation of the proposed solutions make the process of validating the results issued from this collective endeavour a very challenging task. Just as the issues addressed by SecureChange are heterogeneous, so are the results expected for each work package, ranging from an overall security architecture to specific methods and algorithms for the evolution of secure software, from a set of working tools for the design, implementation and verification of secure code, to purely conceptual frameworks and meta-models. Therefore, it was necessary to perform different and customized validation activities. Leaders of the technical work packages interacted and collaborated with the leaders of the validation tasks in defining the evaluation and validation objectives (Section 1 Validation Objectives), criteria (as identified in D1.2 [3]) and methods for the work package they have been responsible for, because of their awareness of the issues at stake and the solutions developed within their work package. In order to identify realistic and challenging validation criteria and to support collaborations among the technical work packages and the industrial partners (to minimise the risk that each work package would define its own criteria of success independently of the others partners with little or no interaction within the SecureChange project), we used the real-world case-studies as means for evaluating how SecureChange meets its main goals. Figure 1 stresses the critical role of a proper, realistic and coordinated validation plan for a successful SecureChange validation.

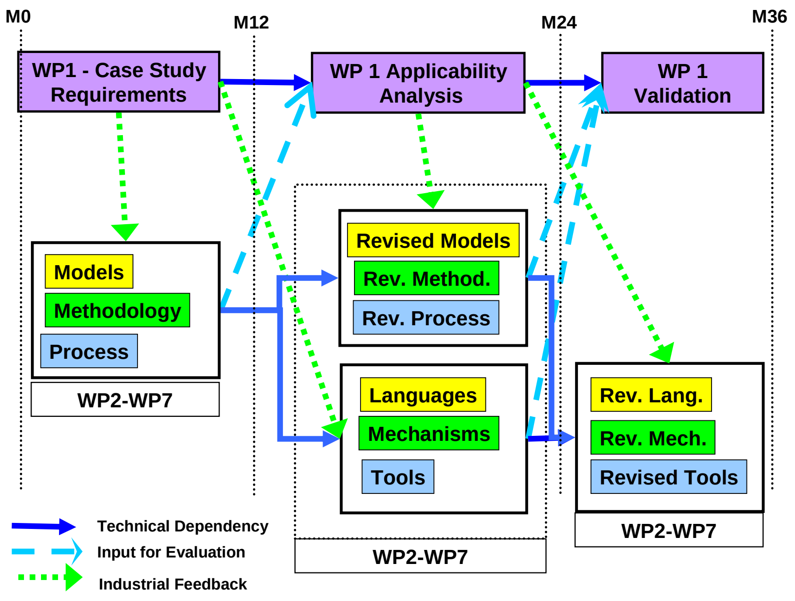


Figure 1 Work package 1 and other work packages

In order to provide some guidance to the validation activities across the three different case studies, we have identified a general and widely used Validation Process (Appendix A). The Validation Process has been drawn from industry practices and it has been used within the SecureChange project to guide and coordinate the different evaluation tasks. Each work package interacted with one or more case studies, according to main scopes and security characteristics of the case studies themselves. Table 1 summarises the interactions among technical work packages and the case studies. These interactions have contributed to the validation of the SecureChange outcomes. The validation results for each work package are reported for the three case studies: ATM (Section 2 ATM CASE STUDY), HOMES (Section 3 HOMES CASE STUDY) and POPS (Section 4 POPS CASE STUDY).

Table 1 Interaction among industrial case studies and technical work packages

	WP2	WP3	WP4	WP5	WP6	WP7
POPS			X		X	X
HOMES	X				X	X
ATM	X	X	X	X		

Tools, techniques and methodologies developed by the work package have been to a certain extent integrated and applied within the selected case study to specific scenarios highlighting the peculiarities, innovation and applicability of the SecureChange outcomes under validation. Therefore, the Validation of the SecureChange has been designed and carried out by following a *case-study-oriented structure*.

Project Rationale and Results

System evolution has captured centre stage in software engineering research and practice for good reasons. There is growing demand to continuously evolve systems to meet changing business needs, new regulations and policies, novel technologies and computing infrastructures. The main objective of SecureChange is to develop techniques and tools that ensure "lifelong" compliance to security requirements for an evolving software system. This is particularly challenging because these security requirements are not necessarily preserved by system evolution. The project has focused on the challenging, long term objective of rethinking processes and tools that support design techniques for evolution, testing, verification, re-configuration and time deployment analysis of evolving software.

SecureChange successful achievements consist of these tangible **scientific and technological results**:

1. A **specification** of the SecureChange software design process based on meta-models. This also includes the specification of a security architecture supporting the adaptable configuration of security functionality on the basis of the concept of security as a service.

2. A series of **methods and algorithms** supporting the process at each step. These methods provide mechanisms for:
    a. Specifying code-level security properties in a light-weight way (via programming models) in order to cope with changing vulnerability classes.
    b. Verifying code that is loaded on a device in light of changing requirements or risk models, computing impact of requirement evolution on already installed code, and re-validating already installed code again new requirements.
    c. Incrementally reacting to changes in requirement and design models.
    d. Eliciting evolutionary requirements.
    e. Specifying design-level models for evolutionary systems and their security requirements and formally analyzing the models against the requirements.
3. A suite of **software tools** supporting SecureChange solutions:
    a. A tool managing evolving, configurable and highly-interrelated services and their security requirements at several levels of abstraction.
    b. A tool for the automatic transformation of security requirements models into security design models when requirements change or the automatic re-assessment of requirements when design models and processes are changed.
    c. A tool to automatically and formally verify design models of evolutionary systems against evolutionary security requirements.
    d. A tool for the verification of evolving security requirements and the automatic transformation of security requirements models into security design models when requirements change or the automatic re-assessment of satisfied or unsatisfied security requirements when design models and processes are changed (i.e. in process re-engineering).
    e. A tool for automatic or interactive simulation of changes to the risk picture as a function of executing changes in the underlying system description (with change as a first class citizen).
    f. A model-based testing tool prototype to automatically (re)generate the test repository and automated test scripts according with property and requirements evolutions, and managing the priority detected from risk analysis.
    g. A tool to verify security properties of code that is annotated according to a programming model.
    h. A tool to automatically and formally verify design models of evolutionary systems against evolutionary security requirements.
    i. A tool to verify mobile code at loading time against flexible security requirements and to detect the impact of new requirements on already installed code.
4. The **industrial validation of the scientific and technological results** has been done in three case studies. The industrial validation involved:
    a. A **requirement and success criteria collection** of the SecureChange case studies. This includes a complete description and a detailed specification of security requirements of each of the case studies.

b. A **process validation**. This consists of validating that the lifelong development cycle designed in SecureChange can be applied in industrial scenarios.

c. A **final evaluation of the technologies**. Technologies studied, developed or improved during the project have been evaluated to validate its applicability to industrial contexts.

Purpose of This Document

This document identifies and defines a validation strategy for the SecureChange final results, with the identification of the validation objectives and of the validation exercise validation and analysis of the related outcome. The SecureChange validation analyses the final and consolidated version of the project results, so as to demonstrate that SecureChange can comply with changes requirements and security features drawn from industrial contexts, while addressing the problems for which it was developed. The purpose of this document is to provide insights for future further improvements, refinements and exploitations of the SecureChange results. Particular attention has been paid to the usability of the project results in the selected industrial contexts. In particular, the identified validation criteria include the applicability on validation exercises designed to provide feedback about SecureChange objectives. This deliverable reports the results of the validation including the evaluation of the applicability in realistic industrial contexts, together with recommendations for the future improvements and refinements of the project results.

Document Structure

The document organisation takes into account the interactions among the technical work packages and the case studies. Section 1 identifies the Validation Objectives for each WP. The validation criteria (identified in D1.2 [3]) have been reported together with validation scenarios, exercises and results for each validation objective. This allowed us to identify specific feedback for all SecureChange results. The remainder of the document is organised per case study. Section 2 reports the validation activities and results of the WPs that worked on the ATM CASE STUDY. Section 3 reports the validation activities and results of the WPs that worked on the HOMES CASE STUDY. Section 4 reports the validation activities and results of the WPs that worked on POPS CASE STUDY. Section 5 summarises the validation results. The Appendixes provide specific supporting material (e.g. definitions, notations details of exercises). Three appendixes detailed the validation plans adopted for the case studies with respect to the assessed WPs. Appendixes B, F, and H show the validation plans for the ATM, HOMES and POPS case studies, respectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Validation Objectives

This section identifies the SecureChange solutions, i.e. methodologies and tools, developed by the technical WPs. They are the focus of the validation activities. This section reports the main Validation Objectives for each developed solution. The validation objectives that have driven the first two year of the project have been tailored to the different case studies (as identified in D1.2 [3]) and they are reported together with the validation activities conducted on the SecureChange Solutions. The work in the last year of SecureChange has focused on the validation of WPs' technical artefacts, rather than the development of new ones.

1.1 WP2 – Architecture and Design Process

The main outcome for WP2 was the development of concepts and tools for architecture, design and operation of lifelong systems. In particular, WP2 delivered a framework for a **Security Architecture for Evolving Requirements** and a **tool-based change-driven security engineering process**. Main Objectives to be fulfilled by the Security Architecture for Evolving Requirements are **COVERAGE** and **APPLICABILITY.**

Regarding **COVERAGE**, criteria identified for its validation state that:

1. Every type of security requirement must be realized through one or more technical security service in the Target Security Infrastructure (TSI).
2. Every type of security requirement must be realized through one or more abstract security service in the Security Architecture (SA).
3. Abstract security services and compositions map to configurable technical services or compositions thereof in the Target Security Infrastructure (TSI).

Regarding **APPLICABILITY**, we identified the following criteria:

1. The TSI must cover change scenarios which are related to the ATM and/or the HOMES case study.
2. Service composition in the SA and TSI must be consistent.
3. The mapping of the security requirements onto the services in the SA and TSI must be valid.

Relevant goals for an effective Conceptual Model for a security engineering process for evolving systems are traceability, change-driven process and modularity. Formally stated the Objectives are **COVERAGE**, **APPLICABILITY**, and **ANALYZABILITY**.

Regarding **COVERAGE** the following criteria are to be fulfilled:

1. Relevant artefacts which are processed in security engineering must be categorized and identified.
2. Relevant activities which are executed in security engineering must be categorized and identified.

APPLICABILITY validates the change-driven by ensuring that:

1. Change-driven security engineering process must cover change scenarios which are related to the HOMES and/or ATM case study.

We identified the following criteria with respect to **ANALYZABILITY**:

1. Models with states of the processed artefacts must be analyzable by using reasonable techniques.

1.2 WP3 – Requirements

The main results of WP3 involve: a conceptual model for the characterisation of evolving requirements, a methodology base on such conceptual model, algorithms and reasoning techniques for incremental requirements models evaluation and transformation, and a tool to analyse the impact of changes in requirements models. Among the WP3 result is the identification of the key features of requirements models that are subject to evolutionary transformations and the definition of the methodological aspects for their graphical and conceptual representations, their versioning capture and their run-time monitoring. Objectives to be fulfilled in the definition of the conceptual model for the characterisation of evolving requirements are: **COVERAGE**, **APPLICABILITY** and **ANALYZABILITY** of the model.

For the **COVERAGE**, two criteria are identified:

1. All requirements types must be representable in the model.
2. Evolution of requirements must be representable in the model.

Regarding the model **APPLICABILITY**, one of the main relevant criteria is:

1. Case study requirements must be representable in the model.

Finally, the **ANALIZABILITY** of the developed model states that:

1. The model must be analyzable by using reasoning techniques.

The requirement conceptual model and the associated general methodology should be able to handle the changes on security requirements, including how to represent security requirements, how to model the changes of them, how to manage the changes and how to argue that the changes are fit for the purposes. The conceptual models were refined to better satisfy WP3 objectives.

Other WP3 results involve algorithms and reasoning techniques for incremental requirements models evaluation and transformation. Objectives to be fulfilled in the definition of the algorithms and reasoning techniques for incremental requirements models evaluation and transformation are: **COVERAGE**, **APPLICABILITY** and **PERFORMANCE**.

For the **COVERAGE**, one criterion is identified:

1. The transformation algorithms are able to propagate any type of change.

For the **APPLICABILITY**, one criterion is identified:

1. Transformation algorithms must be able to propagate change between different requirements models and design models.

For the **PERFORMANCE**, two criterions should be obtained:

1. Changes are detected and propagated faster than by traditional transformation algorithms.
2. Complex models can be handled by the algorithm.

WP3 has also developed a prototype of the tool to analyze the impact of a change in the requirements model. Objectives for the tool are: **FUNCTIONALITY** and **USABILITY**.

For the **FUNCTIONALITY**, two criteria are identified:

1. The representation of the conceptual model for requirements should be supported by the tool.
2. Impact of a change can be assessed by the tool.

For the **USABILITY** one criterion is:

1. The representation of the requirement conceptual model and the change analysis should be easy to use functionality for end-users.

1.3 WP4 – Model Design

WP4 developed methodologies for modelling evolution in security designs and requirements for integrating security design, modelling and assessment technologies as well as models into a security engineering process. Main result of SecureChange is a conceptual process related to industry practices for the characterisation of evolving design. Properties relevant for the model developed are:

COVERAGE / **APPLICABILITY** of the model, that means:

1. The design must be representable in the model.
2. Change must be representable or detectable in the model.

SOUNDNESS of the model with respect to changes and evolution:

1. The model-based approach must be sound with respect to change.

Formal **ANALYZABILITY** of the model:

1. The model must be analyzable by using reasoning techniques.

1.4 WP5 – Risk Assessment

During the first year of project, WP5 evaluated existing methods and principles for assessment of security, privacy and dependability and identified strengths and weaknesses of existing methods with respect to assessment of long-lived and evolving systems. Then WP5 pinpointed open issues that needed to be addressed, while still making full use of existing knowledge in the field. Furthermore, WP5 during the first year developed a risk modelling language for documenting forecasts of future evolvement of a system. The language should have the expressiveness to capture future evolvement of a system, while still being suitable for use in an assessment process that involves analysts with different backgrounds and levels of training. Thus, main objectives for the definition and development of a proper and effective language addressing evolvement are:

COVERAGE, that is:

1. Change must be representable in the model.
2. Relevant risks must be representable in model.

APPLICABILITY, that is:

1. Models must be easy to make, use and read.
2. Model must be scalable.

During the second year of the project, WP5 developed a method for security, privacy and dependability assessment with respect to forecast of future evolvement, a framework for integrated documentation of system and assessment results, and a threat management method incorporating the parameter space of applications. For the methods the main objectives are:

COVERAGE, that is:

1. Methods must discover relevant risks.
2. Method must make useful predications.

APPLICABILITY, that is:

1. Method must be usable for heterogeneous groups.

For the documentation framework the main objectives are:

APPLICABILITY, that is:

1. Models must be representable.
2. Traceability of models.

For the threat management method the main objectives are:

COVERAGE, that is:

1. Parameter space of application representable in model.
2. Method must make useful risk estimations.

During the third year of the project, WP5 has developed techniques, methods and tools for automatic or semi-automatic revalidation of assessments with respect to change.

1.5 WP6 – Verification

In WP6, a new programming model has been defined for an emerging vulnerability class. This high-level objective is demonstrated by the following concrete results:

- A conceptual model characterizing a new programming model.
- A notation supporting the programming model.

The objectives of this WP, during year one, were to identify an emerging relevant vulnerability and to define a suitable programming model to counter it. Consequently, a user-friendly notation has been defined to be used in order to annotate the source code. The notation should have the following 3 main characteristics:

1. **COVERAGE**
   (a) The programming model must ensure absence of a well-defined class of security vulnerabilities.
   (b) The programming model should support common coding patterns (as long as they do not violate security).
2. **SOUNDNESS**
   (a) A program that is verified to comply with the programming model should not contain vulnerabilities of the class covered by the programming model.
3. **LOW ANNOTATION OVERHEAD**
   (a) The programming model should not impose too much additional effort on developers.

1.6 WP7 – Testing

WP7 focused on developing a methodology and a prototype of model-based testing tool for testing the evolutions. Test generation is ensured by a model-based approach. Test sets are based on particular coverage criteria, e.g. security requirements criteria. Studying the impact of evolution on model-based testing approach, and taking into account the state of the art, WP7 proposed extensions of methods and tools to deal with evolution. WP7 generated tests to emphasize the correctness of the system with respect to evolution, on the base of the requirements and model changes. Objectives to be covered for the validation of the Behavioural and Security Model developed in WP7 are:

1. **COVERAGE:**
   (a) Security properties specified in addition to the model.
   (b) Evolution of the requirements can be expressed as model modifications.
2. **APPLICABILITY:**
   (a) The security aspects are representable in the model.
   (b) The model evolutions can be translated as updates of its transition relation.

2 ATM CASE STUDY

This section describes the validation activities and results based on the ATM case study. The validation organisation has been tailored to capture specific validation objectives with respect to the SecureChange artefacts and ATM domain features. The overall validation organisation, activities and objectives build over the previous WP1 deliverables [1][2][3], which have defined the scope and feasibility of SecureChange artefacts.

2.1 VALIDATION ORGANISATION AND CONDUCT

The ATM case study focused on four work packages (i.e. WP2 Architecture and Design Process, WP3 Requirements, WP4 Model Design and WP5 Risk Assessment) and their artefacts. Due to the nature of the ATM case study (mainly concerning with technological changes from an organisational viewpoint) the WPs focusing on requirements, design and assessment aspects, the ATM case study has contributed towards the validation of relevant artefacts supporting specific design and assessment activities while preserving critical security features. Each WP has produced different artefacts (e.g. methodologies, tools). Hence, it has been necessary to tailor the validation activities to the different peculiarities of the artefacts and their developmental stages. This required WP-tailored validation activities. This section describes the validation activities for each WP from an organisational viewpoint and discusses them.

2.1.1HIGH–LEVEL OBJECTIVES

The validation objectives of the ATM case study have concerned the relevance of SecureChange artefacts and their assessments by ATM domain experts (e.g. Air Traffic Controllers) and potential end-users (e.g. IT and operational experts within an Air Traffic Control Service provider). The validation activities have been tailored for each WP and related artefacts. This is to take into account the different nature of the artefacts (e.g. methodologies, modelling languages, tools). Moreover, it has been necessary to support different developmental paths of the artefacts. All SecureChange artefacts delivered by the ATM-related WPs have been validated by subsequent activities in order to support their developments through subsequent refinements (i.e. adjustments due to feedback). The main validation activities fall into three major categories: Methodology Evaluation (modelling), Walkthrough and Tool Live Demo with ATM Experts. Methodology evaluation consisted of modelling exercises focusing on specific changes and security requirements in order to refine and consolidate the underlying modelling languages and their methodologies, respectively. Walkthrough activities involved step-by-step evaluation of the SecureChange methodologies with ATM experts. This allowed to assess the proposed methodologies with domain experts and to identify alternative usages (with respect to current practices within the ATM domain). Finally, tool live demo activities and exercises allowed the validation (in terms of usability and acceptance by ATM experts) of the tools supporting the SecureChange methodologies. Figure 2 shows the subsequent activities and their focus on the SecureChange artefacts forming the ATM validation. The ATM validation is consistent with the SecureChange Validation Iterative Process (as identified and described by the WP1 deliverables [1][2][3]).

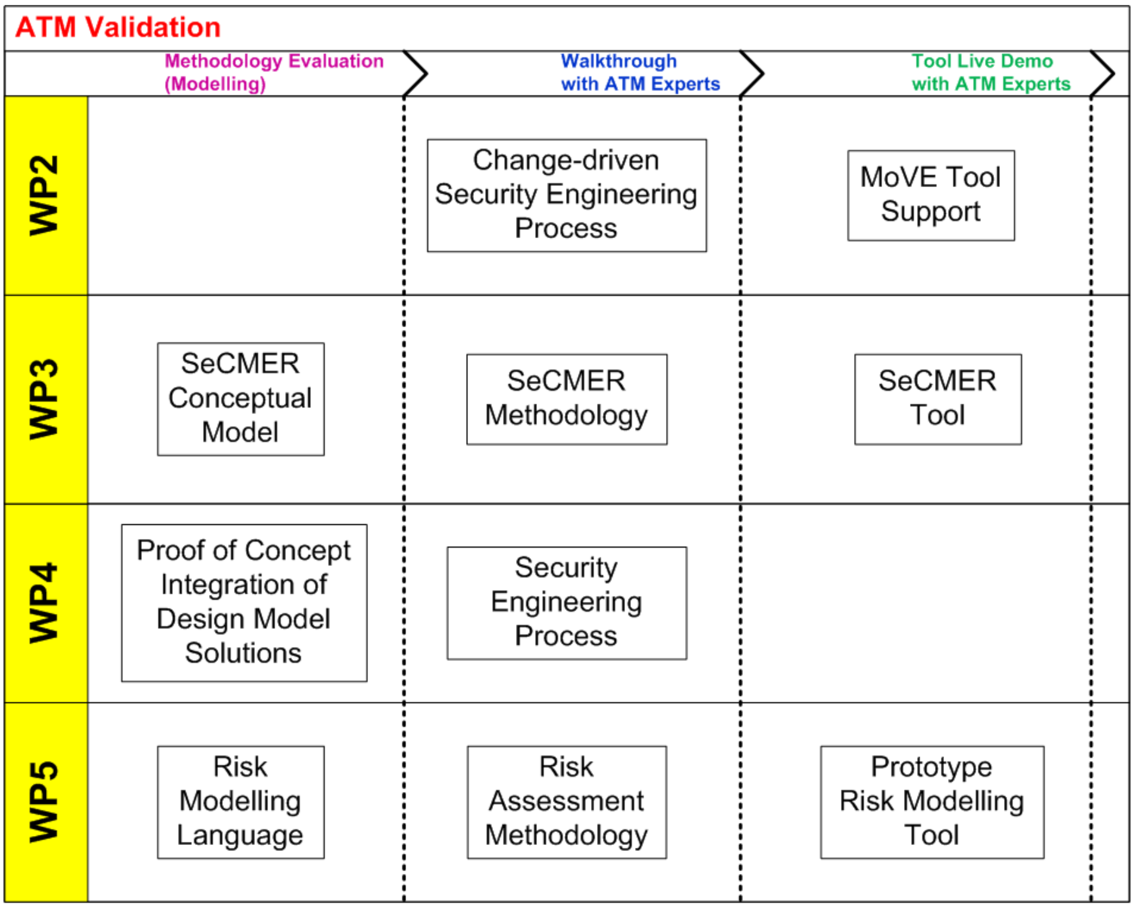

ATM Validation
Methodology Evaluation (Modelling)
Walkthrough with ATM Experts
Tool Live Demo with ATM Experts
WP2
WP3
WP4
WP5
Change-driven Security Engineering Process
MoVE Tool Support
SeCMER Conceptual Model
SeCMER Methodology
SeCMER Tool
Proof of Concept Integration of Design Model Solutions
Security Engineering Process
Risk Modelling Language
Risk Assessment Methodology
Prototype Risk Modelling Tool

Figure 2 ATM Validation

Each validation activity involved ATM experts in order to assess SecureChange artefacts from a practitioner viewpoint and to identify opportunities for exploitation of project results within the ATM domain. The ATM case study identifies specific user needs and expectations for the ATM industrial domain. In particular, the ATM validation highlights how SecureChange solutions can be used in the application domain and expected improvements to comply with industry practices. The remainder of this section describes the validation outcomes for each WP artefact. For each validated SecureChange artefact, it describes the main artefact features, the high-level validation objectives (as identified in [3]) and the validation results.

2.2 WP2 Architecture and Design Process

2.2.1 WP2 ARTEFACTS

The validation activities for WP2 focused on the two main artefacts delivered:

1. Change driven security engineering process.

2. Tool-Support by MoVE Framework.

The change driven security engineering process highlighted a methodological account of security engineering from a process viewpoint. The procedural account of security engineering is fundamental in order to coordinate and relate different modelling artefacts. The change driven security engineering process was presented to and validated by ATM experts in a dedicated workshop (Rome, June 2011). The MoVE framework captures the change driven security engineering process. The tool implementing and supporting the MoVE framework, hence the change driven security engineering process, has been presented to and validated by ATM experts in another dedicated workshop (Rome, September 2011). The following sections report the results of the validation activities concerning the WP2 artefacts (i.e., the change driven security engineering process and the tool-Support by MoVE Framework).

2.2.1.1 Change Driven Security Engineering

The change driven security engineering process has been initially validated in a dedicated workshop (Rome, June 2011) with ATM experts. The initial validation focused on the aspects that concern the process. The rationale for the process is that it is necessary to support different stakeholders, who may need to be informed about specific changes affecting critical security properties. The SecureChange process is concerned in particular with those business stakeholders who are responsible for the design, implementation and deployment of ICT in complex application domains. The change driven security engineering process aims to support collaboration among stakeholders in order to assess the impact of changes on security properties. Moreover, it would enable them to identify what aspects of the ICT have changed and their impact on critical security properties. This requires the process to support different modelling artefacts that are useful to the stakeholders (see Figure 3).

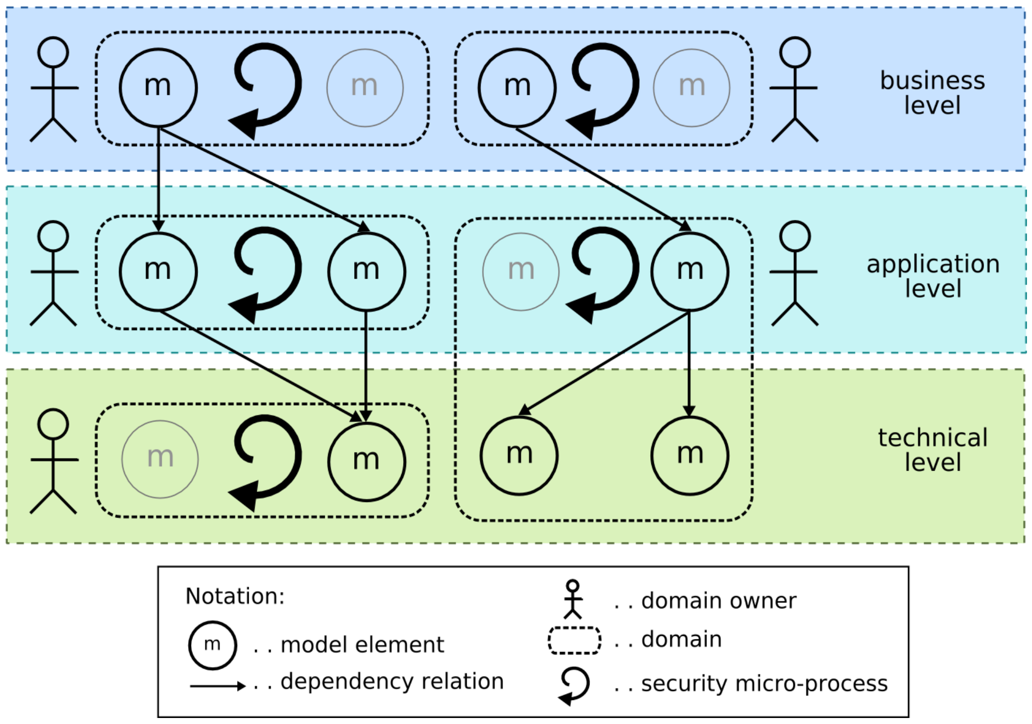


Figure 3 Collaboration support

The change driven security engineering process copes with the identified challenges by:

- being fully driven by change events and change propagation,
- supporting change propagation based on dependencies between artefacts,
- providing an integrated view of the entire system, since change cannot be tackled from a single perspective anymore,
- supporting a tight integration of functional and security aspects,
- defining clear domains and responsibilities for the various stakeholders.

The change driven security engineering process supports different stakeholders by means of *common system views* (consisting of functional system models, security models domains and responsibilities) and *mechanisms of reflecting changes* (consisting of model element states, change events and change propagations). Figure 4 shows the functional system meta-model underlying the change driven security engineering process. The different views provide direct mappings on the level of model elements (e.g. requirement models, system models, architecture models) that are tailored to support different stakeholders. The coordination of such elements, hence the corresponding views, is central to the change driven security engineering process.

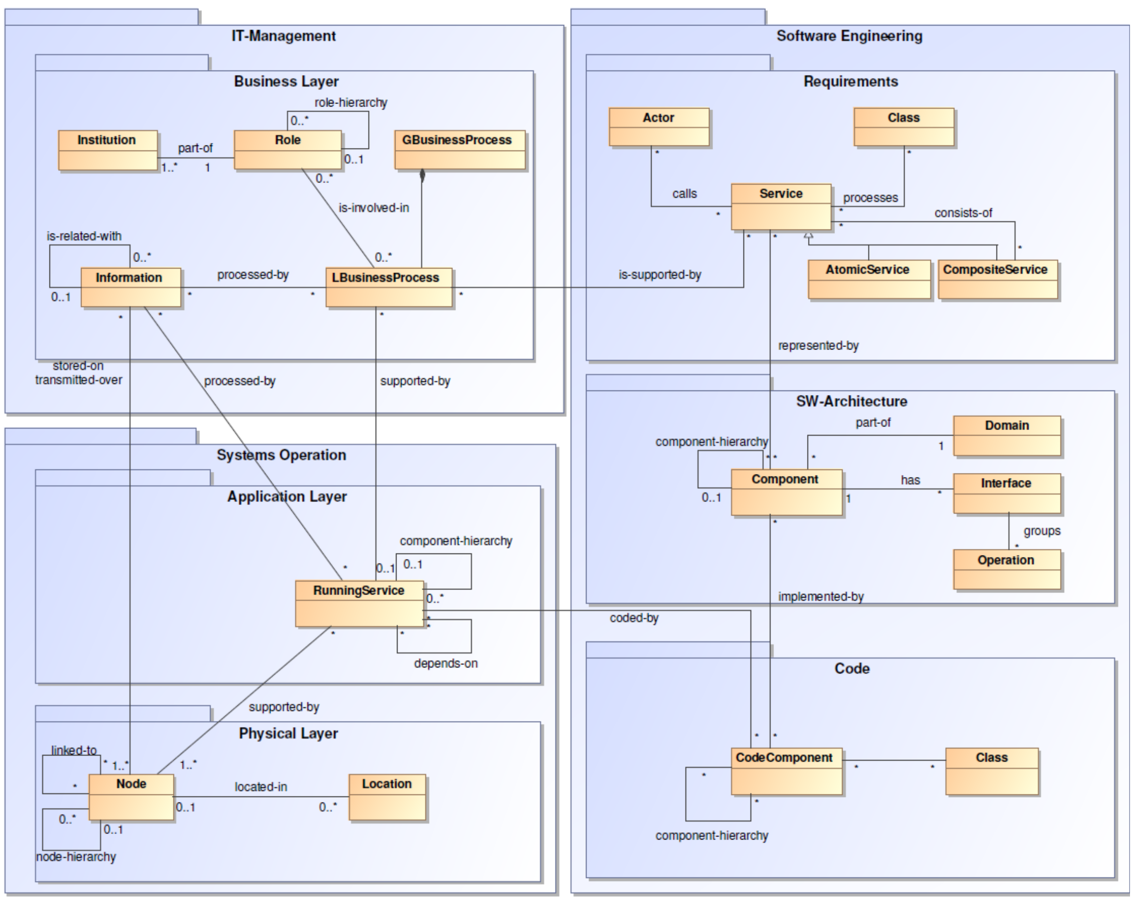


Figure 4 Functional system meta model

VALIDATION SCENARIOS and EXERCISES

The validation workshop consisted of a walkthrough of the ATM case study. The WP2 models and basic notions were presented to ATM experts and contextualised for the ATM case study (in terms of changes and security properties). WP2 validation focused on the *ATM Organizational Level Change* and on the security properties of *Information Protection* and *Information Provision* [1][2][3]. The impact of the introduction of the AMAN (Arrival MANager) was assessed according to different viewpoints: information view, business process view and system view. Figure 5 shows for instance the impact of the introduction of the AMAN from a business process viewpoint of analysis. The analysis of changes involved also an information view (highlighting changes in the information flow) and a system view (highlighting changes in the system architecture). All these viewpoints of analysis contributed to the assessment of the impact of changes, hence, the risk assessment. The changes triggered new requirements that were assessed. The risk analysis was then updated in order to account for AMAN introduction and the potential impact on the security properties. This required a coordination of different models (e.g. requirements, architecture and risk) while updating their status as modified by relevant engineering activities.

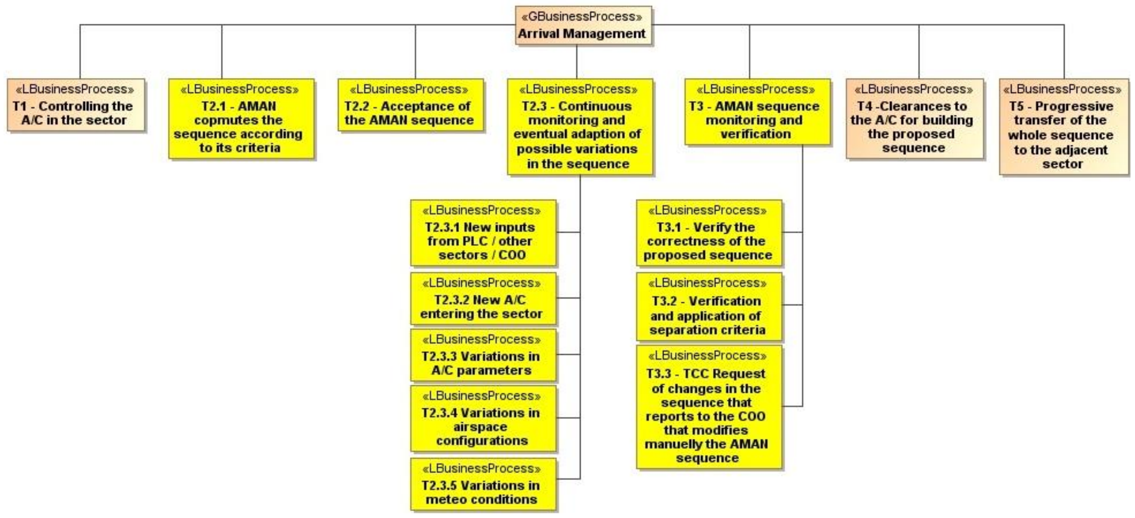

«GBusinessProcess» Arrival Management
«LBusinessProcess» T1 - Controlling the A/C in the sector
«LBusinessProcess» T2.1 - AMAN copmutes the sequence according to its criteria
«LBusinessProcess» T2.2 - Acceptance of the AMAN sequence
«LBusinessProcess» T2.3 - Continuous monitoring and eventual adaption of possible variations in the sequence
«LBusinessProcess» T3 - AMAN sequence monitoring and verification
«LBusinessProcess» T4 -Clearances to the A/C for building the proposed sequence
«LBusinessProcess» T5 - Progressive transfer of the whole sequence to the adjacent sector
«LBusinessProcess» T2.3.1 New inputs from PLC / other sectors / COO
«LBusinessProcess» T2.3.2 New A/C entering the sector
«LBusinessProcess» T2.3.3 Variations in A/C parameters
«LBusinessProcess» T2.3.4 Variations in airspace configurations
«LBusinessProcess» T2.3.5 Variations in meteo conditions
«LBusinessProcess» T3.1 - Verify the correctness of the proposed sequence
«LBusinessProcess» T3.2 - Verification and application of separation criteria
«LBusinessProcess» T3.3 - TCC Request of changes in the sequence that reports to the COO that modifies manuelly the AMAN sequence

Figure 5 Impact on business process view

The change driven security engineering process relies on different state machines capturing the model states while they are updated. These state machines can be tailored to capture domain specific engineering processes. The state machines reflect the lifecycle of artefacts (e.g. state machine for the system model). Transitions between states define change handling and fire new change events to other models (e.g. evaluation of all security requirements causes the security objectives to change their states to evaluated). Figure 6 shows for instance a state machine capturing the different states for a requirements change.

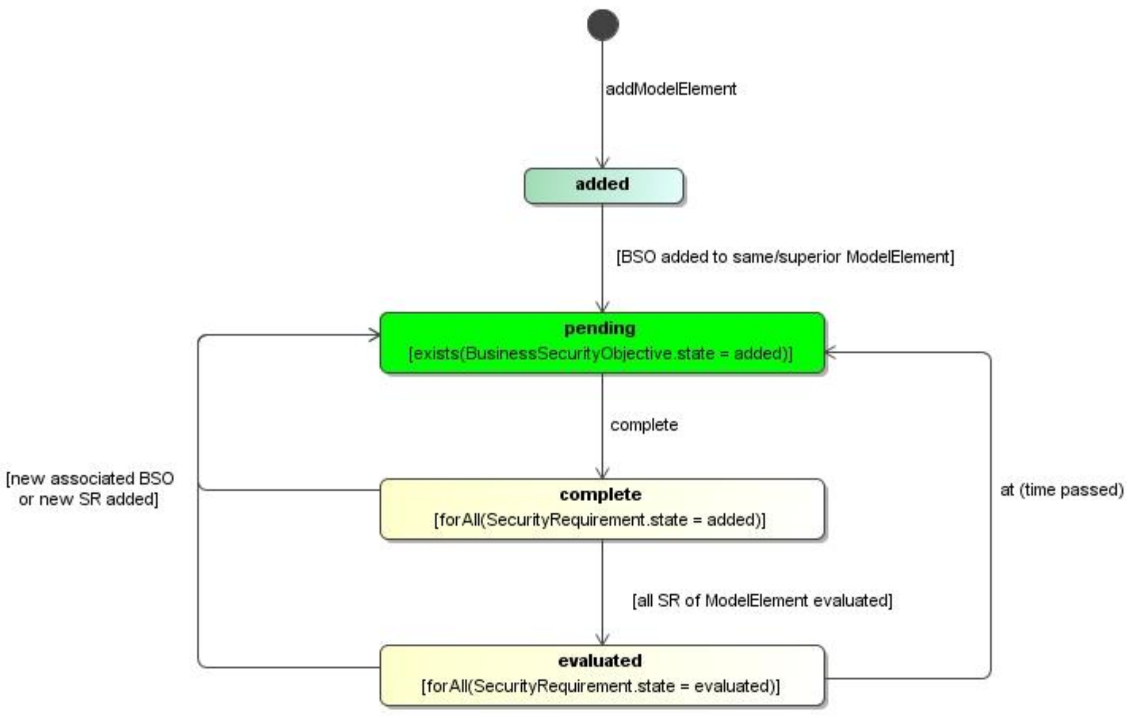
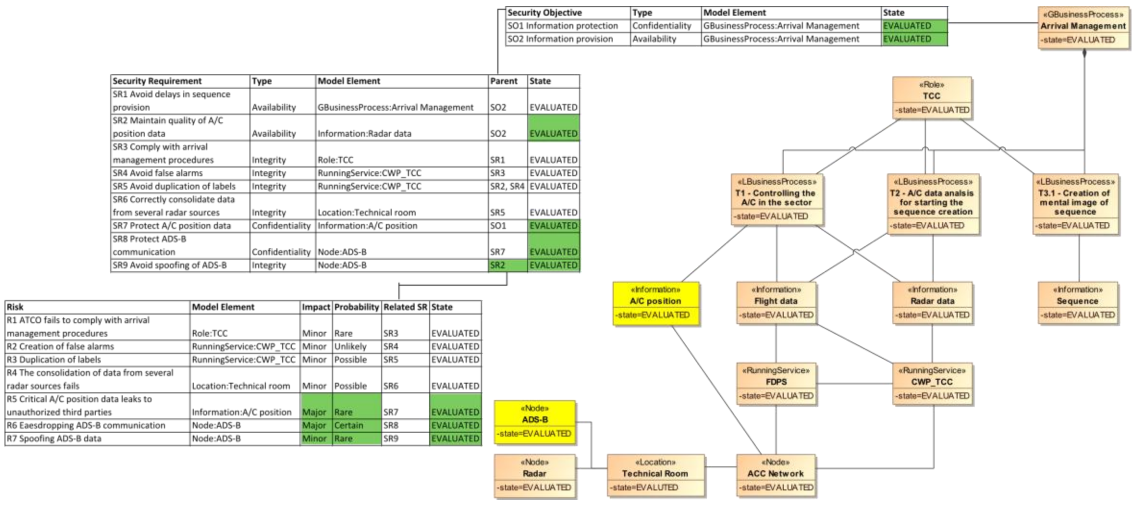

addModelElement
added
[BSO added to same/superior ModelElement]
pending
[exists(BusinessSecurityObjective.state = added)]
complete
complete
[forAll(SecurityRequirement.state = added)]
[all SR of ModelElement evaluated]
evaluated
[forAll(SecurityRequirement.state = evaluated)]
[new associated BSO or new SR added]
at (time passed)

Figure 6 Sample state machines for requirements changes

Figure 7 shows sample models (i.e. security requirements, architecture changes and risk estimates) for the ATM case study once all models are reassessed in order to take into account the introduction of the AMAN.



Security Objective	Type	Model Element	State
SO1 Information protection	Confidentiality	GBusinessProcess:Arrival Management	EVALUATED
SO2 Information provision	Availability	GBusinessProcess:Arrival Management	EVALUATED

Security Requirement	Type	Model Element	Parent	State
SR1 Avoid delays in sequence provision	Availability	GBusinessProcess:Arrival Management	SO2	EVALUATED
SR2 Maintain quality of A/C position data	Availability	Information:Radar data	SO2	EVALUATED
SR3 Comply with arrival management procedures	Integrity	Role:TCC	SR1	EVALUATED
SR4 Avoid false alarms	Integrity	RunningService:CWP_TCC	SR3	EVALUATED
SR5 Avoid duplication of labels	Integrity	RunningService:CWP_TCC	SR2, SR4	EVALUATED
SR6 Correctly consolidate data from several radar sources	Integrity	Location:Technical room	SR5	EVALUATED
SR7 Protect A/C position data	Confidentiality	Information:A/C position	SO1	EVALUATED
SR8 Protect ADS-B communication	Confidentiality	Node:ADS-B	SR7	EVALUATED
SR9 Avoid spoofing of ADS-B	Integrity	Node:ADS-B	SR2	EVALUATED

Risk	Model Element	Impact	Probability	Related SR	State
R1 ATCO fails to comply with arrival management procedures	Role:TCC	Minor	Rare	SR3	EVALUATED
R2 Creation of false alarms	RunningService:CWP_TCC	Minor	Unlikely	SR4	EVALUATED
R3 Duplication of labels	RunningService:CWP_TCC	Minor	Possible	SR5	EVALUATED
R4 The consolidation of data from several radar sources fails	Location:Technical room	Minor	Possible	SR6	EVALUATED
R5 Critical A/C position data leaks to unauthorized third parties	Information:A/C position	Major	Rare	SR7	EVALUATED
R6 Eaesdropping ADS-B communication	Node:ADS-B	Major	Certain	SR8	EVALUATED
R7 Spoofing ADS-B data	Node:ADS-B	Minor	Rare	SR9	EVALUATED

«GBusinessProcess» Arrival Management -state=EVALUATED
«Role» TCC -state=EVALUATED
«LBusinessProcess» T1 - Controlling the A/C in the sector -state=EVALUATED
«LBusinessProcess» T2 - A/C data analsis for starting the sequence creation -state=EVALUATED
«LBusinessProcess» T3.1 - Creation of mental image of sequence -state=EVALUATED
«Information» A/C position -state=EVALUATED
«Information» Flight data -state=EVALUATED
«Information» Radar data -state=EVALUATED
«Information» Sequence -state=EVALUATED
«RunningService» FDPS -state=EVALUATED
«RunningService» CWP_TCC -state=EVALUATED
«Node» ADS-B -state=EVALUATED
«Node» Radar -state=EVALUATED
«Location» Technical Room -state=EVALUTED
«Node» ACC Network -state=EVALUATED

Figure 7 Sample models in an evaluated state again

The walkthrough of the ATM case study highlights how the change driven security engineering copes with the identified challenges. In particular, it is fully driven by change events and change propagation. Change propagation is based on dependencies between artefacts. It provides an integrated view of the entire system, since change cannot be tackled from a single perspective anymore, and a tight integration of functional and security aspects. It defines clearly domains and responsibilities for the various stakeholders.

VALIDATION CRITERIA

The evaluation criteria of Applicability and Human Effort (as identified in deliverable D1.2 [3]) for the change driven security engineering process are detailed as follows:

- **Applicability**: The change-driven security engineering process can be applied to the ATM case study. We operate with the following increasing levels of fulfilment:
    - The change-driven process case study *can be conducted* by the researchers developing the methodology.
    - The report documenting the results of the case study *can be understood* by the relevant stakeholders.
    - The major principles of the change-driven process *can potentially be established* by a software provider.
    - The principles of the change-driven process *can be fully applied* by a software provider.
- **Human effort**: The second evaluation criterion is that the change-driven software engineering process can produce the desired results with less effort than by using alternative, traditional methods. We operate with the following increasing levels of achievement:
    - The steps of the security engineering process *are doable*, no matter the level of required human effort.
    - Handling a change request with the change-driven security engineering process is doable with the *same* level of human effort as traditional methods and/or manual approaches.
    - A change request can be handled with *significantly less* human effort than by using traditional methods and/or manual approaches.

VALIDATION RESULTS

The walkthrough of the ATM case study was evaluated by a feedback session and a questionnaire evaluation. Both activities allowed us to capture a rich set of information to evaluate the change driven security engineering process with respect to the ATM case study. The feedback session concluded that:

1. The change driven security engineering process supports properly capturing and analysing changes and evolutions of complex domains such as ATM. All the methodology steps are relevant to evolution.

2. The Change-Driven Security Engineering Process allows the modelling of complex security problems. However, it would be necessary to be tested on more complex problems and detailed pilot studies. This is because it is related to the fragmentation of problems and the required level of detail to model these aspects.

3. The change driven security engineering process captures ATM organizational settings and operational procedures. It would deal also with new systems and be related to system usages too.

4. It is clear how all the artefacts/steps of the Change-Driven Security Engineering Process are linked in a well-defined methodology. However, it is not very clear and easy to grasp how states change from one step to the other. Probably ways to better highlight the consequence of change handling step are needed.

5. The change driven security engineering process could be applied in the ATM domain with R&D, for industrial usage not clear yet. It would need more integration and further evolution.

A questionnaire evaluation followed the feedback session. The evaluation questionnaire consisted of six groups of evaluation criteria (four concerning in general SecureChange artefacts, i.e., Methodology, Modelling Language, Algorithm and Tool, one concerning the relevance for ATM domain and a final one tailored specifically for the WP). Figure 8 and Figure 9 show the median score for each evaluation group or criterion, respectively.

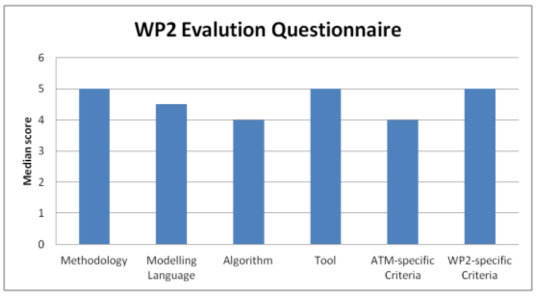


Figure 8 Median score grouped for each criterion category

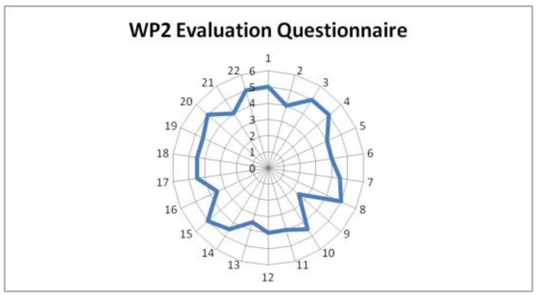


Figure 9 Median score for each evaluation criterion

The relevance for the ATM domain focused on the following criteria (**ATM Relevance**):

- The methodology/artefact complies with practices drawn from the Air Traffic Management (ATM) domain.
- The methodology/artefact can be easily understood and applied by ATM experts.
- The methodology/artefact deals with requirements changes drawn from ATM case study.
- The methodology/artefact deals with security properties drawn from the ATM case study.

The evaluation of specific WP's artefacts focused on three main criteria:

- **Applicability:** The change-driven security engineering process can be applied to the ATM case study.
- **Usability:** The change-driven security engineering process can produce the desired results with less effort that by using alternative traditional methodologies.

- **Tool Support:** The change-driven security engineering process is supported and demonstrated by a specific tool.

2.2.1.2 MoVE Tool

The MoVE Tool has been devised in order to support the change driven security engineering process and its main steps. It is a means to coordinate different state machines that relate to the status of different design artefacts (e.g. requirements models, risk analysis models). The main rationale is that different system artefacts that concern with the development environment are subject to changes. Different tools are often used in order to support developmental activities (e.g. requirements modelling, risk analysis). Different stakeholders are responsible for the management of such artefacts. Unfortunately, the coordination of artefacts and developmental activities is little supported. The MoVE tool addresses such aspects by:

- notifying stakeholders of relevant changes,
- fostering collaboration among stakeholders,
- supporting the analysis of aspects of systems that are affected by changes.

Figure 10 shows the conceptual architecture of the MoVE tool. The coordination among different artefacts is supported by specific adapters that capture the different states. The MoVE tool highlights changes and communicates them to stakeholders.

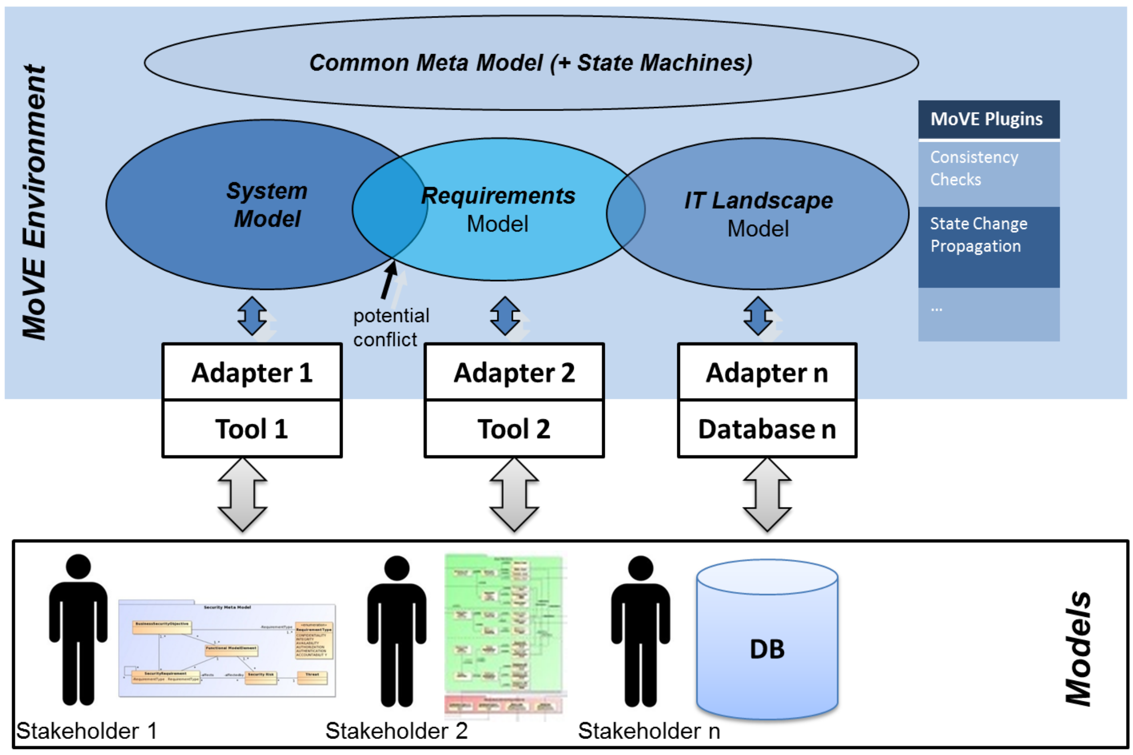
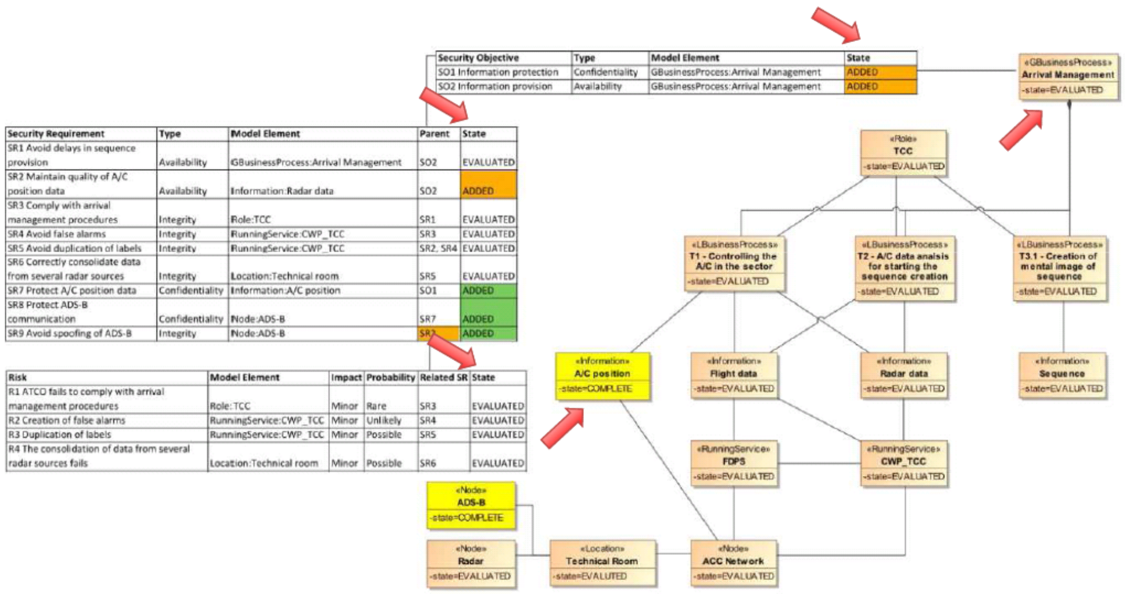


Figure 10 Conceptual Architecture of the MoVE Tool

VALIDATION SCENARIOS and EXERCISES

A dedicated workshop (Rome, September 2011) was organised in order to validate the implementation of the MoVE tool and its support for the change driven security engineering process. The validation activities consisted of a walkthrough user story supported by the MoVE tool of the selected changes requirements and security properties. The main aim was to assess and validate how the MoVE tool supports the different underlying aspects of the change driven security engineering process. In particular, how the MoVE tool supports the coordination and communication of changes across different but related developmental artefacts. Figure 11 highlights that it is necessary to coordinate stage changes in order to support the change driven security engineering process.



Security Objective	Type	Model Element	State
SO1 Information protection	Confidentiality	GBusinessProcess:Arrival Management	ADDED
SO2 Information provision	Availability	GBusinessProcess:Arrival Management	ADDED

Security Requirement	Type	Model Element	Parent	State
SR1 Avoid delays in sequence provision	Availability	GBusinessProcess:Arrival Management	SO2	EVALUATED
SR2 Maintain quality of A/C position data	Availability	Information:Radar data	SO2	ADDED
SR3 Comply with arrival management procedures	Integrity	Role:TCC	SR1	EVALUATED
SR4 Avoid false alarms	Integrity	RunningService:CWP_TCC	SR3	EVALUATED
SR5 Avoid duplication of labels	Integrity	RunningService:CWP_TCC	SR2, SR4	EVALUATED
SR6 Correctly consolidate data from several radar sources	Integrity	Location:Technical room	SR5	EVALUATED
SR7 Protect A/C position data	Confidentiality	Information:A/C position	SO1	ADDED
SR8 Protect ADS-B communication	Confidentiality	Node:ADS-B	SR7	ADDED
SR9 Avoid spoofing of ADS-B	Integrity	Node:ADS-B	SR[illegible]	ADDED

Risk	Model Element	Impact	Probability	Related SR	State
R1 ATCO fails to comply with arrival management procedures	Role:TCC	Minor	Rare	SR3	EVALUATED
R2 Creation of false alarms	RunningService:CWP_TCC	Minor	Unlikely	SR4	EVALUATED
R3 Duplication of labels	RunningService:CWP_TCC	Minor	Possible	SR5	EVALUATED
R4 The consolidation of data from several radar sources fails	Location:Technical room	Minor	Possible	SR6	EVALUATED

«GBusinessProcess» Arrival Management -state=EVALUATED
«Role» TCC -state=EVALUATED
«LBusinessProcess» T1 - Controlling the A/C in the sector -state=EVALUATED
«LBusinessProcess» T2 - A/C data anaisis for starting the sequence creation -state=EVALUATED
«LBusinessProcess» T3.1 - Creation of mental image of sequence -state=EVALUATED
«Information» A/C position -state=COMPLETE
«Information» Flight data -state=EVALUATED
«Information» Radar data -state=EVALUATED
«Information» Sequence -state=EVALUATED
«RunningService» FDPS -state=EVALUATED
«RunningService» CWP_TCC -state=EVALUATED
«Node» ADS-B -state=COMPLETE
«Node» Radar -state=EVALUATED
«Location» Technical Room -state=EVALUTED
«Node» ACC Network -state=EVALUATED

Figure 11 The MoVE Tool highlighting state changes

VALIDATION CRITERIA

The evaluation criteria of Applicability and Human Effort (as identified in deliverable D1.2 [3]) for the MoVE TOOL are detailed as follows:

- **Applicability**: The framework is applied to the Change Driven Security Process. We operate with the following increasing levels of fulfilment:
  - An implementation of the framework *is available and demonstrated* with academic examples.
  - An implementation of the framework *is available and is applicable* to the ATM case study.
  - The implementation of the framework *can be adopted* by relevant stakeholders and applied to their tool landscape.
  - The framework and its interfaces *are adopted* by software providers and further developed.
- **Human effort**: The second evaluation criterion is that a process supported by the MoVE Framework can produce the desired results with less effort than by using

alternative, traditional methods. We operate with the following increasing levels of achievement:

- The installation of the framework *enables* the implementation of a change driven security process.
- The installation of the framework *reduces* the communication and synchronization overhead, reducing human effort.

VALIDATION RESULTS

The WP2 ATM Validation Workshop (Rome, September 2011) was concerned with the following objectives:

1. Introduction on the change driven security engineering Process to ATM experts.

1. Demonstration of MoVE Tool support for the SecureChange Integrated Process.

2. Evaluation of the tool and results.

The management of models in a change driven process requires an effective set of networked tools. In the context of SecureChange we employ the infrastructure of MoVE (Modelling and Versioning Environment) that provides a framework to build such a tool support. Figure 12 shows a sample screenshot of the MoVE Tool.

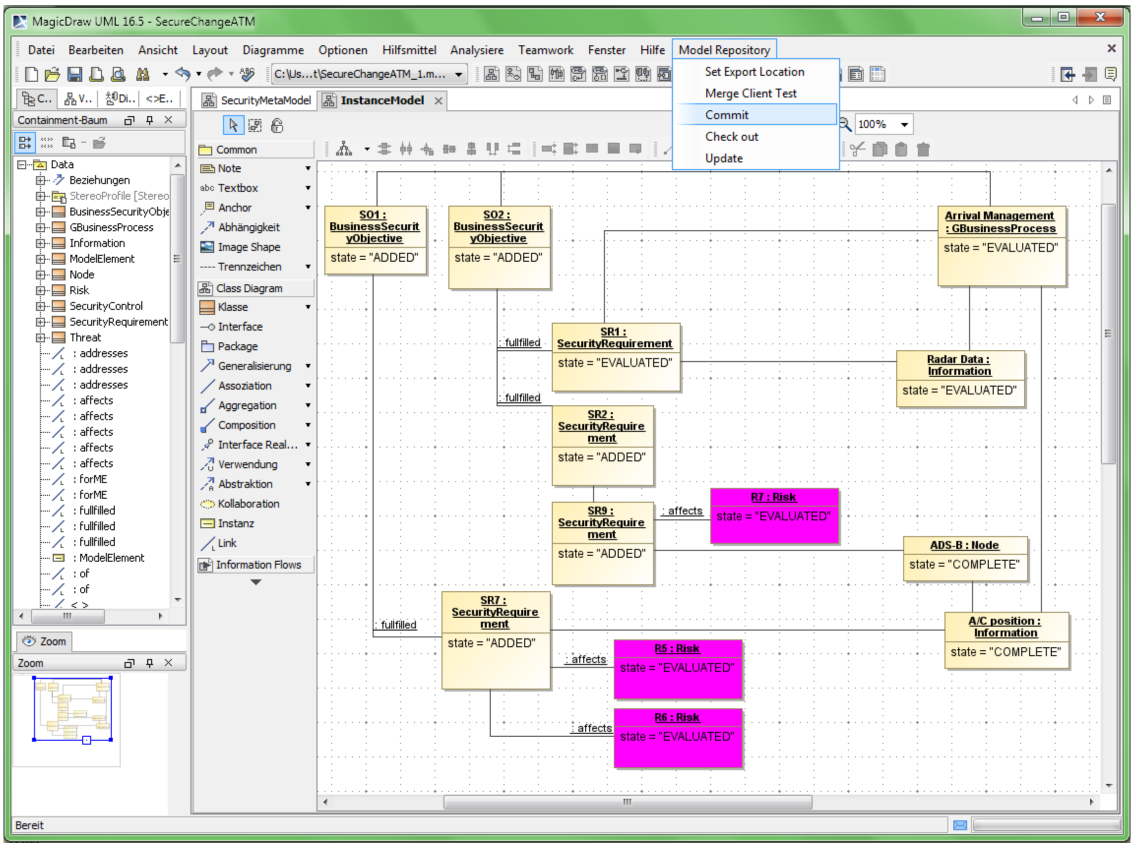

MagicDraw UML 16.5 - SecureChangeATM
Datei Bearbeiten Ansicht Layout Diagramme Optionen Hilfsmittel Analysiere Teamwork Fenster Hilfe Model Repository
Set Export Location
Merge Client Test
Commit
Check out
Update
SecurityMetaModel
InstanceModel
S01 : BusinessSecurityObjective state = "ADDED"
S02 : BusinessSecurityObjective state = "ADDED"
Arrival Management : GBusinessProcess state = "EVALUATED"
SR1 : SecurityRequirement state = "EVALUATED"
Radar Data : Information state = "EVALUATED"
SR2 : SecurityRequirement state = "ADDED"
SR9 : SecurityRequirement state = "ADDED"
R7 : Risk state = "EVALUATED"
ADS-B : Node state = "COMPLETE"
SR7 : SecurityRequirement state = "ADDED"
R5 : Risk state = "EVALUATED"
R6 : Risk state = "EVALUATED"
A/C position : Information state = "COMPLETE"
: fulfilled
: affects
Bereit

Figure 12 Screenshot of the MoVE Tool

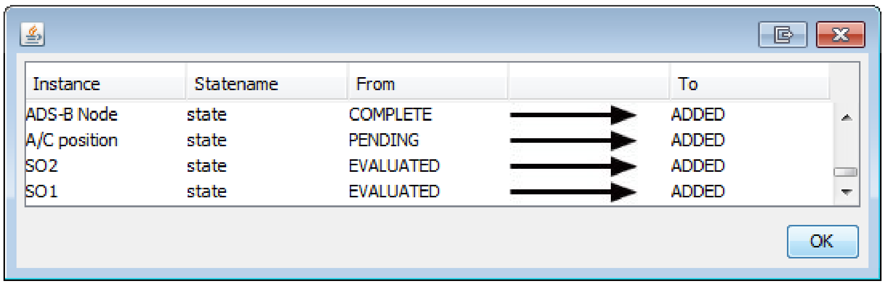
MoVE provides version control features leveraging classical subversion technology and extends it using Eclipse Modelling Framework (EMF) to provide sophisticated model versioning methods. To support the Living Security process the MoVE tool provides model element states and respectively state machines for model elements. State machines are implemented based on SCXML and OCL as a query and constraint language. Together with change propagation, state machines fulfil the need of a change-driven process. MoVE adapters allow tight integration and connection of heterogeneous modelling artefacts.

The user story of the walkthrough validation session followed specific steps in order to highlight the different functionalities of the MoVE tool and its support to the change driven security engineering process. The user story involved the following steps (in summary):

1. **Project Setup** – a script loads different models (e.g. UML system models, lists of requirements state machines) and information (e.g. configuration of MoVE plug-in). This created an initial repository with a basic set of models.

2. **Design Changes** – Checking the system model (Figure 13 shows an example of an UML system model that is modified) and adding two new elements (each with state ADDED): A/C position and ADS-B.

3. **Committing Changes** – The MoVE repository starts state machines and changes Business Security Objectives SO1 and SO2 to state ADDED according to state machine definition. This can be viewed in the commit log and in the state machine change window (Figure 13). Changing the state of A/C position and ADS-B to state PENDING and committing changes.



Instance	Statename	From		To
ADS-B Node	state	COMPLETE	→	ADDED
A/C position	state	PENDING	→	ADDED
SO2	state	EVALUATED	→	ADDED
SO1	state	EVALUATED	→	ADDED

Figure 13 Screenshot of the state machine change window

4. **Checking Security Requirements** – Switching to requirements model, adding two Security Requirements SR7 and SR8 and committing changes. Adding Security Requirement SR9 with parent SR2 and committing changes result in an additional change in SR2s state. System Designer is informed and changes the state of A/C position and ADS-B to state COMPLETE. Commit changes.

5. **Risk Assessment** – Add 3 new Risks R5 – R7 with State ADDED to model. Commit changes. Some Security Requirements change state to COMPLETE. Complete requirement definition and set state to EVALUATED. Commit changes.

6. **Changes Evaluated** – Previously added Security Requirements change their state to EVALUATED. Therefore SR2 changes to EVALUATED too. This triggers also a state change in SO1 and SO2. The two elements of A/C position and ADS-B change their state to EVALUATED.

After the evaluation session, we gathered feedback by questionnaire to collect structured feedback and brainstorming session to collect suggestions about the MoVE tool and the supported change driven security engineering process. The brainstorming session finally was useful to interpret some of the questionnaire outcomes. The following points summarise the final evaluation remarks for the MoVE tool and the change driven security engineering process. Figure 14 and Figure 15 show the median score for each evaluation group or criterion, respectively.

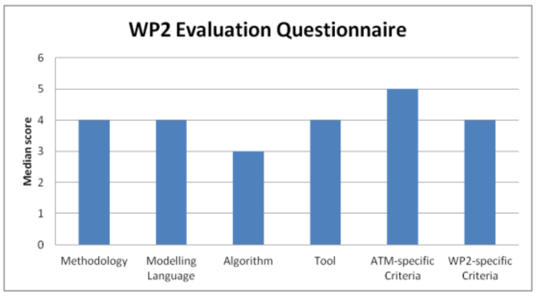


Figure 14 Median score grouped for each criterion category

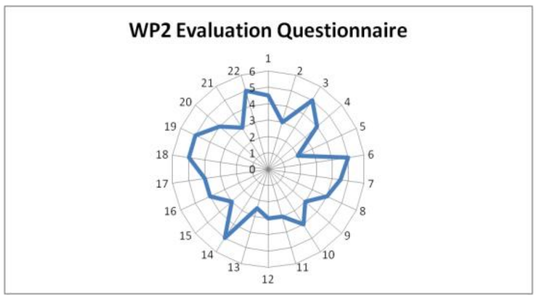


Figure 15 Median score for each evaluation criterion

2.2.2 Validation Remarks

The evaluation activities for the WP2 artefacts stress the following evaluation remarks:

1. It is critical the role of State Machines as a means to coordinate change management processes in order to safeguard critical security properties.

2. Different State Machines may support different project phases (e.g. requirements modelling, system design and risk assessment). However, tailoring such State Machines and building domain-specific adapters (e.g. adapters to integrate tools within the ATM domain) for different models require substantial effort.

2.3 WP3 Requirements

2.3.1 WP3 ARTEFACTS

The SecureChange Methodology for Evolutionary Requirements (SeCMER) supports: *Requirements Elicitation*, *Requirements Evolution* and *Argumentation Analysis*.

- **Requirements Elicitation.** The SeCMER's requirements elicitation step produces a requirements model (i.e. an instance of the SeCMER conceptual model), which combines concepts drawn from Problem Frames, SI* and security properties.

- **Requirements Evolution.** The SeCMER's requirements evolution step is concerned with detecting changes that might have an impact on the satisfaction of a security property. Such changes are detected by means of evolution rules that are event-condition-action rules. The event and condition part of a rule match a possible change in a requirement model while the action part specifies corrective actions to be applied to the requirement model.

- **Argumentation Analysis.** The SeCMER's argumentation analysis step checks that security properties are preserved by evolution and identifies new security properties that should be taken into account. The output of this phase is an argumentation model that provides evidence for denial of satisfaction of a security property.

Figure 16 shows the main steps supported by the SeCMER methodology.

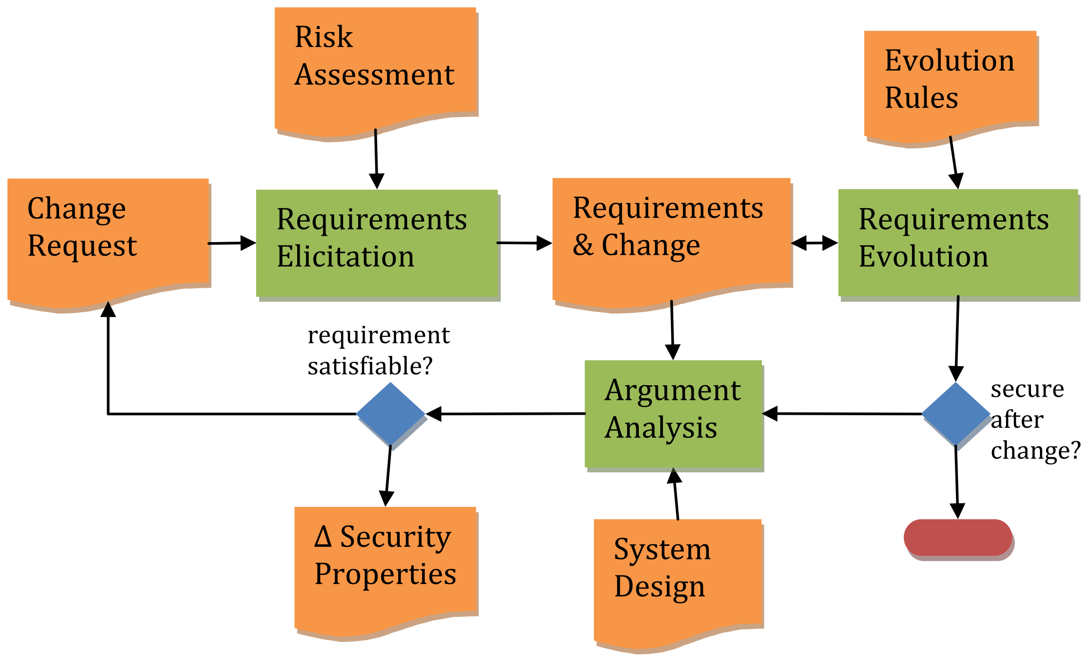


Figure 16 Overview of SeCMER

The validation activities for WP3 focused on the modeling of evolution as obervable and controllable rules as well as on the two main delivered artefacts:

1. SeCMER Methodology.

2. SeCMER Tool.

The SecMER methodology was validated by two dedicated workshops (Rome, April and July 2011) with ATM experts. The SecMER tool (and its support to the SecureChange Methodology for Evolutionary Requirements) was validated by a dedicated workshop with ATM experts (September 2011).

2.3.1.1 SeCMER Methodology

The validation activities of the SeCMER modelling methodology focused on the assessment of: modelling requirements evolution, estimating the impact of changes on requirements and supporting of the different phases. The SeCMER methodology consists of different modelling artefacts that combined together deal with requirements changes and security properties. The SeCMER methodology supports:

- modelling requirement evolution by different requirement methodologies, i.e. SI*, Problem Frames and SecMER ontology,
- automatically detecting requirement changes and violation of security properties by means of evolution rules,
- argumentation analysis to check security properties are preserved by evolution and to identify new security properties.

The main output from the methodology is therefore either an argumentation that system changes do not affect required security properties or a formulation of security properties that are satisfied by new system design. The first workshop (Rome, April 2011) concerned with the assessment of the SeCMER conceptual model. In particular, the validation activities focused on small-scale modelling exercises in order to assess the understandability of requirements evolution's representations. The second workshop (Rome, June 2011) concerned with the assessment of the different phases supported by the SeCMER methodology. The following sections report the validation results.

VALIDATION SCENARIOS and EXERCISES

The first workshop (Rome, April 2011) concerned with the assessment of a graphical representation of requirements evolution (i.e., of evolution rules for requirements). The workshop focused on the following points:

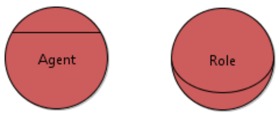
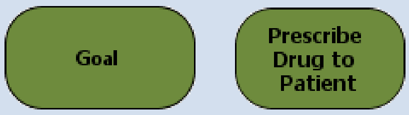
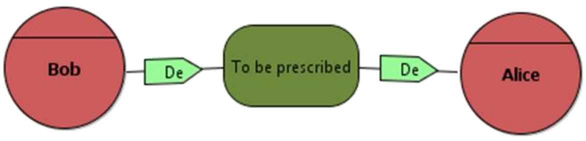
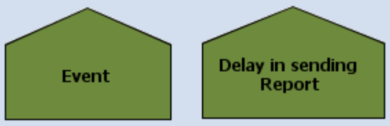
1. Stakeholders' desire of requirements evolution analysis – *what type of support do stakeholder expect to receive from an evolution analysis?*
2. Notation and graphical representation of evolution rules for requirements – *How to represent the observable evolution rules? Are the rules able to capture the mental model of evolution of the users?*
3. Preliminary elicitation of quantitative estimates (e.g. likelihood of changes) – *What kind of questions to ask stakeholders to get numbers (i.e. probabilities of evolutions)?*

The workshop involved a role-playing scenario in which different stakeholders were concerned with changes requirements. Two main roles took part in the workshop's validation activities:

- **Requirement engineers (modellers)** who are responsible to conduct elicitation activities (e.g. brainstorming sessions) with stakeholders about stakeholder's desires, and present graphical representation of rules in the designer workshop.
- **Business Stakeholders (Domain Experts)**, e.g. Air Traffic Controllers, who will evaluate and provide feedback on the graphical representations of evolution rules.

The workshop consisted of different sessions. The initial session introduced the general aspects of goal-oriented requirements engineering. In particular, the introduction provided a rationale for capturing requirements (and requirements evolution) in a structured way. It described the graphical notation of SI* and a general modelling requirements engineering process. Table 2 shows some examples of goal-oriented concepts and their representations in SI*. Examples drawn from the ATM case study in ATM were represented by an English description as well as a requirements model.

Table 2 Sample concepts and graphical representation in SI*

Agent/Role	
Goal	
Delegation	
Event	

The first workshop (Rome, April 2011) aimed at assessing alternative representations of requirements changes. In particular, the ATM case study was modelled in order to investigate how requirements evolution modelling would enhance reasoning about changes from a goal-oriented viewpoint, that is, how requirements changes would affect responsibilities from an operational viewpoint. Figure 17 shows an example of observable evolution rules stressing changes in terms of goals due to the introduction of the AMAN.

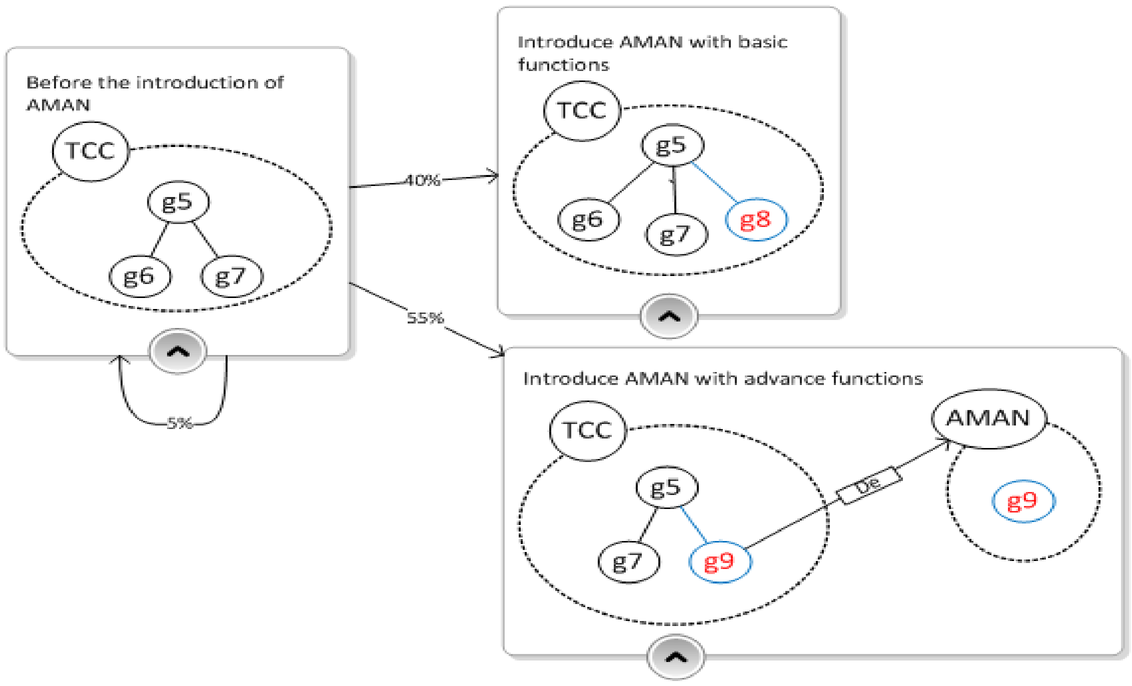
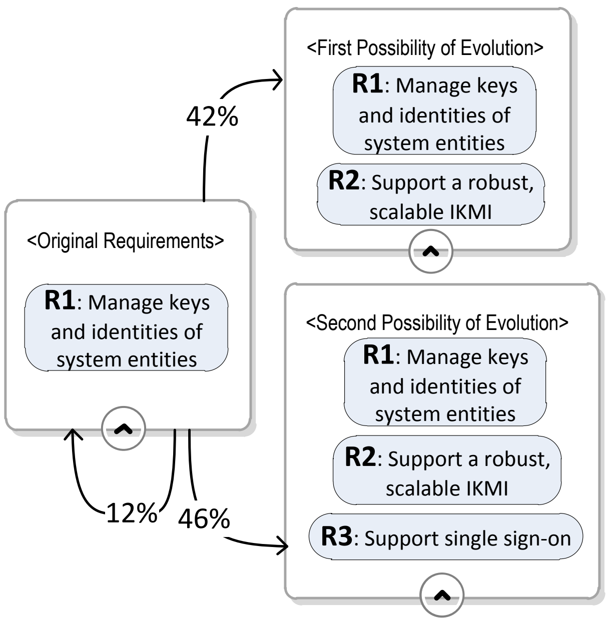

Before the introduction of AMAN
TCC
g5
g6
g7
5%
40%
55%
Introduce AMAN with basic functions
TCC
g5
g6
g7
g8
Introduce AMAN with advance functions
TCC
g5
g7
g9
De
AMAN
g9

Figure 17 Alternative goal-oriented changes due to AMAN introduction

Similarly, during the workshop, ATM experts (e.g. Air Traffic Controllers) assessed the representation of changes (in terms of goals), the likelihood of particular change scenarios and the representation of such chances. The combination of change representation and change likelihood is useful in order to capture evolution rules to be monitored. ATM experts pointed out that currently, in the whole ATM domain, an increasing interest is devoted to methodologies and processes supporting and documenting the decision making activities within the ATM developments. They have confirmed that change management and the need of a formal methodology to trace and assess the introduction of new operational concepts and their impact on ATM Key Performance Areas are among R&D problems. Currently, change management processes are supported by influence diagrams that allow to trace strategic objectives to operational solution and that allow to perform what if analysis to understand the impact of a proposed change. However, the Evolution Elicitation and Probability Estimation might be useful during the brainstorming phase to identify the alternative operational requirements associated with a proposed change. The reasoning phase instead can be used to support the decision makers in identifying the best solution at operational level to be implemented. The SeCMER methodology to model and reason on evolution would support managers and controllers during the change management process. Figure 18 shows another example of observable evolution rules.



<First Possibility of Evolution>
R1: Manage keys and identities of system entities
R2: Support a robust, scalable IKMI
42%
<Original Requirements>
R1: Manage keys and identities of system entities
12%
46%
<Second Possibility of Evolution>
R1: Manage keys and identities of system entities
R2: Support a robust, scalable IKMI
R3: Support single sign-on

Figure 18 Modelling evolution as a set of rules

Overall, during the ATM workshop for, the evaluation of modelling and reasoning on evolution highlighted that:

- The ATM experts pointed that Evolution Elicitation and Probability Estimation might be useful to identify the alternative operational requirements associated with a proposed change.
- Reasoning on evolution can be used to support decision makers in identifying the best solution at operational level to be implemented.

- Predicting the probability of evolutions is not trivial. The ATM experts suggested that an incremental approach should be adopted to identify all possible evolutions for a given before-evolution requirements model.

The second ATM workshop (Rome, June 2011) was useful to consolidate the validation of the SeCMER methodology and to get some initial feedback about the tool support. The validation activities involved the presentation of the SeCMER methodology to ATM experts and a walkthrough change scenario. The evaluation was concerned with each modelling artefact supporting the SeCMER methodology. Modelling examples captured the ATM case study. Figure 19, Figure 20 and Figure 21 show examples of requirements model, security pattern and argumentation analysis, respectively.

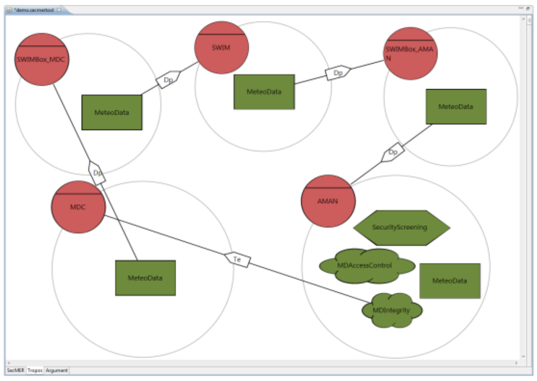


Figure 19 Example of requirements model

```
pattern
assetLeak(ConcernedActor,UntrustedActor,SecGoal,Asset) {
        find want(ConcernedActor, SecGoal);
        find securityGoal(SecGoal);
        find protect(SecGoal, Asset);
        find delegate(Concerned Actor, UntrustedActor, Asset);
        neg find trust(Concerned Actor, UntrustedActor, Asset);
}
```

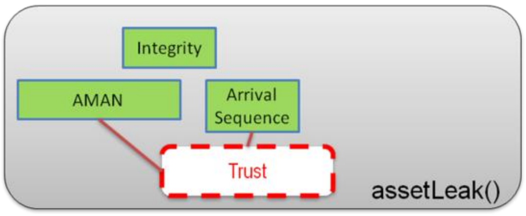
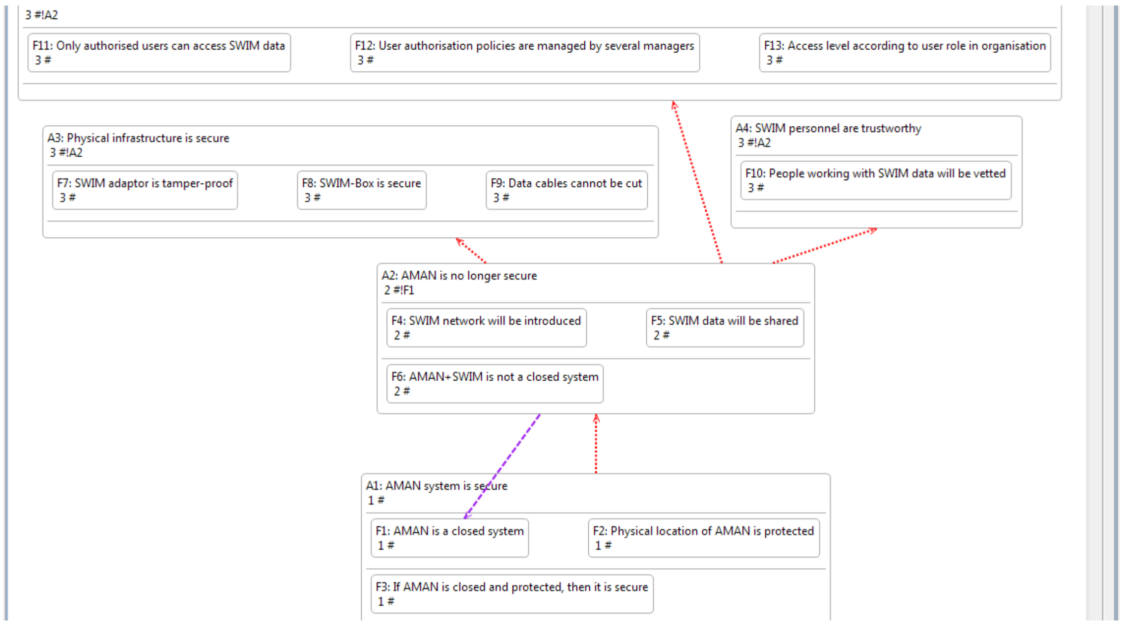


Figure 20 Example of security pattern



3 #!A2

F11: Only authorised users can access SWIM data
3 #

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

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

A3: Physical infrastructure is secure
3 #!A2

F7: SWIM adaptor is tamper-proof
3 #

F8: SWIM-Box is secure
3 #

F9: Data cables cannot be cut
3 #

A4: SWIM personnel are trustworthy
3 #!A2

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

A2: AMAN is no longer secure
2 #!F1

F4: SWIM network will be introduced
2 #

F5: SWIM data will be shared
2 #

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

A1: AMAN system is secure
1 #

F1: AMAN is a closed system
1 #

F2: Physical location of AMAN is protected
1 #

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

Figure 21 Example of argumentation analysis

VALIDATION CRITERIA

The evaluation criteria take into account two aspects of the SeCMER methodology: the modelling language and the overall methodology (supporting different phases, i.e. requirements elicitation, requirements evolution and argumentation analysis). The evaluation criteria of Applicability and Human Effort (as identified in deliverable D1.2 [3]) for the SeCMER methodology are detailed as follows:

SeCMER Modelling Language

- **Applicability:** The first evaluation criterion is that the SeCMER modelling language can be applied on the ATM case study for modelling and reasoning on evolving requirements.
    - Both functional and security requirements characterizing the introduction of the AMAN must be modelled using SeCMER concepts.
    - Evolution of requirements associated with the introduction of the AMAN must be modelled using SeCMER concepts.
    - The requirement models related to the introduction of the AMAN must be analyzable by using reasoning techniques.
    - The requirement modelling must be computer-aided.
- **Human effort:** The second evaluation criterion is that the modelling of changing requirements in the ATM case study can be conducted with less effort than by using state of the art requirements modelling languages or techniques.
    - The modelling of changing requirements using SeCMER methodology is doable.
    - The modelling of changing requirements using SeCMER methodology saves effort.

SeCMER Methodology

- **Applicability:** The first evaluation criterion is that the SeCMER methodology can be applied on the ATM case study for modelling and reasoning on evolving requirements. We can identify several sub criteria for the applicability to the ATM case study
    - The SeCMER methodology should consists of well defined, precise and easy to apply steps:
        - Each step can be understood and applied by the researcher.
        - Each step can be understood and applied by the stakeholder.
        - Each step can be understood and applied by the stakeholder, at least partially.
        - Each step can be understood and applied by the stakeholder, in complete independence.

- Explicit linkage of produced artefacts with SeCMER methodology steps.
- The methodology can be applied to the case study:
  - Can be done by the researcher.
  - Results can be understood by the stakeholder.
  - Can be done by the stakeholder, at least partially.
  - Can be done by the stakeholder, in complete independence.

- **Human effort:** The second evaluation criterion is that the SeCMER methodology can be applied to the ATM case study with less effort than other existing requirement engineering approached.
  - SeCMER methodology steps can be executed no matter the level of required human effort.
  - SeCMER methodology steps can be executed with the same level of human effort as traditional methods and/or manual approaches.
  - SeCMER methodology steps can be executed with (significantly) less human effort than by using traditional methods and/or manual approaches.

VALIDATION RESULTS

The evaluation of the SeCMER methodology by ATM experts highlighted the applicability of the methodology within the ATM domain. The ATM experts pointed out that the methodology can be applied to the ATM domain. However, they reported that the additional value of having the SecMER conceptual model needs to be better outlined. This was due to the fact that many concepts in the SeCMER conceptual model were new to the ATM experts. Figure 22 and Figure 23 show the median score for each evaluation group or criterion, respectively.

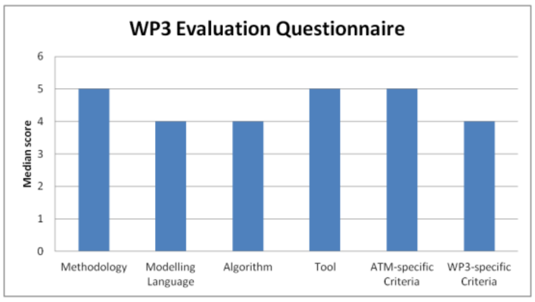


Figure 22 Median score grouped for each criterion category

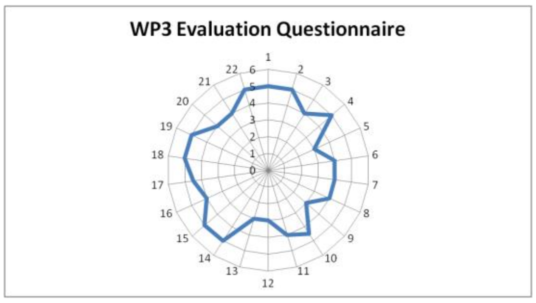


Figure 23 Median score for each evaluation criterion

2.3.1.2 SeCMER Tool

The SeCMER tool was validated in a dedicated workshop (Rome, September 2011) with ATM experts. The validation exercise involved a walkthrough change scenario of the ATM case study in order to present the different aspects of the SeCMER methodology and their tool implementation.

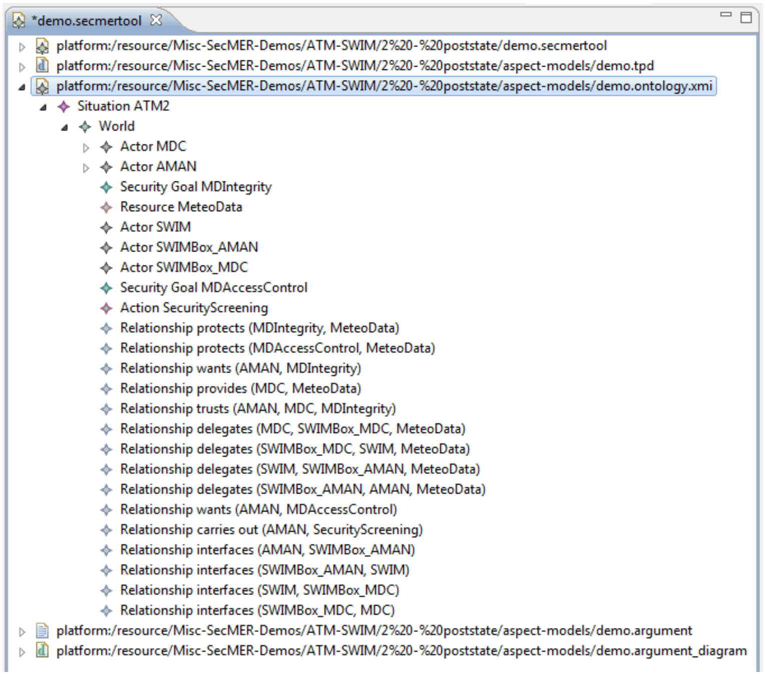
The SeCMER tool, implemented as an Eclipse plug-in, provides basic viewing and editing functionality to the integrated aspect models (Si*, abstract SeCMER model and Argument model currently). Additionally, the tool supports the following mechanisms:

- On-the-fly bi-directional synchronization between SeCMER and Si* representation of requirement models.
- Evolution rules detect violations of certain security patterns and offer automated solutions. Violations appear as Eclipse problem markers (of level WARNING). The suggested solutions appear as Quick Fix rules.
- Traceability is established between the argument and requirement models, requirement changes that make an argument obsolete can be automatically detected, and the user is notified by a message box.

Changes made in the abstract EMF representations (like the tree editor and the GMF Tropos Diagram) are transformed and synchronized between the SeCMER and Si* aspects on the fly. Textual formats (like the .ontology format of the SeCMER requirement model) are more detached: updates to and from them are only propagated upon saving. The walkthrough ATM change scenario was a means to present the different functionalities supported and implemented by the SeCMER tool. The implemented plug-in supports the different phases of SeCMER methodology.

VALIDATION SCENARIOS and EXERCISES

The different functionalities implemented and supported by the SeCMER tool were presented to ATM experts by means of a walkthrough change scenario drawn from the ATM case study. The SeCMER Tool supports Viewing and editing of requirements models. The SeCMER tab of the tree editor of the .secmertool EMF resource (see Figure 24) allows viewing the abstract representation of the contents of all associated models.



```
*demo.secmertool
platform:/resource/Misc-SecMER-Demos/ATM-SWIM/2%20-%20poststate/demo.secmertool
platform:/resource/Misc-SecMER-Demos/ATM-SWIM/2%20-%20poststate/aspect-models/demo.tpd
platform:/resource/Misc-SecMER-Demos/ATM-SWIM/2%20-%20poststate/aspect-models/demo.ontology.xmi
  Situation ATM2
    World
      Actor MDC
      Actor AMAN
      Security Goal MDIntegrity
      Resource MeteoData
      Actor SWIM
      Actor SWIMBox_AMAN
      Actor SWIMBox_MDC
      Security Goal MDAccessControl
      Action SecurityScreening
      Relationship protects (MDIntegrity, MeteoData)
      Relationship protects (MDAccessControl, MeteoData)
      Relationship wants (AMAN, MDIntegrity)
      Relationship provides (MDC, MeteoData)
      Relationship trusts (AMAN, MDC, MDIntegrity)
      Relationship delegates (MDC, SWIMBox_MDC, MeteoData)
      Relationship delegates (SWIMBox_MDC, SWIM, MeteoData)
      Relationship delegates (SWIM, SWIMBox_AMAN, MeteoData)
      Relationship delegates (SWIMBox_AMAN, AMAN, MeteoData)
      Relationship wants (AMAN, MDAccessControl)
      Relationship carries out (AMAN, SecurityScreening)
      Relationship interfaces (AMAN, SWIMBox_AMAN)
      Relationship interfaces (SWIMBox_AMAN, SWIM)
      Relationship interfaces (SWIM, SWIMBox_MDC)
      Relationship interfaces (SWIMBox_MDC, MDC)
platform:/resource/Misc-SecMER-Demos/ATM-SWIM/2%20-%20poststate/aspect-models/demo.argument
platform:/resource/Misc-SecMER-Demos/ATM-SWIM/2%20-%20poststate/aspect-models/demo.argument_diagram
```

Figure 24 Tree editor of the abstract model

The abstract model captures the following information:

- within the .secmertool file proper, the element Integration Model is responsible for gluing together the various other models, and contains traceability information,
- the abstract EMF model converted from the textual representation of the requirements model,
- the argumentation model associated with the requirements model, which is actually the abstract EMF model transparently parsed from the textual representation of the argumentation model,
- two subtrees contained in the Si* file: the abstract Tropos Model, and the abstract structure of the graphical diagram elements.

The tree editor can be used in conjunction with the Eclipse Properties View to manipulate the abstract form of the SeCMER Requirements model. The editor has two more tabs. The *Tropos tab* contains the Si* Tropos diagram editor to show the graphical representation of the Si* aspect (see Figure 25). This tab can be used as a regular Si* editor. Users typically perform most of the requirement modelling using this view, since modifications are automatically propagated between the abstract SeCMER requirement model seen in the tree editor, and the Si* requirements model. There are certain types of model elements, though, that are not represented in the Si* syntax. The most important is the *Protects relationship* between a security goal and an asset that can be created using the abstract tree editor. The *Argument tab* (see Figure 26) shows the argument diagram for the Argumentation model associated with the requirements model. The SeCMER tool allows the creation of new models by the New SeCMER model wizards (see Figure 27).

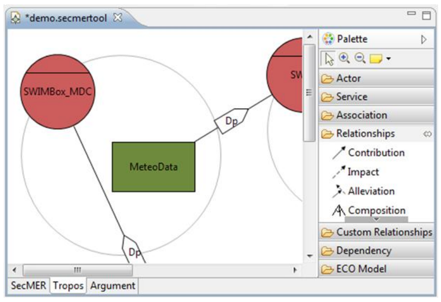


Figure 25 Si* diagram on the Tropos tab

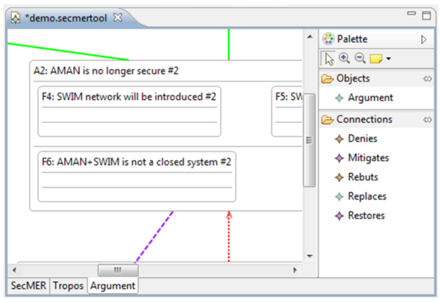


Figure 26 Argument diagram on the Argument tab

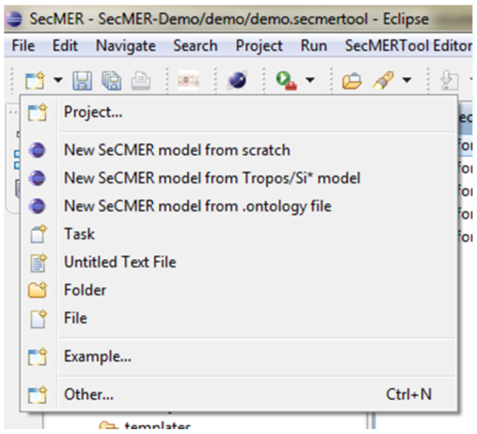
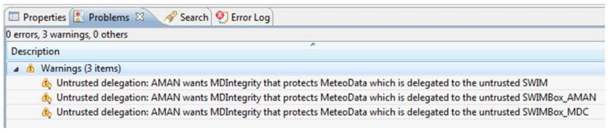

SecMER - SecMER-Demo/demo/demo.secmertool - Eclipse
File
Edit
Navigate
Search
Project
Run
SecMERTool Editor
Project...
New SeCMER model from scratch
New SeCMER model from Tropos/Si* model
New SeCMER model from .ontology file
Task
Untitled Text File
Folder
File
Example...
Other...
Ctrl+N
templates

Figure 27 New SeCMER Model wizards

The SeCMER editor allows the direct manipulation and export of Aspect models. Invoking the Export SeCMER model into .ontology file exports the requirements model into a selected file with the .ontology textual syntax. Invoking Export Tropos / Si* model will export the Si* aspect into the selected .tpd file. Finally, Export Argument Model extracts the argument model in the .argument textual syntax. The SeCMER tool allows the evaluation of some security properties based on the requirements models in a completely automated fashion. Detected security violations show up as a Warning marker in the "Problems" View of the Eclipse Workbench (see Figure 28). The SeCMER tool suggests default solutions or solution templates automatically. A dialog will appear listing possible ways to resolve the violation in question. Such solutions can be selected and executed automatically (see Figure 29).



0 errors, 3 warnings, 0 others

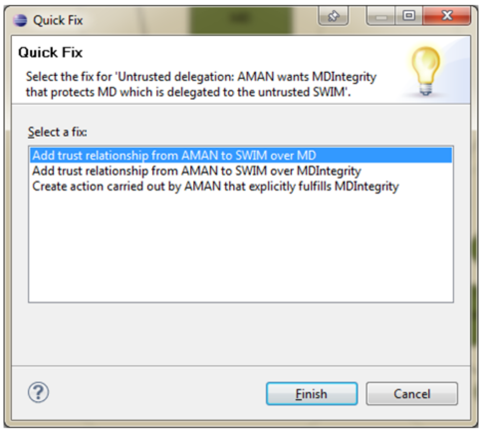
Description

Warnings (3 items)

Untrusted delegation: AMAN wants MDIntegrity that protects MeteoData which is delegated to the untrusted SWIM

Untrusted delegation: AMAN wants MDIntegrity that protects MeteoData which is delegated to the untrusted SWIMBox_AMAN

Untrusted delegation: AMAN wants MDIntegrity that protects MeteoData which is delegated to the untrusted SWIMBox_MDC



Quick Fix

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

Select a fix:

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

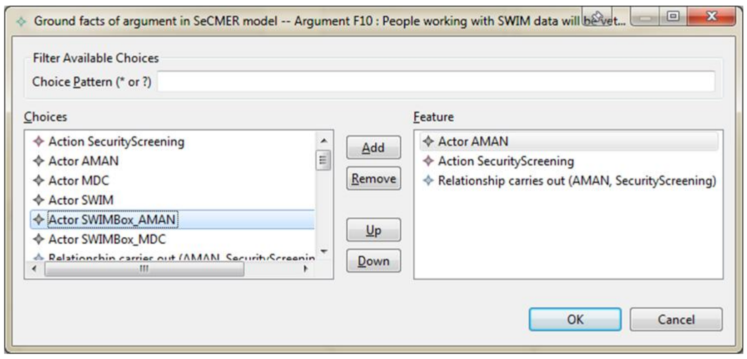
Figure 28 Detected security issues

Figure 29 Automatically suggested solutions

The SeCMER tool currently supports two kinds of traceability information that connect the Argument model and the Requirements model:

- The ground facts or evidence of an argument (typically an empty argument, i.e. fact) are Requirements model elements about which the argument states a proposition. Many arguments (typically the composite ones) don't have ground facts.
- Some top-level arguments may have supported goals in the requirements model. If the argument is valid, then the goals can be considered satisfied.

These traceability links can be established both in the Argument Diagram or supported goals in SeCMER model. Figure 30 shows a screenshot of the window for selecting ground facts and linking them to an argument.



Ground facts of argument in SeCMER model -- Argument F10 : People working with SWIM data will be vet...

Filter Available Choices

Choice Pattern (* or ?)

Choices

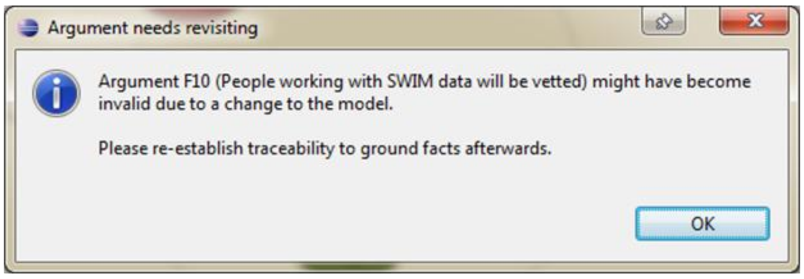
Action SecurityScreening
Actor AMAN
Actor MDC
Actor SWIM
Actor SWIMBox_AMAN
Actor SWIMBox_MDC
Relationship carries out (AMAN, SecurityScreenin

Feature

Actor AMAN
Action SecurityScreening
Relationship carries out (AMAN, SecurityScreening)

Figure 30 Selecting ground facts for an Argument

The point of maintaining traceability of supported goals is that reports of security violations are suppressed from the Problems View if the violated security goal is supported by an argument. In other words, manually verified arguments can override automatic problem detection. While argumentation is a powerful framework for early-stage analysis of security properties based on the requirements model, by default it considers a single state of the model. The challenge is in detecting arguments that have potentially been invalidated by changes, and revisiting these arguments to reflect the evolution, while no costly revision process is required for unaffected arguments. This is the purpose of keeping the traceability information on ground facts, so that model changes involving the ground facts may trigger a notification that alerts the user about the possibility that the argument may have become invalid due to the change. Figure 31 shows a sample screenshot of the warning window detecting an invalid argumentation.



Argument F10 (People working with SWIM data will be vetted) might have become invalid due to a change to the model.

Please re-establish traceability to ground facts afterwards.

Figure 31 Detecting the invalidation of an argument

VALIDATION CRITERIA

The evaluation criteria (as identified in deliverable D1.2 [3]) for the SeCMER TOOL are detailed as follows:

- Technical Usability:
  - Look and Feel.
  - Learnability/Memorability.
- User Acceptability.
- Human Effort.
- Presentation of Information.
- Domain Applicability:
  - The SeCMER CASE Tool can be used to model and analyse the case study:
    - Can be done by the researcher.
    - Results can be understood by the stakeholder.
    - Can be done by the stakeholder, at least partially.
    - Can be done by the stakeholder, in complete independence.
  - Additional knowledge or research is required to run the SeCMER CASE Tool.
  - The SeCMER CASE Tool cannot be used in the existing ATM software engineering processes.
  - The SeCMER CASE Tool can be used only with revising the existing processes.

  - The SeCMER CASE Tool can be used without major revision of the processes.
  - A tool for requirement evolution management is already used.
  - The SeCMER CASE Tool contributes to a better support for ATM evolution requirement management.
- Impact of a change can be assessed by the SeCMER CASE Tool.
- The SeCMER CASE Tool can present the analysis of the change in a usable format for end-users.

VALIDATION RESULTS

The evaluation workshop (Rome, September 2011) organised for the SeCMER tool evaluation gathered feedback by ATM experts, who assessed how the tool captures and supports each aspect of the methodology. One of the outcomes stressed the need to improve the user interface. A user-friendly interface would be critical in order to support users who might have received limited training on the SeCMER methodology. The SeCMER tool could be really useful as decision-support tool during the brainstorming phase of the change management process applied by Air Navigation Service Providers (ANSPs) to understand all the possible implications of changes. The evaluation outcomes allowed the identification of specific improvements, which have been implemented into the final version of SeCMER Tool [9]. The workshop feedback contributed to the improvement of the tool. Numerous changes were made to the tool at the request of the ATM experts, such as a customization of the Si* User Interface to fit SeCMER concepts and workflow. The changes include the following:

- An improved Graphical User Interface, with user-friendly features including wizards and export / import functionality. For instance, help functionality has been added on how to perform the main steps of SeCMER.

- A customized adaptation of the Si* diagram features to be more aligned with SeCMER concepts.

- An extended mapping between Si* and SeCMER to enable Si*-based modelling for all concepts involved in the Year 2 and Year 3 demonstration scenarios. This required that the concepts in the SI* interface to be renamed in order to avoid confusion between the SI* view and the SeCMER view.

- New Security Patterns have been added to cover further security issues, including violations of the least privilege principle.

- Added support for generating a dynamic list of quick fixes for a single security violation.

- Miscellaneous bug fixes (e.g. the fault in the saving project functionality has been fixed).

The implemented changes improved the overall usability of the SeCMER tool. Further feedback has been collected by means of evaluation questionnaires. Figure 32 and Figure 33 show the median score for each evaluation group or criterion, respectively.

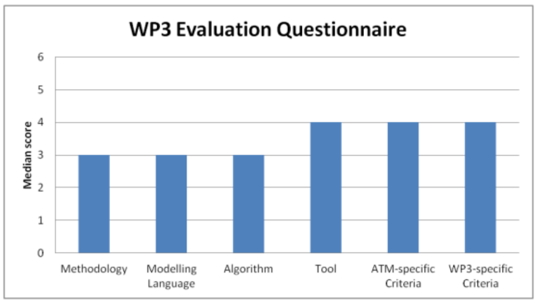


Figure 32 Median score grouped for each criterion category

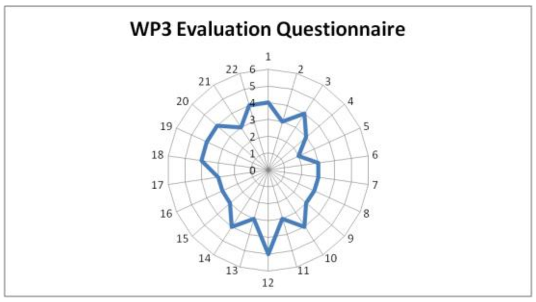


Figure 33 Median score for each evaluation criterion

2.3.2 Validation Remarks

The evaluation activities for the WP3 artefacts stress the following evaluation remarks:

1. The SeCMER methodology extends requirements engineering approaches (i.e. i* and Problem Frames) in order to deal with requirements changes and safeguard critical security properties.

2. The SeCMER methodology and tool require training in order to support capturing of domain knowledge and practice.

3. ATM experts highlighted an alternative use of the methodology/tool to support brainstorming sessions and gathering of requirements rather than modelling system entirely – large models tend to be too complex.

2.4 WP4 Model Design

2.4.1 WP4 ARTEFACTS

The validation activities for WP4 focused on the *Integration of Design Modelling Solutions*. SecureChange methodologies and artefacts support different development phases (e.g. requirements, design, risk analysis, implementation, testing). SecureChange methodologies support such development phases by modelling specific artefacts (e.g. requirements model, risk analysis model) and their evolutions. WP4, within the scope of the ATM case study, focused on the integration of such modelling artefacts for requirements, design and risk analysis. The different SecureChange modelling artefacts were presented to ATM experts in a focused workshop (Rome, September 2011). A general development process tailored to security served as a means to introduce the different modelling artefacts and their relationships.

2.4.1.1 Integration of Design Modelling Solutions

The ATM workshop (Rome, September 2011) for the WP4 Integration of Design Modelling Solutions involved a walkthrough scenario showing different development phases and the corresponding SecureChange modelling artefacts. The main emphasis was on the relationships between different modelling artefacts.

Figure 34 shows the engineering process presented during the ATM workshop in order to introduce the SecureChange modelling artefacts and to highlight their relationships.

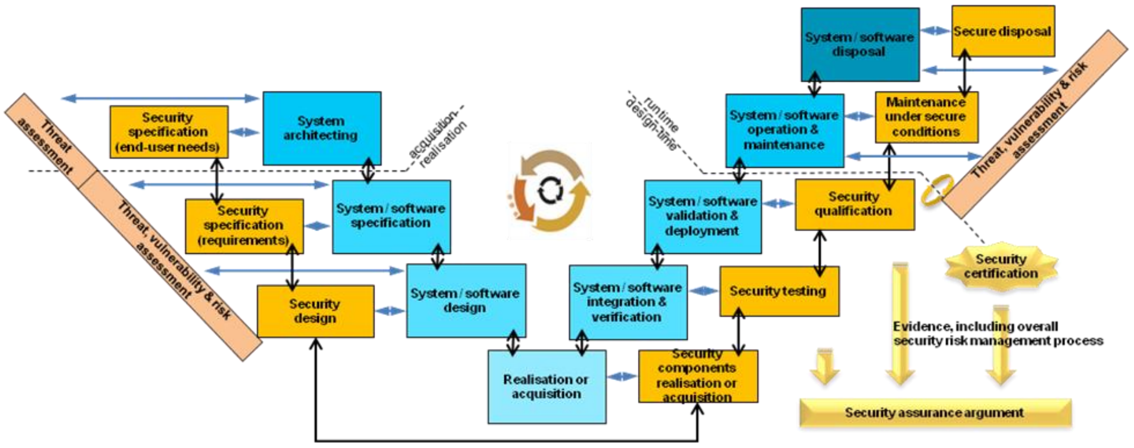

Threat assessment
Threat, vulnerability & risk assessment
Security specification (end-user needs)
System architecting
acquisition-realisation
Security specification (requirements)
System / software specification
Security design
System / software design
Realisation or acquisition
Security components realisation or acquisition
System / software integration & verification
Security testing
System / software validation & deployment
Security qualification
runtime design-time
System / software operation & maintenance
Maintenance under secure conditions
System / software disposal
Secure disposal
Threat, vulnerability & risk assessment
Security certification
Evidence, including overall security risk management process
Security assurance argument

Figure 34 An engineering process for security-critical systems

Note that the process is similar to the one adopted for ongoing developments in the ATM domain (the process is similar to the one adopted for the development of SWIM, System Wide Information Management [5]).

VALIDATION SCENARIOS and EXERCISES

The walkthrough scenario (presented during the validation workshop) involved the presentation of different SecureChange modelling artefacts developed for the ATM case study. The validation activity presented proprietary industry tools (developed and adopted by Thales) alongside SecureChange modelling artefacts in order to stress integration among different modelling artefacts and relevance to industry practices. In particular, the walkthrough engineering process highlighted the following engineering development phases and the relevant SecureChange modelling artefacts:

1. **Requirements Modelling** that relies on goal-oriented notations to specify multi-agent systems. Figure 35 shows a sample Si* architectural model that highlights system resources.

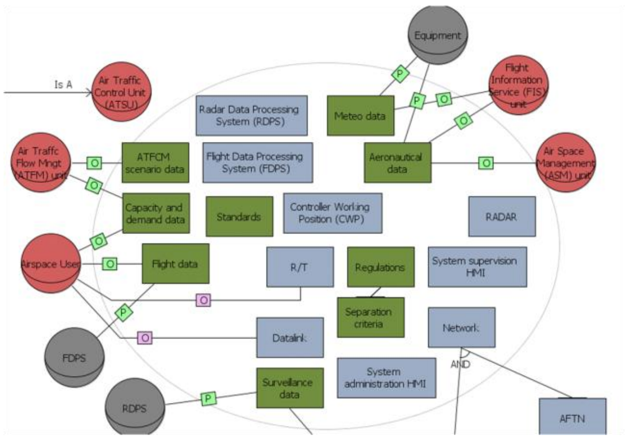

Equipment
Is A
Air Traffic Control Unit (ATSU)
Flight Information Service (FIS) unit
Radar Data Processing System (RDPS)
Meteo data
Air Traffic Flow Mngt (ATFM) unit
ATFCM scenario data
Flight Data Processing System (FDPS)
Aeronautical data
Air Space Management (ASM) unit
Capacity and demand data
Standards
Controller Working Position (CWP)
RADAR
Airspace User
Flight data
R/T
Regulations
System supervision HMI
Separation criteria
Network
Datalink
AND
FDPS
Surveillance data
System administration HMI
RDPS
AFTN
P
O

Figure 35 A sample Si* model capturing system resources

2. **Risk Assessment Modelling** in order to estimate the impact of changes. Figure 36 shows a sample risk assessment model that supports discussing emergent security threats due to changes.

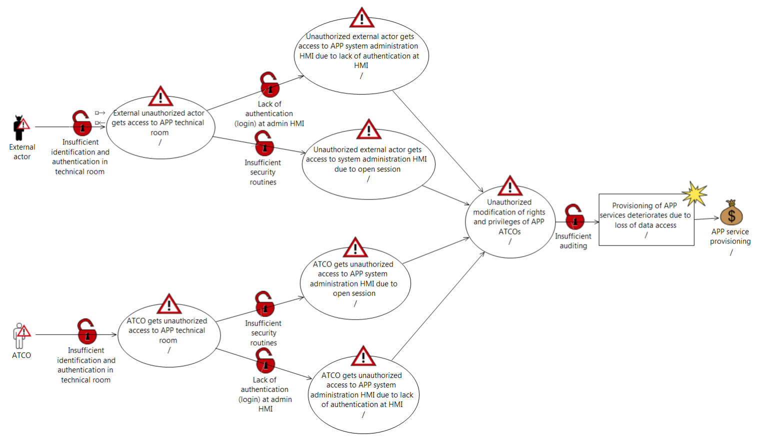

Unauthorized external actor gets access to APP system administration HMI due to lack of authentication at HMI
/
External actor
Insufficient identification and authentication in technical room
External unauthorized actor gets access to APP technical room
/
Lack of authentication (login) at admin HMI
Insufficient security routines
Unauthorized external actor gets access to system administration HMI due to open session
/
Unauthorized modification of rights and privileges of APP ATCOs
/
Insufficient auditing
Provisioning of APP services deteriorates due to loss of data access
/
APP service provisioning
/
ATCO gets unauthorized access to APP system administration HMI due to open session
/
ATCO
Insufficient identification and authentication in technical room
ATCO gets unauthorized access to APP technical room
/
Insufficient security routines
Lack of authentication (login) at admin HMI
ATCO gets unauthorized access to APP system administration HMI due to lack of authentication at HMI
/

Figure 36 A sample CORAS Risk Assessment model

3. **Security Design Modelling** in order to support design, implementation and testing activities. WP4 worked in particular in the adoption of UMLSec to support system specification.

In order to simulate the iterative nature of the engineering process, we have executed modelling exercises of the ATM case study before and after changes. Appendix E reports the modelling exercise.

VALIDATION CRITERIA

The evaluation criteria (as identified in deliverable D1.2 [3]) for the Integration of Design Modelling Solutions are detailed as follows:

- Effective Usage:
  - Overall well-defined system engineering process with clear steps and links.
  - Compliancy with already existing tools, standard and/or work-practices in the ATM domain.
  - Computer aided support for system modelling.
- Usability and Applicability:
  - The research technique can be applied on the ATM case study.
  - Results can be understood by the ATM domain expert.
  - Can be done by the ATM domain expert, at least partially.
- Required human effort:
  - Equivalent to manual approach.
  - Saves effort (in terms of time, workload and needed expertise).
  - Enhance the system models (providing further details and clearer modelling).

VALIDATION RESULTS

The modelling exercises (see Appendix E) highlight the Integration of Design Modelling Solutions with respect to an engineering process tailored to deal with security requirements and assurance. During the focused workshop with the ATM experts, a brainstorming and feedback session allowed us to identify specific comments concerning the integration of design modelling solutions:

- The process is assessed as being sound and relevant to their professional work in the ATM domain.
- The integration of the processes (i.e. security and engineering activities) is of value. Convergence of safety and security engineering practices is recommended.
- The responsibilities of each stakeholder in the process were unclear; this would be critical in order to integrate the engineering process within current organizational practices. The use of models in litigations was an explicit question. Specific attention should be set on the (legal) responsibilities related to changes.
- The current high tool diversity is an issue.

Figure 37 shows the outcome of the evaluation questionnaires collected during the evaluation ATM workshop. It highlights a general level of acceptability of the engineering process presented for the Integration of Design Modelling Solutions.

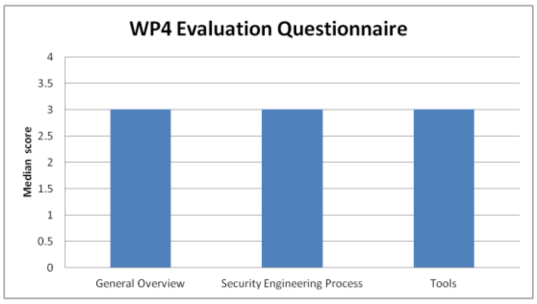


Figure 37 Median score grouped for each criterion category

2.4.2 Validation Remarks

The evaluation activities for the Integration of Design Modelling Solutions with respect to the ATM case study stress the following evaluation remarks:

1. Security Engineering Processes, Tools and Models extend development processes by focusing on security aspects.
2. The security-tailored engineering process intends to coordinate models supporting different development activities (e.g. requirements gathering, risk assessment, design modelling).
3. ATM experts highlighted that models and structured engineering processes might be useful in clearly allocating security responsibilities.

2.5 WP5 Risk Assessment

2.5.1 WP5 ARTEFACTS

The Risk Assessment WP contributed towards the developments of the CORAS risk assessment methodology, modelling language and tool in order to deal with SecureChange scenarios. The WP5 artefacts evaluated are:

1. The **Risk Assessment Language** and **Methodology** – CORAS – as its extension in order to deal with SecureChange objectives.
2. The **Risk Assessment Tool** supporting the CORAS language and methodology, that is, the notation supported by the model-driven risk analysis and the different steps forming the risk analysis methodology.

CORAS [6] is an approach to risk analysis that consists of three tightly integrated parts, namely the CORAS method, the CORAS language and the CORAS tool. The method is based on the ISO 31000 risk management standard [7] and consists of eight steps. The four first steps correspond to the context establishment, whereas the remaining four are risk identification, risk estimation, risk evaluation and risk treatment. The method comes with concrete tasks and practical guidelines for each step, and is supported by several risk analysis techniques. The CORAS language consists of five kinds of diagrams, each of which provides support for specific tasks throughout the whole risk assessment process. The method is supported by the tool, which is an editor for on-the-fly risk modelling. The most important kind of CORAS diagram is threat diagrams which are used for risk identification and risk estimation. The language constructs are firmly based on an underlying well-defined conceptual framework for reasoning about risk, and includes: human and non-human threats, vulnerabilities, threat scenarios, unwanted incidents and assets. Threat diagrams are used for on-the-fly risk modelling during structured brainstorming that involves personnel with expert knowledge about the target of analysis. In such a setting, the diagrams must be intuitive and easy to understand, also for people with little technical background and little experience in risk analysis. For this reason, the CORAS language constructs are graphical, easily understandable symbols. Figure 38 shows the symbols for the main CORAS concepts.

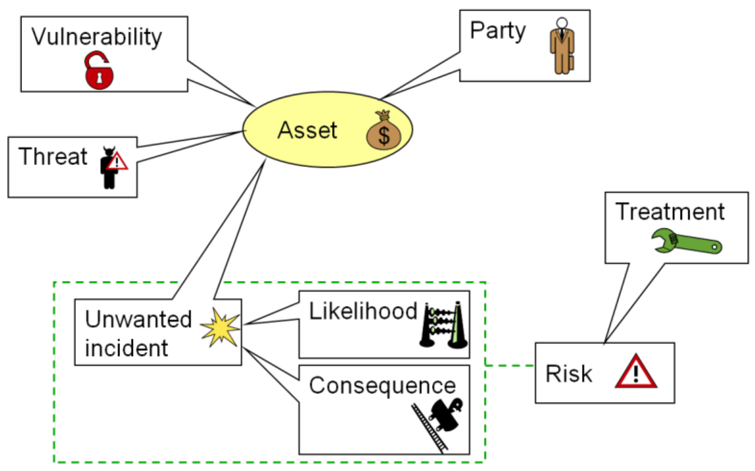
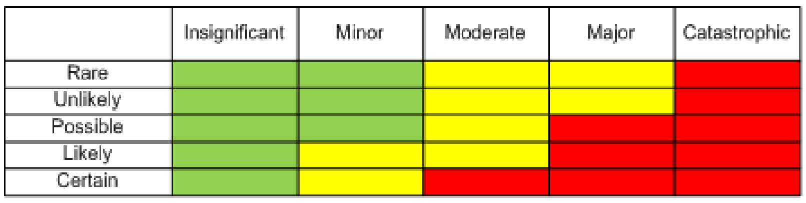


Figure 38 Main CORAS concepts

CORAS has been generalized to provide specialized support for assessing risks of changing and evolving systems. When systems change, so do risks and therefore need to be modelled and analyzed as such. In the following we describe selected parts of the generalized CORAS approach and exemplify with parts of an ATM case study. We focus on the identification and modelling of changing risks since this is the core part of the process.

CONTEXT ESTABLISHMENT. The context establishment includes making the target description, setting the focus and scope of the analysis, identifying the assets, and setting the risk evaluation criteria. In the setting of evolving systems, the context establishment moreover includes the specification of the changes to the target, the changes in assets or asset values, and the changes to the evaluation criteria, if any. When making the target description we need to describe as precisely as possible at a suitable level of abstraction the structure and behaviour of the target of analysis. This includes the actors, roles and components, the work processes and interactions, the interface with the environment, and so on. A well-understood and suitable modelling language should be used for this purpose, for example the UML. For changing and evolving systems we need first to make a target description of the system as is. Then the relevant change requirements must be described as precisely as possible. Using the target description as-is (before changes) and the specification of the change requirements, the target description of the system to-be (after changes) is made.

The context establishment also involves identifying and documenting the assets with respect to which the risk analysis is to be conducted. An asset is something of value that must be protected, and can, for example, be integrity of information, privacy and human life and health. In the ATM example the assets were based on the identified security properties of Information Protection (confidentiality) and Information Provision (availability). For changing and evolving systems, we furthermore need to identify any possible changes to the assets, for example new assets that emerge or changes in asset values or priorities. A further part of the context establishment is to define the scales for likelihood and consequence, the values of which are used during risk estimation for determining risk levels. In the ATM example we based the scale on the EUROCONTROL safety regulatory requirement (ESARR4) [8]. Figure 39 shows the risk evaluation criteria.



	Insignificant	Minor	Moderate	Major	Catastrophic
Rare	🟩	🟩	🟨	🟨	🟥
Unlikely	🟩	🟩	🟨	🟨	🟥
Possible	🟩	🟩	🟨	🟥	🟥
Likely	🟩	🟨	🟨	🟥	🟥
Certain	🟩	🟨	🟥	🟥	🟥

Figure 39 Risk evaluation criteria

RISK IDENTIFICATION. Risk identification using CORAS is conducted as a structured brainstorming involving personnel with firsthand knowledge about the target of analysis. By conducting a walkthrough of the target description, risks are identified by systematically identifying unwanted incidents, threats, threat scenarios and vulnerabilities. The results are documented by means of CORAS threat diagrams. So far, the methods and techniques are as for traditional risk analyses. When dealing with change, a guiding principle for the generalized risk analysis method is that only the risk analysis results that are affected by the system changes should be assessed again.

We therefore provide techniques and language support for tracing changes from the target system to the risk model so as to enable the identification of the parts of the risk models that are not affected by changes and therefore maintain their validity under change. Figure 40 shows a fragment of a CORAS threat diagram resulting from the identification of changing risks.

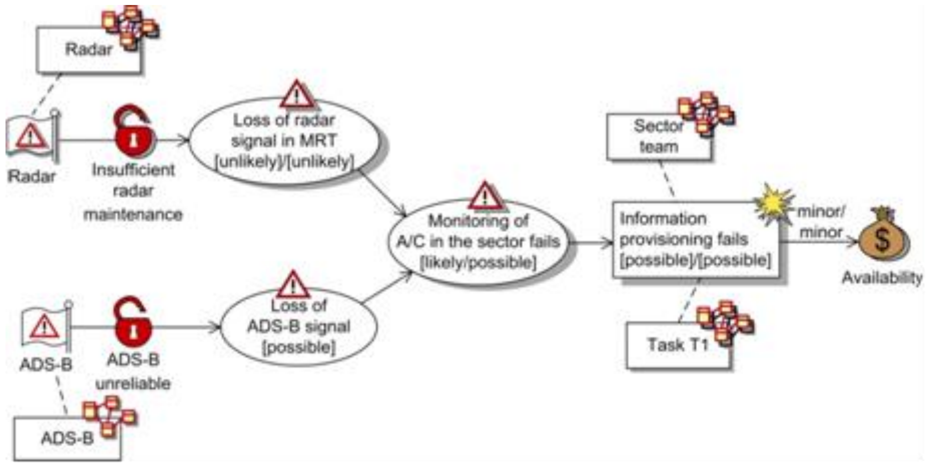

Radar
Radar
Insufficient radar maintenance
Loss of radar signal in MRT [unlikely]/[unlikely]
Monitoring of A/C in the sector fails [likely/possible]
Sector team
Information provisioning fails [possible]/[possible]
minor/ minor
Availability
Task T1
ADS-B
ADS-B unreliable
Loss of ADS-B signal [possible]
ADS-B

Figure 40 Threat diagram with changing risks

Compared with the standard CORAS language, there are two main language extensions to support the risk analysis of evolving systems. First, the rectangle icons with the system diagram symbol (e.g. the one named Task T1 – the first task in the arrival management work process) exemplify the new construct for referring to the target of analysis. Second, the threat diagram constructs of threat, unwanted incident, asset, etc. are generalized to three modes with different appearances, namely the modes before, after and before-after. The before constructs are in gray shade and dashed outline and represent parts of the risk picture that are valid only before the changes. The after constructs are in colour and solid outline and represent parts that are valid only after the changes. The before-after constructs are two-layered and represent parts that are valid both before and after changes. The explicit references to the target system in the threat diagrams facilitate the identification of the parts of the risk picture that are affected by system changes. For example, in the ATM risk analysis, the radar was not subject to the ATM system changes. Hence, the vulnerability, *Insufficient radar maintenance*, and the threat scenario, *Loss of radar signal in MRT* (multi-radar tracking), are maintained under change. The threat scenario Monitoring of A/C (aircraft) in the sector fails, on the other hand, is affected due to the introduction of the ADS-B (automatic dependent surveillance-broadcast).

The different appearance of the three modes of the language constructs facilitates the immediate recognition of the risk changes that are modelled. This feature is an important part of supporting the risk identification brainstorming and for appropriately documenting the results. Furthermore, in order to highlight the risk changes, the CORAS tool implements the functionality of changing between the views of before, after and before-after. This is illustrated in Figure 41 with the before view to the left and the after view to the right. An important feature of our generalized CORAS language is of course the support for giving a combined representation of the risks before and after changes as shown in Figure 40.

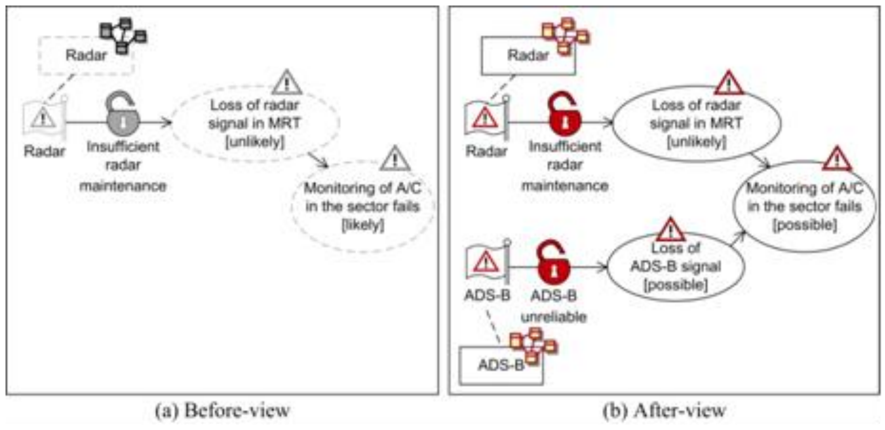

Radar
Radar
Insufficient radar maintenance
Loss of radar signal in MRT [unlikely]
Monitoring of A/C in the sector fails [likely]
(a) Before-view
Radar
Radar
Insufficient radar maintenance
Loss of radar signal in MRT [unlikely]
Monitoring of A/C in the sector fails [possible]
Loss of ADS-B signal [possible]
ADS-B
ADS-B unreliable
ADS-B
(b) After-view

Figure 41 Two views on changing risks

RISK ESTIMATION. The risk estimation basically amounts to estimating likelihoods and consequences for unwanted incidents. Usually, we also estimate likelihoods for threat scenarios in order to get a better basis for estimating the likelihood of unwanted incidents and to understand the most important sources of risks. To the extent that risks before changes are completely unaffected by the changes, the risk estimates need not be conducted twice for these risks. Diagram elements of mode before-after are assigned a pair of likelihoods. The former denotes the likelihood before the changes. The latter denotes the likelihood after the changes. Diagram elements of mode before or mode after are assigned only a single likelihood. The distinction is likewise for the consequence estimates. Hence, the threat diagrams document not only risks that emerge, disappear or persist, but also how risk levels change. For example, the threat scenario Monitoring of A/C in the sector fails is assigned the likelihood likely before the changes and the likelihood possible after the changes. The likelihood drops due to the introduction of the ADS-B. Information provisioning fails is an unwanted incident, and therefore constitutes a risk. Its likelihood is possible both before and after the changes, while its consequence for the Availability asset is minor as annotated on the relation between the unwanted incident and the asset.

RISK EVALUATION. During the risk evaluation we first calculated the risk levels by using the risk matrix exemplified in Figure 1 and the likelihood and consequence estimates from the risk estimation. We then compare the risk levels with the risk evaluation criteria to determine which risks that must be treated or evaluated for treatment. The risk estimation is supported by CORAS risk diagrams which we do not show here as the focus is on threat diagrams and risk identification. Risk diagrams show the changing risks together with the threats that initiate them and the assets they harm. The unwanted incident Information provisioning fails, for example, has the likelihood possible and the consequence minor before and after the ATM system changes, which yields a low risk level.

RISK TREATMENT. The purpose of the risk treatment is to identify options for risk mitigation for the unacceptable risks. In the setting of evolving systems, the treatments should ensure that an acceptable level of risk is maintained under planned changes or foreseen evolutions. This final step of the process is conducted as a structured brainstorming with a walkthrough of the threat diagrams documenting the unacceptable risks. The task is supported by CORAS treatment diagrams.

The WP5 Risk Assessment artefacts have been evaluated as follows:

- Workshop (Rome, June 2011) with ATM Experts to present, analyse and review the Risk Assessment Methodology. Possible foreseen exploitation in the SESAR Programme.
- Technical and operational workshop (Rome, June 2011) to evaluate the completeness, expressibility and flexibility of the Risk Modelling Language.
- Interactive Demo of the WP5 Prototype and final tool (Rome, September 2011). Some simple modeling activities carried out by ATM experts with the support of WP5 technical partners.
- Off-line questionnaire evaluation by ATM stakeholders of WP5 Risk Assessment Methodology and Tool description to collect feedback about the applicability and effectiveness of WP5 artefacts in ATM.

The remainder of this section reports the evaluation results of the CORAS Risk Assessment Language, Methodology and Tool.

2.5.1.1 Risk Assessment Language and Methodology

VALIDATION SCENARIOS and EXERCISES

The validation activities concerning the CORAS risk assessment language and methodology involved a dedicated workshop (Rome, June 2011) with ATM experts. The workshop consisted of a walkthrough ATM change scenario in order to perform a risk assessment supported by the CORAS methodology. The Risk Assessment Language and Methodology were further evaluated by the offline collection of questionnaires. The CORAS methodology and the requirements changes were presented together with relevant questionnaires (collected in the period November-December 2011, concerning with Safety Culture, Artefact Evaluation and Evolutionary Risk Analysis) to ATM experts. This allowed us to question any relationship between ATM safety cultures and validation aspects of the CORAS Risk Assessment Language and Methodology with respect to the requirements changes.

VALIDATION CRITERIA

The main case study in WP5 is ATM. It is therefore the ATM case study that provides the most thorough basis for the evaluation.

- Effective Usage: The criteria of effective usage of the artefacts require that the artefacts can be applied in the ATM case study. We provide evaluation criteria for applicability and for the required human effort. The degree of fulfilment is given by categorizing the level of achievement of applicability and effort.

Risk Assessment Methodology

- Applicability: The first evaluation criterion is that the risk assessment methodology and its techniques can be applied on the ATM risk assessment. We operate with the following increasing levels of fulfilment:

- The ATM risk assessment can be conducted by the researchers developing the methodology.
    - The report documenting the results of the case study can be understood by the relevant ATM stakeholders such as the external risk assessment participants.
    - The ATM risk assessment can be conducted only partially by a risk analyst trained in traditional risk assessment methods.
    - The ATM risk assessment can be fully conducted by a risk analyst in complete independence.
- Human effort: The second evaluation criterion is that the risk assessment methodology and its techniques can produce the desired results with less effort than by using alternative, traditional methods. We operate with the following increasing levels of achievement:
    - Conducting the ATM risk assessment is doable, no matter the level of required human effort.
    - Conducting the ATM risk assessment is doable with the same level of human effort as traditional methods and/or manual approaches.
    - The ATM risk assessment can be conducted with (significantly) less human effort than by using traditional methods and/or manual approaches.

Risk Assessment Language

- Applicability: The first evaluation criterion is that the risk modelling language can be applied on the ATM case study for modelling and assessing changing risks. We operate with the following increasing levels of fulfilment:
    - The consistent and syntactically correct modelling, as well as the semantically correct interpretation, of the ATM risk models can be conducted by the researchers developing the risk modelling language.
    - The ATM risk models can be understood by the relevant stakeholder both during the risk identification and assessment, and as part of the documentation of the results.
    - The consistent and syntactically correct modelling, as well as the semantically correct interpretation, of the ATM risk models can be conducted only partially by a risk analyst trained in traditional risk modelling.
    - The consistent and syntactically correct modelling, as well as the semantically correct interpretation, of the ATM risk models can be conducted by a risk analyst in complete independence.
- Human effort: The second evaluation criterion is that the modelling of changing risks in the ATM case study can be conducted with less effort that by using traditional risk modelling languages or techniques. We operate with the following increasing levels of achievement:
    - Conducting the modelling of changing risks is doable, no matter the level of required human effort,
    - Conducting the modelling of changing risks is doable with the same level of human effort as using traditional risk modelling languages or techniques,
    - The modelling of changing risks can be conducted with (significantly) less human effort than by using traditional risk modelling languages or techniques.

Similar validation criteria apply to the Tool supporting the Risk Assessment Language and Methodology.

VALIDATION RESULTS

The first validation workshop (Rome, June 2011) concerned with the evaluation of the CORAS risk analysis language and methodology. The main aims were to present how the CORAS approach deals with requirements changes and to assess the concepts underlying the methodology. CORAS models and related requirements changes drawn from the ATM case study were presented to ATM experts. The risk identification and risk estimation make active use of CORAS threat diagrams. These diagrams support the identification, modelling and documentation of unwanted incidents, the assets that are harmed, the threats that initiate unwanted incidents, the threat scenarios that are initiated by threats and lead to unwanted incidents, as well as the vulnerabilities that are exposed. Because the CORAS threat diagrams are firmly based on an underlying conceptual framework for reasoning about risk, the ATM experts were exposed to emerging issues that must be discussed during risk analysis and how they are related. Moreover, when shifting from before changes to after changes, the ATM experts were exposed to threat diagrams to identify changes to risks and explicitly model such changes. We collected feedback and comments of ATM experts by questionnaires and brainstorming sessions during the workshop. Figure 42 and Figure 43 show the median score for each evaluation group or criterion, respectively.

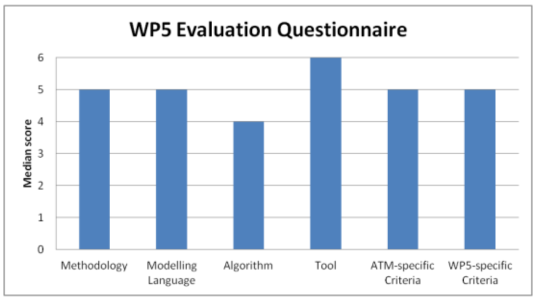


Figure 42 Median score grouped for each criterion category

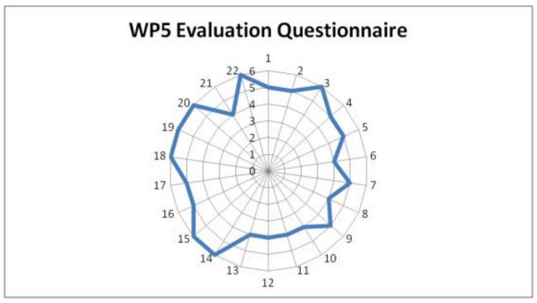


Figure 43 Median score for each evaluation criterion

In order to consolidate the validation of the CORAS risk assessment language and methodology, we distributed the ATM change requirements, the description of the CORAS methodology and relevant evaluation questionnaires (i.e. Safety Culture, Artefact Evaluation and Evolutionary Risk Analysis) to ATM experts. Figure 44 and Figure 45 show the median score for each evaluation group or criterion, respectively.

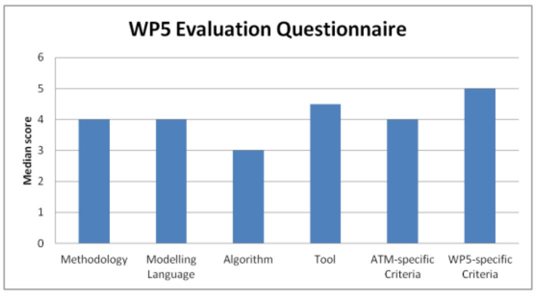


Figure 44 Median score grouped for each criterion category

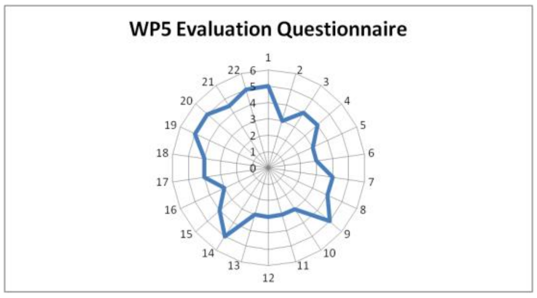
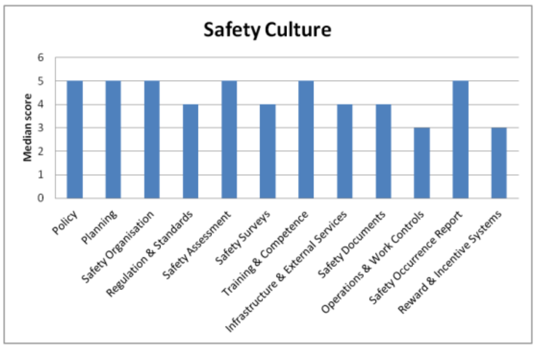


Figure 45 Median score for each evaluation criterion

Figure 46 and Figure 47 show the safety culture and the evolutionary risk analysis profiles, respectively.



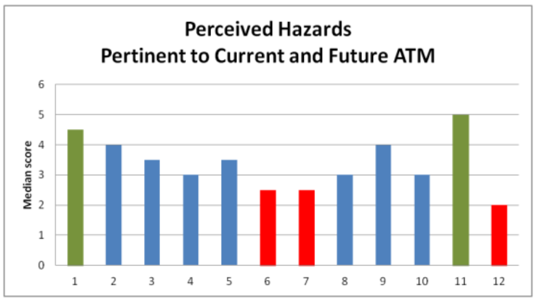


Figure 46 Safety Culture profile by median score

Figure 47 Evolutionary Risk Analysis

This allowed us to generalise the validation results with respect to social and organisational factors like safety culture and risk perception. The comparison of different perspectives (i.e. Safety Culture, Artefact Evaluation and Evolutionary Risk Analysis) highlights some analytical aspects of the CORAS methodology with respect to evolving risks. In particular, Figure 47 highlights those evolutionary risk analysis criteria that ATM experts agreed (i.e. 1 and 11) or disagreed (i.e. 6, 7 and 12) mostly:

> *[1] This Area of Changes increases the likelihood of well-understood current hazards that will exist in the future.*
>
> *[6] This Area of Changes renders the projected safety systems more brittle to off-nominal conditions.*
>
> *[7] This Area of Changes decreases safety levels during non-normal or emergency operations within the projected Future.*
>
> *[11] This Area of Changes creates new conditions that are currently not part of the design assumptions for the Future systems and procedures.*
>
> *[12] This Area of Changes results in decreased skill levels and judgment among operators of Future systems.*

In summary, the CORAS language and methodology supports the analysis of how risk changes due to requirements changes.

2.5.1.2 Risk Modelling Tool

VALIDATION SCENARIOS and EXERCISES

The CORAS Risk Modelling Tool was validated in a dedicated workshop (Rome, September 2011) with ATM experts. The WP5 artefacts validated are the CORAS method for analyzing changing and evolving risks, the CORAS language to support the modelling and assessment of changing and evolving risks, and the CORAS tool that supports the method and risk modelling. The validation activities focused on risk identification and risk estimation. The ATM experts were given the target of analysis, the security properties, the risk evaluation criteria and the change requirements.

The selected focus ensured immediate hands-on active use of core parts of all of the three WP5 artefacts ATM experts were asked to work using the artefacts and conducting the risk assessment on their own. They were able immediately to start building threat diagrams using the tool during structured brainstorming. They had a first iteration where they did risk identification and risk modelling before changes, as well as risk estimation. In the second iteration, they did risk identification and risk modelling after the changes. ATM experts by the hands-on experience got a good understanding of the CORAS method and how the method deals with change. The CORAS tool was actively used all the time, since it was used for building all the threat diagrams and annotating the diagrams with the likelihood and consequence estimates. The validation exercises cover the most important features of the CORAS tool. Figure 48 shows a sample screenshot of the CORAS tool. It shows an example of use of tool to do the risk modelling using the language in conducting the risk identification and risk estimation of the method. In particular, it shows an example of use of the WP5 risk assessment artefacts in identification and estimation of changes to risks.

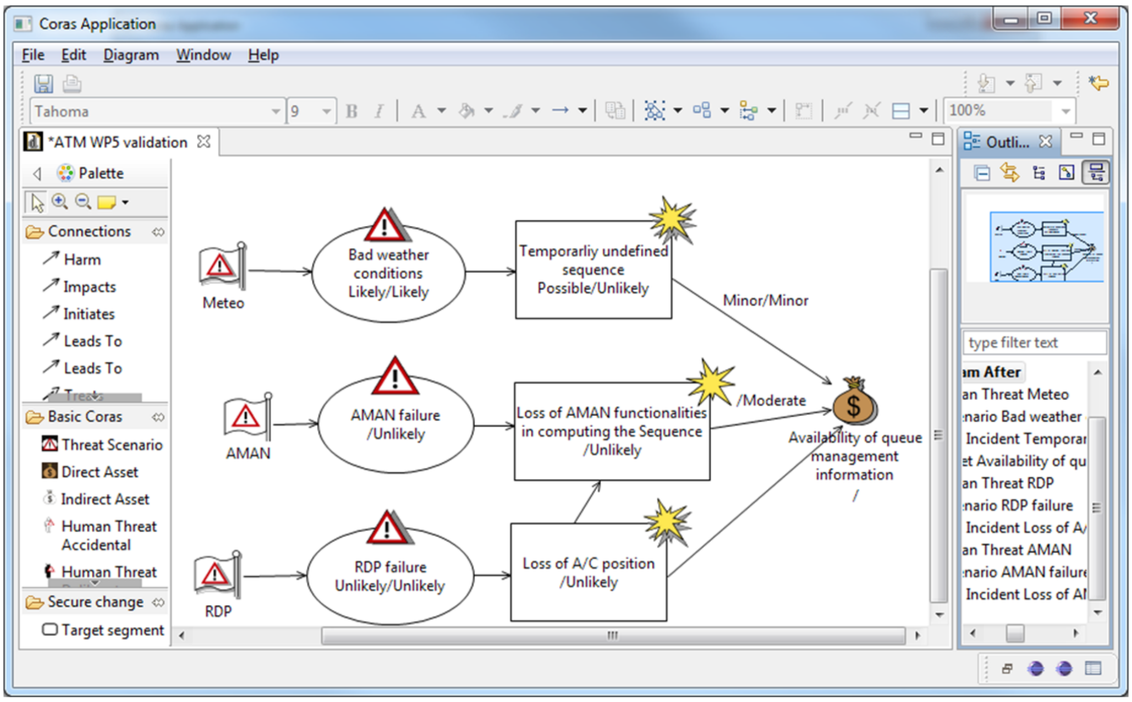


Figure 48 Sample screenshot of the CORAS tool

When conducting the hands-on session, the roles of the participants should be clearly defined. In a CORAS risk assessment, we have on the one hand the analysis team, and on the other hand the target team. The analysis team is commonly of two persons, one in the role of the analysis leader and one in the role of the analysis secretary. The analysis leader is responsible for leading the discussions and directing the participants during the risk identification and risk estimation. The analysis secretary is responsible for documenting the results by doing the on-the-fly risk modelling using the CORAS tool. The target team is commonly representing the customer of the analysis and typically consists of 4-6 persons. One of the roles is the decision maker (e.g. CEO, head of department, or the like) who is responsible for the approval of the focus and scope of the analysis (e.g. what the assets to be addressed are, what the risk evaluation criteria are). The other roles are typically users, domain experts, consultants, developers, security officers, etc. These are people with different insights into and experience with the target of analysis, and contribute to the identification of threats, vulnerabilities, unwanted incidents, and so on. We identified specific roles for the ATM experts. In order to evaluate the CORAS risk assessment, we divided participants into four teams, on the one hand the analysis team and on the other hand the target team. Each team were assigned roles to simulate an actual risk assessment setting: Analysis leader, Analysis secretary and Domain experts. The risk assessment was conducted in two iterations:

1. Identify, estimate and document risks before changes.
2. Identify, estimate and document risks after changes.

VALIDATION CRITERIA

There are not specific validation criteria for the CORAS Risk Modelling Tool. The validation criteria are drawn from the ones identified for the CORAS Risk Modelling Language and methodology – similar validation criteria apply to the Tool supporting the Risk Assessment Language and Methodology.

VALIDATION RESULTS

The overall evaluation of the CORAS Risk Modelling Tool was positive taking into account that risk assessment usually is conducted by people who are trained in using the artefacts. Figure 49 and Figure 50 show the median score for each evaluation group or criterion, respectively.

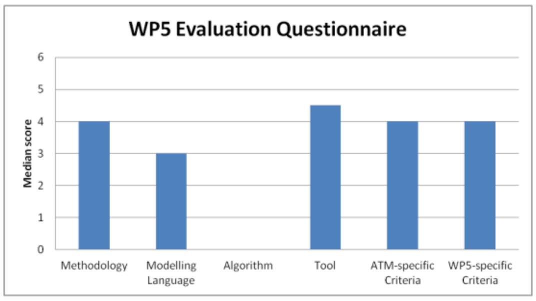


Figure 49 Median score grouped for each criterion category

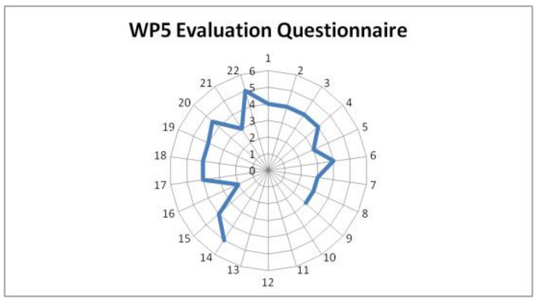


Figure 50 Median score for each evaluation criterion

ATM experts found easy to immediately start the risk identification by using the tool. The CORAS tool can be used with little prior introduction. However, conducting the assigned tasks requires a more thorough understanding of the CORAS method, the underlying concepts and the language. Like many other modelling approaches supported by a graphical notation, CORAS models easily become complex and messed up when dealing with changes. Conducting the assigned risk analysis tasks correctly with little training would be easier with further functionality for support in the tool, e.g. pop-ups with modelling tips when moving the mouse pointer over constructs, guidance in the tool on the pragmatics of the language with respect to the CORAS method, and automated support for systematically generalizing the models to changing risks. Some changes (e.g. flagging of changes, automated support for maintaining consistency when changes are introduced, and systematic tracing of changes over the diagrams) to the CORAS tool have been identified in order to address the evaluation feedback.

2.5.2 Validation Remarks

The evaluation activities for the WP5 artefacts stress the following evaluation remarks:

1. The CORAS framework resulted quite mature and able to deal with changes affecting security properties.
2. Different experts might benefit from the support provided by the CORAS methodology and approach.
3. Supporting different views (before and after changes) allows experts to focus during risk analysis exercises and to scrutinise how changes affect critical security properties.

3 HOMES CASE STUDY

3.1 VALIDATION ORGANISATION AND CONDUCT

This section describes the validation activities and results based on the HOMES case study. The validation organisation has been tailored to capture specific validation objectives with respect to the validate artefacts and HOMES domain features. The overall validation organisation, activities and objectives build over the previous WP1 deliverables [1][2][3], which have defined the scope and feasibility of SecureChange artefacts.

3.1.1 HIGH–LEVEL OBJECTIVES

The validation objectives for the HOMES case study have been focused on the effective usage of the artefacts (including their applicability and degree human effort involved), as well as specific industrial criteria such as perceived value, performance, or usability. Table 3 summarises how these objectives are mapped to the validated artefacts.

Table 3 High-Level Objectives

	WP2 artefacts		WP 6 artefacts	WP7 artefact	
	SeAAS	**Change Patterns**	**VeriFast**	**SxC**	**TTS**
		Effective Usage			
Applicability	X	X	X	X	X
Human Effort	X	X	X	X	X
		Specific industrial criteria			
Value appreciation		X			
Flexibility			X		
Effectiveness			X	X	
Usability			X		X
Performance			X	X	
Automation				X	
Completeness					X

These high-level objectives are decomposed into measurable indicators, as shown later in this section. The main validation activities fall into three major categories: **Methodology** Evaluation (modelling), **Walkthrough** and **Tool Live Demo** with HOMES Experts. Figure 51 shows how these categories are distributed across artefacts.

Methodology evaluation consisted of modelling exercises focusing on specific changes and security requirements in order to refine and consolidate the underlying modelling languages and their methodologies, respectively. Walkthrough activities involved step-by-step evaluation of the SecureChange methodologies with HOMES experts. This allowed to assess the proposed methodologies with domain experts and to identify alternative usages (with respect to current practices within the HOMES domain).

Finally, tool live demo activities and exercises allowed the validation (in terms of usability and acceptance by HOMES experts) of the tools supporting the SecureChange methodologies. Figure 51 shows the subsequent activities and their focus on the SecureChange artefacts forming the HOMES validation.

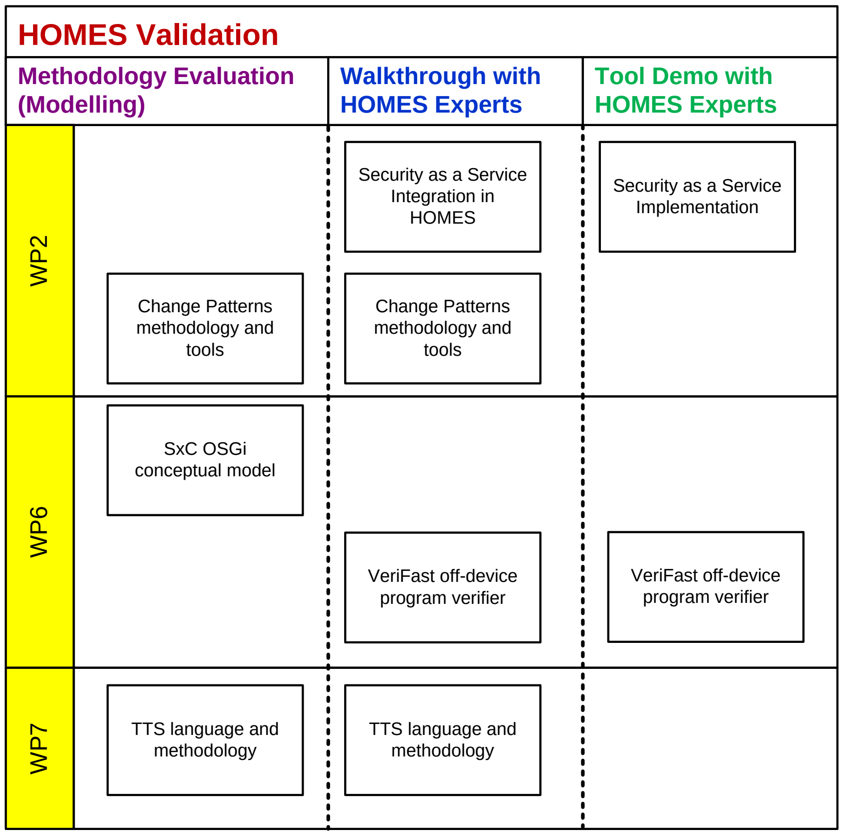

HOMES Validation
Methodology Evaluation (Modelling)
Walkthrough with HOMES Experts
Tool Demo with HOMES Experts
WP2
WP6
WP7
Security as a Service Integration in HOMES
Security as a Service Implementation
Change Patterns methodology and tools
Change Patterns methodology and tools
SxC OSGi conceptual model
VeriFast off-device program verifier
VeriFast off-device program verifier
TTS language and methodology
TTS language and methodology

Figure 51 HOMES validation

The remainder of this section will describe the validation outcomes for each WP artefact.

3.2 WP2 Architecture and Design Process

3.2.1 WP2 ARTEFACTS

The validation activities for WP2 focused on two artefacts:

- A complete SeAAS (Security as a Service) deployment for the HOMES case study supporting the addition of new security functionalities.
- Change Patterns, a tool to assess the impact of trust changes into the system.

3.2.1.1 Security-As-A-Service (SeAAS)

This section reports on the validation of the Security-As-A-Service (SeAAS) artefact from WP2 on the HOMES scenario. Security as a service is an architectural blueprint that transposes the model of Software as a Service to the security domain. This results in a robust architecture with enough flexibility to cope with a broad variety of changes, thus supporting long-lived, evolving systems.

VALIDATION SCENARIOS and EXERCISES

For the Security as a Service artefact in the HOMES scenario, the validation covered the change requirement of Bundle Lifecycle Operations, in combination with the security property of Security Expandability. The change scenario consisted on a degradation of trust between a network operator and a third party service provider, which forced the deployment of new security functionality in the form of a non-repudiation protocol.

The validation exercise for this artefact consisted of the deployment of SeAAS infrastructure for the HOMES case study, along with a new security service implementing a non-repudiation protocol. A sample service consisting on a news feed service was also added to the platform and integrated with this non-repudiation functionality. This deployment was performed by UIB, with the assistance of TID. A detailed description of the HOMES-SeAAS architecture can be found in [1].

On this HOMES-SeAAS architecture, TID domain experts conducted a series of experiments according to the validation criteria previously defined. The experiments included observing the performance of the non-repudiation protocol for the feed service through network and system traces, adapting different OSGi services to use the SeAAS architecture and non-repudiation functionality, and managing the SeAAS configuration both using a manual approach and a model-driven configuration methodology also developed for the SeAAS artefact.

VALIDATION CRITERIA

Two validation criteria have been defined for the SeAAS artefact, related to its effective usage: applicability, and human effort. These have been further divided into measurable indicators, as shown below.

Applicability

- SeAAS architecture can be deployed on the HOMES case study.
- Stakeholders can configure the SeAAS infrastructure.
- Stakeholders can adapt and extend the SeAAS infrastructure and the model-driven configuration approach.
- Service providers can use SeAAS in HOMES services.

Human Effort

- Model-based approach configuration effort.
- Extension effort.

A detailed discussion of the evaluation for each sub-criterion is provided below.

SeAAS architecture can be deployed on the HOMES case study

The validation exercise proved that involved researchers can apply the SeAAS approach to an OSGi-based home gateway like the one presented in the HOMES scenario. UIB and TID were able to deploy a SeAAS architecture on the HOMES platform, along with a non-repudiation security service and basic security services like cryptography, time stamp, or logging, and integrate them with a sample service consisting of a news feed.

It should be noted that the HOMES entities (such as home gateway, network operator, and service provider) in the validation exercise were implemented as OSGi platforms running on Linux virtual machines, for convenience. This allowed replicating the validation environment and testing it at different locations with ease. Nevertheless, this same deployment could be translated to actual home gateway hardware, since the implemented OSGi bundles can fit even in the limited footprint of these devices.

Stakeholders can configure the SeAAS infrastructure

Once the SeAAS architecture was deployed on HOMES, the platform was handed over to TID domain experts to experiment with. One of the tests they performed consisted on managing the security configuration for the SeAAS infrastructure in combination with the sample news feed service, confirming that such configuration was feasible for the stakeholder.

Two configuration approaches were supported in this exercise. The first one was a declarative security approach, where a series of policy files associated with each bundle indicated the security services associated with that bundle, and their configuration. On the second, a model-driven security approach, a GUI was provided for the system administrators to easily choose from the available security services and configurations for a given bundle – after that, the model driven configuration tool would automatically generate the corresponding policy files. The domain experts were able to use both approaches without assistance (other than the architecture documentation), though they found the model-driven tool to be easier to use.

Service providers can use SeAAS in HOMES services

The first sample service to run on the HOMES SeAAS architecture was a news feed service implemented by TID with assistance from UIB. This served to prove that it is possible to develop OSGi services for that platform that benefit from the SeAAS approach, with some assistance from the researchers. A further experiment consisted on taking an OSGi developer with no prior experience with SeAAS (but access to the HOMES SeAAS documentation), and have her create a simple service which could be integrated with the SeAAS non-repudiation protocol. This was performed successfully, demonstrating that a service provider acting in complete independence could benefit from SeAAS functionalities.

Stakeholders can adapt and extend the SeAAS infrastructure and the model-driven configuration approach

Due to time constraints, it was not possible for TID to run experiments on the extension of the SeAAS infrastructure. That said, after studying the SeAAS architecture and source code, TID experts have concluded that extending this architecture (e.g. to implement new security services, or new options for existing security services) should not be particularly challenging, provided they have access to a sample security service over SeAAS, like the non-repudiation protocol, to use as reference.

Any future extension to the SeAAS infrastructure should also be translated to the model driven security tool. Extending the MDS tool relatively simple, since it just requires adjusting the system meta-model with the new changes, and creating a template for the policy file required after the change. Though this has not been possible to try in practice either, we believe it to be feasible without much effort.

Model-based approach configuration effort

On the validation exercise, we found that configuring the infrastructure using the model-based security tool was easy and quick. Using this tool, changing the security services associated with a bundle or the configuration parameters of these security services only took a few clicks. It was an intuitive process thanks to the friendly graphical user interface. Overall, the configuration process using this tool was considerably faster than under the declarative security approach (i.e. editing policy files by hand), particularly for less experienced users. That said, experienced system administrators can typically make more efficient use of text-based editing tools, thus reducing the difference between the declarative and model-based approach – though the model-based tool should still require less effort, even in that scenario.

One point that couldn't be explored in depth in our validation exercises is the behaviour of both approaches when configuring more elaborate scenarios. The setup we used represented a simplified home gateway with just a couple of services, whereas actual deployments tend to have a much higher number of services, with the corresponding increase in system configuration complexity. Overall, the results observed in this simple scenario (i.e. model-based configuration performs better) should remain valid regardless of the amount of services in the system, but a more realistic scenario might reveal interactions or bugs that are not observable this way.

Extension effort

As has been explained above, it has not been possible to experiment with infrastructure extension during the validation exercises, so we have no direct data on the amount of effort required by the extension process. That said, both the SeAAS architecture and the model-driven security solution have been designed with extensibility in mind, and have a series of features that should reduce extension effort compared with other approaches. As an example, SeAAS allows the introduction or modification of security services to cope with new requirements without needing to propagate the changes to every endpoint in the system, as happens in endpoint security or declarative security. Likewise, from a configuration standpoint, the model-driven security approach allows us to introduce changes just by adjusting meta models and policy templates, since the tool takes care of the generation and distribution of any new policy files.

VALIDATION RESULTS

An overview of the validation results is shown in the following table (Table 4). The results can be summarized as follows. The applicability criteria for the HOMES scenario has been demonstrated to be met for the most part: SeAAS can be deployed on an OSGi home gateway, and it is possible for TID to configure this infrastructure, and for service providers to use its security functionalities on their services – without external assistance. The one point regarding applicability that hasn't been proved by experience (due to time constraints) is TID's ability to adapt and extend SeAAS, though analysis strongly suggests that this should in fact be feasible for the stakeholder.

As for the human effort criteria, the validation exercises have shown that the model-based configuration approach allows the configuration of security parameters in the HOMES scenario with less effort than alternate approaches like declarative security. The extension effort is also believed to be reduced by the use of a SeAAS architecture and model-based configuration.

Table 4 Security-As-A-Service (SeAAS) validation results

Applicability	Can be applied by stakeholder with little or no assistance
A SeAAS architecture can be deployed on the HOMES case study	Yes
Stakeholders can configure the SeAAS infrastructure	Yes
Service providers can use SeAAS in HOMES services	Yes, in complete independence
Stakeholders can adapt and extend the SeAAS infrastructure and the model-driven configuration approach	Feasible (not tested)
Human effort	**Compares favourably to declarative approach**
Model-based approach configuration effort	More efficient than declarative approach
Extension effort	Should be more efficient (untested)

Industrial considerations

Although the validation exercises for the HOMES-SeAAS infrastructure have focused on the use of this methodology in isolation, practical development tends to involve a wide variety of processes and tools, and it's important to know how these can be integrated. With relation to this, there has been work on the integration of different artefacts for the HOMES scenario, resulting in a demonstration case study which showcases an Integrated Process and an integrated MoVE tool. In this case study, the Security-as-a-Service architecture and model-driven security tool have been used successfully alongside the CORAS risk modelling methodology, and UML-based system modelling. Thus, it is demonstrated that SeAAS can work in combination with other methodologies and tools in a realistic development scenario.

3.2.1.2 Change Patterns

The WP2 artefact Change Patterns consists on a methodology to assess and minimized the impact of changes in the architecture of a system, as well as a tool to assist in its use. In the context of the HOMES scenario, Change Patterns has been applied to deal with changes in the trust relationships between actors in the system.

VALIDATION SCENARIOS and EXERCISES

For the Change Patterns artefact in the HOMES scenario, the validation covered the change requirement of Bundle Lifecycle operations, in combination with the security property of resilience to trust changes. In this artefact, a change scenario referring to the degradation of trust between a network operator and a 3rd party service provider is defined. Based on this change scenario, a catalogue of change patterns is presented to help developers handle these changes in the HOMES scenario.

The validation exercise for this artefact consisted in a workshop where HOMES domain experts from the stakeholder (TID) applied the Change Patterns methodology to a series of changes in the HOMES architecture, assisted by researchers from KUL. After this workshop, which was carried out via videoconference and recorded for posterior analysis, the domain experts filled a questionnaire about their experience with the methodology, and participated in a series of interviews with the researchers.

Appendix G reports a detailed description of the workshop.

VALIDATION CRITERIA

Two categories of validation criteria have been defined for the Change Patterns artefact: Effective usage (which includes applicability and human effort), and specific industrial criteria (which in this case include the appreciation of value for the stakeholder). These criteria have been further decomposed into measurable indicators, as shown below.

Applicability:

- Change scenarios apply to HOMES.
- Suggested solutions can be applied in HOMES.
- The methodology can be applied in HOMES.

Human Effort:

- Learning effort.
- Modelling effort.
- Preparation effort.
- Change Implementation effort.
- Overall effort.

Value appreciation:

- Anticipates changes.
- Saves effort.
- Guides design decisions.

A detailed discussion of the evaluation for each sub-criterion is provided below.

Applicability

The applicability criterion determines whether the change patterns methodology can be successfully applied to the HOMES scenario. For the purposes of evaluating this criterion, we have decomposed the artefact into three parts: the process of the methodology, the defined change stories, and the set of suggested solutions associated with each pattern.

Change scenarios apply to HOMES.

The catalogue of patterns defined for the artefact focuses on change scenarios related with evolution of trust relationships. During the workshop, several examples of how these generic scenarios can apply to the HOMES architecture were shown – evolving trust being one of the primary security concerns for HOMES, to begin with. The trust evolution patterns prove that change scenarios that are relevant for HOMES security exist. However, Change Patterns is a generic methodology that can be applied to other kinds of changes, as long they happen commonly in software architectures and have generic solutions. This means that more patterns could be defined to apply the methodology in other environments that aren't as concerned with trust relationship changes. It also suggests that the use of Change Patterns in HOMES could be expanded to cover other types of changes.

Suggested solutions can be applied in HOMES

In order for Change Patterns to be useful for a project, not only do the change scenarios need to be compatible with the architecture of that project, but the solutions suggested by each pattern need to be applicable as well. In the workshop, we found that for all the patterns that were applied, one or more of the suggested solutions were viable in the context of HOMES. Due to time limitations, not every pattern in the catalogue was used during the workshop, but our analysis indicates that each pattern in the catalogue has at least one solution that can work for HOMES, either directly or with minor modifications. Note that the Change Patterns methodology is not guaranteed to provide the best solution for any given change, since that is highly context-dependent and falls outside the scope of the artefact. Rather, the patterns give a set of common, generic solutions that are known to work in many environments. These solutions should be used as a starting point to guide the design of the architecture, and may later be refined or replaced if optimization is required for the system.

The methodology can be applied in HOMES

In the validation workshop, it was verified that the Change Patterns methodology can be applied on a home gateway architecture such as the one presented in the HOMES scenario. Leaving aside the validity of change stories and suggested solutions (discussed in the previous sections), the process followed by the methodology (which can be summarized as applying patterns to prepare an architecture for future changes during design, and use pre-defined solutions when implementing these change) can easily be translated to any software project. One important factor to take into account regarding the applicability of Change Patterns in an industrial context is that this methodology is intended to be used as early as the design phases of a project. This means that the approach is definitely suitable for new projects. However, it remains to be seen whether the approach also leads to benefits for projects that are already in development or maintenance.

<u>Human Effort</u>

The level of human effort required by this artefact has been evaluated considering five separate categories: the learning effort, the effort associated with the generation of models required by the methodology, the effort required at the preparation phase, and the effort required for implementing a change. Finally, an overall estimation of the effort to implement the methodology is provided and evaluated.

Learning effort

In the validation exercise, it took TID experts 6 hours to start applying the methodology with assistance. This includes learning of i*, SI*, change patterns methodology and catalogue, and top-cased modelling tool. After this time, our experts were able to successfully complete the exercises presented in the validation workshop, though they weren't fully confident of the results, and had to spend additional time checking the documentation. We estimate 10-12 hours, including some practical exercises, before a developer is able to apply the methodology independently on a real project. If we were using an extended pattern catalogue covering more than the 8 patterns defined for SecureChange, the learning effort would increase accordingly. We estimate that between 2 and 3 hours are required to become familiar with a set of 8 patterns, though this would likely be lower for a developer that has already studied many patterns. The degree of similarity between patterns in a set also affects learning effort, to a lesser degree.

Modelling effort

One of the requirements of the Change Patterns methodology is to have an alternate representation of the architecture using the SI* modelling language, to show the trust relationships between actors. This involves an additional modelling effort, which on our experience approaches a couple of hours for the initial model, and a few minutes for changes implemented by following a pattern. An issue that we did not have enough time to observe properly is how the SI* model is affected by changes that are unrelated to the defined patterns. This is highly dependent on context, but whenever the system is subject to a change that affects requirements or architectural assumptions (whether or not that change matches an existing pattern), the SI* model will need to be updated. Furthermore, it is hard to predict the difficulty of such a model update when no pattern is applied. We believe that the overall effort for model maintenance would remain relatively low nevertheless, but some direct experience applying the methodology over a real project would likely be needed for an accurate estimation.

Preparation effort

During the preparation phase of Change Patterns (which should coincide with the design phase of the project), the change scenarios most likely to occur in the architecture are selected, and solutions are chosen for them following the suggestions in their respective patterns. Applying each of these patterns in this step usually involves introducing minor changes in the model and implementing them in the architecture. This is not intended to take a significant level of effort – at this point the changes consist mostly on preparing interfaces, adding stubs, and ensuring that certain security services can be met in the future. Though the preparation cost for each individual pattern should remain low, the overall preparation effort for Change Patterns is proportional to the number of applied patterns. This means that there is a risk of wasting resources (and needlessly complicating the architecture) by applying an excessive number of patterns. On the other hand, we want to make sure that the most likely changes for our architecture are adequately covered by a pattern. Thus, developers need to strike a balance that maximizes pattern coverage while minimizing the risk of unused patterns. Given that the cost to prepare for a pattern is relatively low, whereas the potential benefit if that change scenario comes up is considerable, we'd recommend to err on the generous side (i.e. prepare for more patterns) in case of doubt.

Change Implementation effort

The advantages of the Change Patterns methodology become evident at the change implementation phase, since whenever a change scenario comes up that matches a pattern added to the system during preparation, the effort to implement the change will be noticeably reduced. At this point, the design of the solution should be mostly accomplished, and a developer should only need to implement the missing functionality required by the change, and plug it into an architecture that is already prepared for it. The precise amount of effort saved will vary depending on the architecture, the specific pattern, and the nature of the change.

Overall effort

When evaluating the effort involved in this methodology, the main question is whether the time savings achieved while implementing changes compensate for the added costs of learning and modelling.

As seen on the previous sections, the learning effort isn't too high, and can be minimized if we expect developers to use the Change Patterns methodology over several projects. Likewise, the additional costs associated with the SI* models are fairly low. Thus, the two main effort components that should be taken into consideration are preparation effort, and change implementation effort savings. The application of this methodology will be advantageous from an effort standpoint whenever:

- the savings in change implementation for a given pattern exceed the costs of change preparation (i.e. patterns actually save effort and do not just move it to an earlier stage),
- the rate of pattern application (i.e. the frequency of system changes, multiplied by the likelihood of a change matching an existing pattern) is high enough for these savings to exceed the fixed costs of the methodology (learning and modelling).

It is hard to provide concrete values for the first point, the effort balance for each pattern. However, we do know that implementing a change that has been prepared in advance never costs more than doing so without preparation, and that introducing drastic changes in an unprepared architecture can be costly, as well as prone to errors. So we can estimate that patterns typically result in significant effort savings.

The second point, pattern application rate, is easier to characterize. Projects where frequent changes are expected are best for this methodology, and a careful effort in selecting the set of prepared patterns will greatly improve performance.

<u>Specific Industrial criteria</u>

The specific industrial criteria that was defined for the Change Patterns artefact was the value that the methodology can provide to the industrial stakeholder. We identified two properties of the methodology that were of value for TID, in the context of the HOMES scenario: the ability to prepare for changes in advance in order to minimize errors and improve system stability and the reduction of effort in systems with frequent changes. In addition, a secondary value-adding property was found: the fact that the methodology can help in making design decisions.

Anticipates changes

The ability to anticipate changes and prepare for them in advance with minimal effort is one of the main selling points of Change Patterns. Unexpected architectural changes when a project is late in development or already in production can be costly and error prone. By contrast, making these same changes after preparing for them early in design (such as when applying a pattern) is much safer and easier, and better preserves architecture stability. Each time the methodology succeeds in anticipating a change, it provides great value to the stakeholder. The probability to anticipate a change for a given project depends on the number and relevance of patterns that are applied during design, and on the rate of changes for that project. In general, applying more patterns in the preparation phase increases the amount of changes that will benefit from the methodology, as long as their change scenarios are relevant to the architecture. However, this tends to yield diminishing returns, since once the main change scenarios have been covered, applying new patterns will only address scenarios that are either less likely or have less impact on the architecture. The overall rate of changes in an architecture is also significant towards change anticipation, since the more changes there are, the more likely that any given pattern will come up. In the case of the HOMES scenario, we expect that the home gateway architecture will be subject to a relatively high rate of changes, and that trust evolution will be present in many of these changes, making the chances of successfully applying the patterns in the current catalogue quite high. That said, we believe there is still room for growth. There is potential for pattern categories other than trust evolution that could be useful in HOMES. For this reason, we consider that extending the existing pattern catalogue shows promise as a future line of work.

Saves effort

As we have seen in the discussion of the Human Effort criteria, the use of change patterns under the right conditions results in an overall reduction of development effort once changes are applied. This is a very valuable property for the stakeholder.

Guides design decisions

One secondary aspect of Change Patterns that can nevertheless prove valuable to developers is its ability to guide design decisions. This is manifested in two ways: by helping identify common problems, and by providing generic solutions. When a developer looks at a catalogue of patterns and attempts to apply it to the architecture of a system, he’s likely to identify potential problems or future system evolutions that he was previously unaware of. This is not to say that the change scenarios in a pattern catalogue cannot be identified by other means, such as analysing an architecture, but the catalogue approach is usually the best way to ensure that a given set of changes is properly covered by the architecture. Related to the previous point is the ability of a pattern catalogue to suggest generic solutions to address a change scenario. Though one should not expect the list of solutions provided by a pattern to be exhaustive, or to include the optimal solution for a certain context, this list does provide insight on what approaches are commonly useful in a type of scenarios. Even when they don’t include the best approach for a given situation, the suggested solutions can be seen as a useful reference – either a starting point to be later optimized on, or a sub-optimal but workable solution that is still good enough for the less likely or important change scenarios. At any rate, having access to these pre-defined solutions is a useful tool that can reduce development time.

When applying Change Patterns on HOMES, we found the pattern catalogue to be helpful in this regard, by providing one or more suggested solutions that were viable for the change scenarios that came up. By following the patterns, the designers identified some future problems and their corresponding solutions with relatively low effort.

VALIDATION RESULTS

An overview of the validation results is shown in the following table (Table 5). The results can be summarized as follows. The applicability criteria for the HOMES scenario are fulfilled, since the methodology can be fully applied by a researcher on this case study. Moreover, though this was not considered for the original specification of evaluation criteria, we believe that the stakeholder would be able to apply the methodology for HOMES in an industrial context with little or no assistance.

Table 5 Change Patterns validation results

Criteria	Evaluation
Applicability	**Can be applied by researcher on case study**
Change scenarios apply to HOMES	Yes
Suggested solutions can be applied in HOMES	Yes
The methodology can be applied in HOMES	Yes
Human Effort	**Can save effort under certain conditions**
Learning effort	10-12 hour
Modelling effort	2 hours initially, plus ~10 minutes per change
Preparation effort	Low, increases with # of patterns
Change Implementation effort	Lower than manual approach
Overall Effort	Depends on effort saved per change, and rate of changes
Value to stakeholder	**Valuable if frequent changes and catalogue is appropriate.**
Anticipates changes	Very valuable. Depends on # of patterns, and rate of changes.
Saves effort	Very valuable. Depends on effort per change, and rate of changes
Guides design decisions	Less valuable. Helps developers.

Regarding the criteria of human effort, we have determined that the methodology can save effort over the course of a project, provided that the architecture is subject to frequent changes, and that these changes would be costly under the manual approach. Effort overheads such as learning and modelling are relatively low, and are concentrated at the beginning of a project.

For the specific case of the HOMES scenario, the projected rate of changes in the HOME gateway architecture makes it suitable for the methodology, thus allowing for such effort savings. As for the specific industrial criteria, which has been defined as the value provided to the stakeholder, our evaluation is positive. Two major sources of value have been identified in the ability to anticipate changes and prepare for them in advance, which prevents errors and increases architecture stability, and the effort savings mentioned above. Both properties would be of application in the HOMES scenario, making the methodology valuable for TID in this case. In addition, we have identified a secondary valuable property of the methodology in its ability to guide design decisions, easing the work of developers. Overall, we consider that the Change Patterns artefact is a good match for the needs of a home gateway architecture, and that adapting it to other industrial scenarios should be feasible.

Industrial considerations

A major consideration for the use of Change Patterns is its place in the development lifecycle. By its nature, the methodology should be applied as early as possible during a project, since a significant phase takes place during design. This makes it ideal for adoption in new developments, but might challenge its application on existing architectures – though this has not been assessed so far. That said, there may be a case for using Change Patterns in an ongoing project that is expected to suffer frequent changes over a long period of time. Although this is clearly not as ideal an scenario as its use on new projects, and despite the fact that in this case even changes prepared and implemented with the help of Change Patterns are likely to be costly and risky, it could nevertheless be worthwhile to use the methodology. The benefit of planning in advance for architectural changes without the pressure of close deadlines cannot be underestimated. Though we haven't had the chance to explore this use of the methodology, we believe it shows promise, since it could greatly expand its scope of application.

3.3 WP6 Verification

3.3.1 WP6 ARTEFACTS

The validation activities for WP6 focused on two artefacts:

- VeriFast, a tool to validate some core security modules (programs written in C language).
- Security-by-Contract (SxC), a methodology to verify the *software contract* of OSGi bundles.

3.3.1.1 VeriFast

This section reports on the validation of the VeriFast software verification tool as provided by WP6. The VeriFast tool allows the verification of C and Java software, taking as input the source code along with annotations consisting in method contracts written in separation logic, inductive data type and fix-point definitions, lemma functions and proof steps. Theoretical background and technical details on VeriFast are presented in D6.2.

VALIDATION SCENARIOS and EXERCISES

As discussed in D1.2, the experimental focus of WP6 off device verification on HOMES is on the "secure extensibility" security property, in the context of the "core security module update" change requirement. The goal is to verify one of the HOMES software modules, ensuring that no illegal operations (such as dividing by zero or illegal memory access) are performed in the code, and (since the module is multi-threaded) that there are no data races.

The validation exercise for this artefact consisted in the application of the VeriFast tool to verify the C code of a HOMES module, the PEP (Policy Enforcement Point). From the PEP source code provided by TID, an engineer generated the annotations required for the verification process. In parallel, the VeriFast team worked on extending C language support in the tool in order to make it fully compatible with the PEP code. Finally, HOMES domain experts examined and evaluated the process and the achieved results.

PEP implements a Policy Enforcement Point for home gateways. The program, consisting of approximately 1700 lines of C code, facilitates the application of security policies in Network Admission Control scenarios. That is, for an authenticated network device, PEP will receive an access policy from a Policy Decision Point. This policy is then put in place by configuring the gateway's network interfaces accordingly. An extended description of the case study is given in D1.1. This case study is eminently interesting and challenging due to PEP's close interaction with Linux networking components. It is the first case study in which VeriFast is employed on low-level network management software.

Note on validation time line

The PEP source code is released to WP6 in November 2010. Subsequently, an initial assessment of the feasibility of fully verifying the program is carried out in December 2010. It is concluded that conducting the verification will require VeriFast's support for the C language to be extended, imposing substantial implementation effort on the VeriFast team. Work on extending VeriFast with the objective to verify PEP starts in March 2011. In September and October 2011 an extended case study on verifying PEP is conducted. In the course of this case study WP6 produces a modified version of PEP so as to only use C language features that are supported by VeriFast. In addition, initial annotations, i.e., method contracts for the PEP implementation, are produced. WP6 concludes that fully verifying PEP is feasible. Yet, it is not clear whether this work can be finished before the submission of the final project deliverables. In consequence, this report is based on a validation of ongoing work on the case study.

VALIDATION CRITERIA

The validation criteria considered in this report are given in D1.2. In summary, we investigate whether the VeriFast tool can be applied by the owner of the case study and whether, and to what extent, interaction with researchers and the developers of VeriFast is required. Of particular interest for TID is the usability of VeriFast for developers and testers with few or no experience in the field of formal methods.

We also determine the required human effort, judging whether the application of VeriFast imposes additional costs or saves effort as compared to manual verification or pure testing.

Focusing on further technical aspects of VeriFast, we evaluate the flexibility of the tool, judging its applicability to further security modules without major change. We also validate the tools effectiveness, i.e., that a number of true property violations is reported while the rate of false-positive error reports is comparably low. Considering performance aspects, we require VeriFast to show similar run-times as a compiler to enable efficient integration into existing development processes.

The iterative decomposition of these evaluation criteria into measurable indicators is shown below.

Effective Usage

- Applicability:
    - Verification process can be performed on HOMES.
    - Verification results can be understood.
- Human Effort
    - Learning effort.
    - Verification effort.
    - Comparison to manual approach.

Specific Industrial Criteria

- Flexibility:
    - Tool can be applied to different software modules.
- Effectiveness:
    - False negative rate.
    - False positive rate.
- Usability:
    - Non-experts can check error reports on the tool.
- Performance:
    - Verification time.

Verification process can be performed on HOMES

At present, the independent use of VeriFast by the stakeholder is constrained by the still incomplete support of the C programming language in the verification tool. In particular, VeriFast lacks support for compound data structures (structs) that are not dynamically allocated. Furthermore, initialisers for compounds and arrays are also not supported. While it is certainly possible to change the PEP implementation (or any other security module) to avoid these programming constructs, this is highly inconvenient. WP6 confirms that most issues with respect to language support can theoretically handled by the verification algorithm in VeriFast. Hence, enabling the stakeholder to perform verification in complete independence of the tool developers is mainly an issue of additional tool development effort.

Given that, the verification can be currently done partially by the stakeholder. Eventually, as C language support in the tool improves, the stakeholder should be able to perform the verification with little or no assistance.

Verification results can be understood

VeriFast provides an interactive verification experience as verification times are short and errors can be diagnosed using a symbolic debugger. Thus, the results of applying VeriFast, i.e. bug reports, can be easily inspected and understood by the stakeholder, and specifically by developers with no direct knowledge of the tool algorithm.

One limitation when displaying verification results in the current version of the tool is that only the first error in the code is shown – in order to observe further errors, a developer has to first address or comment the code causing the current one. Having the verification tool show a full list of errors in a section of code would make the process of understanding and fixing problems with the code significantly faster and easier.

Learning effort

As part of the verification process, it is required to annotate the code to be verified. Writing these annotations is difficult and would require a developer or tester to be trained in the use of formal methods. Our evaluation and validation efforts show that an experienced developer may need a month to learn using VeriFast effectively.

Since learning the use of annotations involves such a considerable commitment, we believe that the only cost-effective solution is to delegate these verification tasks on specialized personnel. Thus, a number of verification engineers, (similar to testing engineers) would be in charge of annotating code to be verified across multiple projects.

Verification effort

As a drawback, the tool suffers in terms of usability from the amount and complexity of annotations that are to be put in place in order to perform verification. As reported by WP6, applying VeriFast imposes an average overhead of 0.5 lines of annotations per line of code. Moreover, writing these annotations is difficult and would require a developer or tester to be trained in the use of formal methods, as explained in the previous section.

Given the high cost involved in the verification of a section of code, this is not an operation that should be taken lightly. In fact, only highly sensitive software should ever be subject to verification of its full code. For most systems (including HOMES), developers should take advantage of the fact that VeriFast can check specific sections of code, by annotating and using the tool to verify only the most critical parts that can really benefit from the level of assurance provided by this process. Non-critical code can still be checked by other, less expensive means.

Comparison to manual approach

The use of formal software verification tools like VeriFast is intended to provide very strong guarantees for the correctness of an implementation artefact. That is, the software will behave correctly with respect to its specification (given in terms of source-code annotations in the case of VeriFast) for all possible input vectors. Other techniques, including testing and code inspection, are complementary to formal verification but do not provide the same level of assurance. Thus, they cannot be compared in terms of effort – VeriFast achieves results that the manual approach can't match.

What needs to be determined, then, is whether a system requires the high level of assurance provided by VeriFast. In the case of safety-critical systems, this is typically the case.

In a case like HOMES and the PEP module, a selective application of VeriFast may reveal critical errors that would otherwise emerge after deployment – so it can be expected to reduce the effective maintenance costs of a software project.

Tool can be applied to different software modules

VeriFast may readily be employed for the verification of other security modules, provided their code falls within the supported subset of the C programming language. Note that ongoing work to expand C language support for the tool contributes to mitigating this limitation. Eventually, it should be possible to use the tool on most C language software. In addition, verification of Java code is also supported by the tool, though this falls outside of the scope of the HOMES validation exercises.

Finally, there are two points that puts into perspective how limiting the restriction to a subset of C really is. On the one hand, outside of extremely sensitive systems (which the HOMES gateway architecture isn't), it is not necessary for VeriFast to cover 100% of the system code – rather, it is expected that only select portions of the code (such as those dealing with threading or memory allocation) should typically be verified. Apart from that, it is possible to sidestep the lack of compatibility for certain language features by rewriting the code of the verified system so that it provides equivalent functionality without accessing these features. This could be a very expensive solution if the whole system code had to be modified this way, but the fact that it is possible to perform verification on just a part of the code (as explained above) makes this manageable. In fact, this approach has been successfully applied to the HOMES PEP code during earlier stages of the validation exercises, while C language compatibility was still relatively reduced.

False negative rate

One of the crucial properties of the VeriFast tool is its low rate of false negatives – it provides a very good level of assurance that a verified module is free of the types of errors covered by the algorithm (including threading errors and memory problems).

It should be noted that there are a few factors that can increase the false negative rate. The best performance is achieved when 100% of the code is verified, and the corresponding annotations have been perfectly defined. However this will not always happen in practice. As has been discussed, the cost of annotation will lead to verification only covering critical portions of code for many systems. In this case, VeriFast offers no guarantees whatsoever that the unverified code is error-free though, if the verified code has been selected carefully, any unverified errors will have relatively low impact.

Quality of annotations can also affect the amount of false negatives, in that in that a section of code with insufficient or poorly defined annotations will not be properly verified, leading to a lower level of assurance in practice. Thus, verification performed by less experienced developers is less reliable. Because of this, special attention should be devoted to the learning process of the VeriFast tool. Also, the development of tools that assist in the generation of annotations becomes a highly interesting topic – and in fact there is ongoing research in Leuven following that direction.

False positive rate

Another important parameter to describe the quality of the verification is the rate of false positives. Though not directly affecting the level of assurance provided by the tool, the amount of false positives is significant in that an excess of such positives can artificially increase the effort required to fix errors after verification, by forcing developers to waste time on incorrectly diagnosed bugs. In our experience, VeriFast has a low rate of false positives. It is worth clarifying that the verification process performed by VeriFast does not necessarily point to sections of code that are actively causing problems in the software, but to dangerous code that has the potential to do so. This kind of bugs is not considered a false positive even when a particular bug might not be negatively affecting the behaviour of a program at a given time – they tend to have unpredictable interactions, and are likely to cause problems over time. Thus, identifying and fixing them is always desirable.

The fact that the validation exercise for HOMES is still in progress makes it difficult to provide specific examples for this evaluation criteria. Still, having applied VeriFast only to a subset of PEP so far, WP6 already reported a strong indication for a race condition in the code. Final conclusions on the effectiveness of VeriFast on this case study can only be drawn once the exercise is complete. As with false negatives, the rate of false positives can be affected by low quality annotation – the considerations regarding learning and tool usage, mentioned in the previous section, also apply here.

Non-experts can check error reports on the tool

Unlike the annotation generation step, result inspection in VeriFast does not require any particular knowledge of the algorithm. In VeriFast, errors can be diagnosed using a symbolic debugger. Thus, the bug reports provided by the tool can be easily inspected and understood by the stakeholder. In practice, this will mean that in a typical project, only a small number of developers will need to be aware of the nuances of VeriFast, in order to properly annotate the code – once that is done, anyone can access and use the information provided by the verifier.

Verification time

Considering performance aspects, we require VeriFast to show similar run-times as a compiler to enable efficient integration into existing development processes. VeriFast performs efficiently on the example. In fact, verification time is consistently short and on par with compile time. This is an important point towards integrating VeriFast in existing validation processes: the tool may be used interactively by the developers.

VALIDATION RESULTS

An overview of the validation results is shown in the following table (Table 6). The results can be summarized as follows. The applicability criteria for the HOMES scenario are mostly fulfilled, since currently the stakeholder can use the tool on the case study with moderate assistance (to extend C language support on the tool). Once better C language compatibility has been achieved for VeriFast, the stakeholder should require little to no assistance to use the tool.

Table 6 VeriFast validation results

Criteria	Evaluation
Applicability	**Can be applied with some assistance**
Verification process can be performed on HOMES	Yes, now with assistance, later independently
Verification results can be understood	Yes
Human Effort	**Doable, but significant**
Learning effort	1 month
Verification effort	0.5 annotation lines / code line
Comparison to manual approach	Not comparable
Flexibility	**Yes, moderate changes may be required**
Tool can be applied to different software modules	Yes, conditioned to language support
Effectiveness	**Tool is effective**
False negative rate	Very low, depends on annotations, % of verified code
False positive rate	Low, depends on annotations
Usability	**Easy inspection for non-experts**
Non-experts can check error reports on the tool	Yes
Performance	**Allows interaction**
Verification time	~compile time

Regarding human effort, experience shows that VeriFast requires significant investments both for learning and for the verification process. However, these can be mitigated with techniques such as the use of specialized verification engineers and selective verification of code. At any rate, an effort comparison with manual verification approaches cannot really be made, since these approaches fail to provide the level of assurance achieved by VeriFast. As for specific industrial criteria, we have confirmed that the tool is flexible and can be translated to other software systems without a major effort, though the level of C language support is a limiting factor for now. Also, experience shows that VeriFast is an effective tool, with a very low rate of false negatives and a reasonable level of false positives. In addition, the usability criterion is also met, when it comes to result collection – any developer with no knowledge of the algorithm can inspect bug reports in an intuitive way. Finally, the performance of the tool is good, with comparable delays to those shown by a compiler, allowing for interactive use and integration with IDEs. Overall, VeriFast is a formal verification tool with a strong potential for applicability in industrial practice. The tool's strengths are effectiveness for finding bugs that would hardly be identified with testing techniques, as well as the tool's flexibility and performance. It also has shortcomings with respect to language support and usability, which are currently being worked on.

Industrial considerations

For future work we recommend that VeriFast's support for C should be extended to enable further case studies in the domain of low-level system management components. Furthermore, research should focus on simplify annotations and reducing annotation overhead, as well as on providing automated tools to assist in annotation generation. The VeriFast tool would also benefit from debugger-like integration in commonly used integrated development environments such as Eclipse.

Since using the tool involves a significant effort investment, the decision of when and how to use it in a project should be taken with care. For highly sensitive systems, the high level of assurance given by VeriFast is more than likely to justify the costs by itself, but in other scenarios, including the HOMES gateway, that much assurance is not really required for the whole system. In those cases, applying the verification process on select modules and code sections will yield a reasonable level of assurance with considerably less effort.

Other considerations need to be taken when using VeriFast. The learning effort required to apply the annotations for the tool encourages using specialized developers that focus on the task of annotating code, possibly across different projects.

3.3.1.2 Security-by-Contract (SxC)

This section reports on the Security-by-Contract (SxC) methodology for the HOMES scenario as provided by WP6. In an OSGi platform such as the one used in HOMES, the SxC methodology enables each bundle coming onto the platform with a contract embedded into its manifest file. This contract contains details about the bundle's functional requirements, also listing permissions to access its services and packages. The PolicyChecker entity embedded on the platform checks during installation and monitors at run-time that the requirements of the bundle are satisfied. Thus the SxC methodology enables dynamic functionality and security enforcement in a changing environment when bundles from different providers are installed or removed while various provided services can be launched or stopped.

VALIDATION SCENARIOS and EXERCISES

Since the HOMES case study is considered a secondary case study for SxC (the primary one being ATM), and due to resource limitations, it has not been possible to implement a proof-of-concept prototype of the proposed SxC solution for HOMES. Instead, the HOMES SxC work was focused on providing a detailed description of how the SxC approach could be applied to this scenario. As a consequence, validation for this artefact was not based on experimental exercises, but on analysis and discussion of the presented methodology.

As discussed in D1.2, the SxC artefact on homes is focused on the security property of "secure extensibility", in combination with the "bundle lifecycle operation" change requirement. The goal is to improve the control of interactions between Java OSGi bundles with respect to permitted/forbidden information flow paths, particularly between bundles provided by different stakeholders.

For the validation exercise of the SxC artefact, the domain experts at TID were provided with a technical report describing in detail the Security-by-Contract methodology for OSGi platforms (such as the HOMES gateway). This description included the definition of bundle interactions and specification of a bundle contract, the definition of the OSGi platform security, the sketch of the architecture of the SxC extension of the OSGi platform and the checks to enforce security on the OSGi platform. The domain experts analysed this report in depth, and conducted a series of discussions with the SxC researchers. With the feedback and suggestions captured in this way, a new version of the report was generated. This version (2.0) has been included as part of deliverable D6.6. Finally, the methodology was evaluated following the criteria defined in deliverable D1.2.

VALIDATION CRITERIA

Two top-level criteria have been defined for the Security-by-Contract artefact in HOMES: effective usage (which includes applicability and human effort), and specific industrial criteria (including performance, effectiveness, and level of automation). The iterative decomposition of these criteria into measurable indicators is shown below.

Effective usage

- Applicability:
  - The methodology can be implemented by the researcher.
  - Bundle contracts can be used on HOMES by the stakeholder.
  - The methodology addresses relevant threats on HOMES.
- Human effort:
  - Framework deployment effort.
  - Learning effort.
  - Contract generation effort.

Specific Industrial Criteria

- Performance:
  - Delay on bundle lifecycle operations.
  - CPU and memory consumption.
- Effectiveness:
  - False negative rate.
- Automation:
  - Contract enforcement automation.
  - Contract generation automation.

A detailed discussion of the evaluation for each sub-criterion is provided below.

<u>Applicability</u>

The methodology can be implemented by the researcher

The HOMES SxC report provides an architectural and functional definition for a software module (the SxC framework) implementing the Security-by-Contract methodology on an OSGi platform. Though it has not been implemented due to resource limitations, TID domain experts have concluded that the OSGi framework presents no obstacles for the implementation of such a system. A likely implementation of the SxC framework would be as an OSGi bundle with permissions to access certain OSGI resources such as the service registry, the framework policy file, and the manifest files and permission files of bundles being loaded on the platform.

The methodology can be used on HOMES by the stakeholder

Once the SxC framework (as described in the previous section) has been implemented, it should be easy for the stakeholder to deploy it on the HOMES gateway and apply the Security-by-Contract methodology there. This will typically involve two differentiated tasks: SxC deployment and configuration, and contract generation.

Deployment and configuration of the SxC framework on a HOMES gateway would be performed by the network operator, and would involve the installation of a SxC bundle with the appropriate permissions. As any system update on OSGi, this could be done dynamically on a running platform without interfering with its operation. It has to be noted that even after deploying the SxC framework, usage of security contracts on bundles is strictly optional, so it wouldn't be necessary to have immediate upgrades of existing gateway services to comply with the methodology. A likely adoption scenario would have a progressive introduction of bundle contracts, starting with new services, and adding them to new versions of existing services as they were updated. Likewise, the inclusion of contracts in a bundle is backwards compatible, allowing the use of contract-enabled services on systems without SxC – though this would likely require additional security measures to compensate for the lack of SxC.

On the other hand, contract generation would be the responsibility of service providers, who would have to define contract permissions according to the needs of each specific bundle. As explained above, providers wouldn't be required to make drastic changes in their services at once, since it would be possible to gradually add contracts to new services and updated versions of existing services.

The methodology addresses relevant threats on HOMES

An important parameter to determine the applicability of this methodology on the HOMES scenario is its ability to address actual threats for a home gateway. The HOMES SxC report has identified several cases where Security-by-Contract allows service providers to define restrictions on bundle usage that are not possible using just the OSGi security framework. As an example, SxC can prevent the install of a bundle under a series of specific services are installed and running on the platform, in order to avoid denial of service attacks.

<u>Learning effort</u>

It is expected that the basics of SxC methodology will require a relatively low effort to learn – a developer should understand enough to create a simple contract after a few hours of study. However, learning to take advantage of the new security functionalities enabled by SxC may prove more complicated, and take a deeper study and some direct experience before developers get it right. This is difficult to estimate until we have had the chance to implement the methodology and see it at work.

Framework deployment effort

We estimate that the deployment of SxC once it is deployed will require very little effort for the operator of a network of home gateways. Installing a bundle (such as the SxC framework) on an OSGi platform like the HOMES gateway is typically a routine task that presents few challenges. SxC deployment should be transparent to and require no effort from the users of a gateway, except from perhaps providing confirmation for a system update. Note that in order for the SxC methodology to work, it's not enough to have the SxC framework deployed – security contracts also need to be added for each individual service, as discussed in the following section.

Contract generation effort

The generation of an SxC contract for an OSGi service should require a small amount of work from the provider of that service – likely in the order of a few developer work days per service. The developers would need to evaluate carefully the set of local permissions required by a bundle, as well as to decide the set of authorizations they are willing to provide for access to their packages and services. Though further bundle updates may require modifying the bundle contract, it is expected that this will be a relatively rare occurrence – so the maintenance work associated with SxC contracts can be considered negligible. One significant upside of the SxC methodology is the fact that it's optional and backwards compatible, which means that service providers do not need to immediately incorporate contracts into each running service as soon as SxC is adopted, but can introduce them gradually as new bundle versions are released. This allows for careful planning of SxC contracts, and to better distribute the workload.

Performance

Delay on bundle lifecycle operations

One of the potential drawbacks of a module like the SxC framework lies in the risk of noticeably slowing lifecycle operations down, thus degrading user experience. Due to the lack of an implementation, we have no hard data on this issue, but analysis strongly suggests that this will not be the case for SxC. On the contrary, the delay introduced by this approach should remain at very low levels, so that the use of SxC would be transparent to users.

The process of an SxC security check performed during lifecycle operations can be summarized as follows: the SxC framework parses SxC headers in the bundle manifest, checks for security stability, and checks for functionality stability. The operations performed are relatively cheap ones, such as parsing a short text file or comparing sets of permissions with bundle interactions, or sets of functional requirements with available services in the platform. It is safe to assume that such a security check will not have a large impact on lifecycle operation delay. It should be noted that, since the contract is included as part of the bundle manifest, it does not need to be signed separately, thus saving the need for a relatively costly decryption operation. Some future extensions for SxC currently under consideration include the Conflict Resolution component, which would handle conflicts between bundles in more complex scenarios based on a combination of framework-specific policy and system policy. Such extensions are likely to involve more complex logic and be more resource-intensive, so special care will need to be taken to preserve a good performance.

CPU and memory consumption

Though no specific measurements can be made until an implementation of the SxC framework is available for OSGi, we know that, as explained in the previous section, the basic working of SxC is based on simple operations which consume little CPU and memory. Moreover, this processing only takes place when a bundle lifecycle operation is performed

– since these operations are typically infrequent, the SxC bundle should remain inactive most of the time. This suggests that deploying SxC would be feasible even on devices with limited resources, like some home gateways.

<u>Effectiveness</u>

False negative rate

Since the checks performed by the SxC framework are fairly straightforward, we expect the methodology to correctly identify and apply the security policies and functional requirements associated with all bundles, barring errors in the contract or deliberate attacks. As a consequence, the false negative rate will be mostly dependent on human errors and attacks. The likelihood of poorly defined contracts leading to errors in SxC performance appears to be low, since the contract headers are not particularly hard to define. That said, special care will be needed so that developers gain a good grasp of the methodology during the learning period. Also, developers should be aware of the importance of contract definition and revise the headers accordingly. As for specific attacks exploiting vulnerabilities of the SxC framework, we haven't been able to study the issue, but this point should be taken into account once the module is implemented.

<u>Automation</u>

Contract enforcement automation

The SxC is expected to work on a fully automated manner, inspecting bundle contracts and enforcing contract policies without need for user interaction.

Contract generation automation

The generation of SxC contracts for OSGi bundles, unlike contract enforcement, cannot be automated in the current framework. Contracts consist of two parts: access control policy for access to the resources of a bundle, and functional dependencies of the bundle. These parts are to be defined by the bundle owner and developer, but they cannot be directly inferred from the bundle code. As a future development, it would be interesting to find ways to automate contract generation, or at least to have automated tools that guide developers in the process.

VALIDATION RESULTS

An overview of the validation results is shown in Table 7. The results can be summarized as follows. The SxC methodology can be applied by the stakeholder in complete independence, requiring a low level of effort – effort comparisons to a "manual approach" have not been provided since no manual equivalent to the SxC checks has been identified. As for industrial criteria, the methodology should offer a good performance without requiring much in the way of hardware resources. We also expect it to accurately identify when the security policies and functional requirements of a bundle are met, outside of human errors (poorly defined contracts) or potential deliberate attacks (the impact of which should be evaluated in a future study). Finally, the working of the SxC is fully automated and requires no user intervention, though the same cannot be said of the generation of contracts by service providers, which remains a purely manual process.

Table 7 Security-by-Contract (SxC) validation results

Criteria	Evaluation
Applicability	**Can be applied fully by stakeholder**
The methodology can be implemented by the researcher	Yes
Bundle contracts can be used on HOMES by the stakeholder	Yes, no assistance
The methodology addresses relevant threats on HOMES	Yes
Human effort	**Doable, low effort required**
Learning effort	Hours to understand, more to master
Framework deployment effort	Very low effort for operator
Contract generation effort	Low effort for provider
Performance	**Good, even on limited devices**
Delay on bundle lifecycle operations	Not noticeable
CPU and memory consumption	Low
Effectiveness	**Estimated as good enough**
False negative rate	Low, caused by human errors or attacks
Automation	**Automated operation, not development**
Contract enforcement automation	Fully automated
Contract generation automation	Not automated

Industrial considerations

The first point of this evaluation to be taken into account from an industrial perspective is the fact that no experimental results have been available due to the lack of a framework implementation. Nevertheless, we believe that the validation results acquired through theoretical analysis are fairly reliable, though we acknowledge the possibility of some unexpected interactions coming up once we have the chance to implement the SxC module over an OSGi platform and see it in practice. Thus, the results of this report could be expanded after this kind of practical experience, but we do not expect to find significant contradictions between the current results and those provided by experimental practice.

Overall, the current SxC specification for OSGi platforms looks like a promising starting point, which has some immediate applications as well as some very interesting research lines (such as Conflict Resolution) which will need to be studied in depth.

3.4 WP7 Testing

3.4.1WP7 ARTEFACTS

The validation activities for WP7 focused on one artefact, a methodology for managing test evolution.

3.4.1.1 Telling Test Stories (TTS)

The WP7 artefact validated in the HOMES scenario consisted in a methodology for managing test evolution. It can be decomposed into a language (a meta-model for expressing tests and connecting model elements) and a process (a test evolution management process for keeping tests and system synchronized.

VALIDATION SCENARIOS and EXERCISES

For the Telling Test Stories artefact in the HOMES scenario, the validation covered the change requirement of Core Security Module update, in combination with the security property of policy enforcement. In addition, the Bundle Lifecycle Operations change requirement was also covered, with the security properties “policy enforcement” and “security expandability”. The goal was to have the test model and, upon some eventual change on the system or requirements, identify the affected tests and derive tests suites. In this case the targets were policy-related and enforcement-related tests. The validation scenario for this artefact consisted in a walkthrough where the test evolution methodology was applied to the HOMES scenario. UIB provided a walkthrough script which was given to TID domain experts. The experts followed the script with assistance from UIB, and then evaluated the artefact according to the evaluation criteria defined in deliverable D1.2. The walkthrough started with a brief presentation to refresh the language and methodology, which the experts had studied previously. Later, a change scenario was defined, consisting on the introduction of a non-repudiation protocol for the HOMES scenario, which would be applied to third party service providers with low levels of trust. The initial model definition (including system, requirements and tests) was examined. Then, a change was introduced in this model, consisting on a new security requirement – the need to use non-repudiation for untrusted providers. In order to cope with this requirement, a new test was created, as well as new components and services in the system. The impact of the changes in system and requirements was evaluated for all tests. Finally, a test suite to validate this change was generated using OCL.

VALIDATION CRITERIA

Two top-level validation criteria have been defined for the Telling Test Stories artefact: Effective usage (which includes applicability and human effort), and specific industrial criteria (of which completeness and usability have been considered). The iterative decomposition of these criteria into measurable indicators is shown below.

Applicability

- Language:
    - Representation of HOMES scenario:
        - HOMES System can be represented.
        - HOMES Requirements can be represented.

        - HOMES Tests can be represented.
    - Can be understood:
        - Meta model can be understood.
        - Model representations can be understood.
        - State machines can be understood.
    - Can be used:
        - Models can be generated by the stakeholder.
        - Models can be updated by the stakeholder.
- Methodology:
    - Change propagation:
        - All relevant model changes are propagated to tests.
    - Test Selection:
        - Test selection criteria can define a variety of test suites.
        - Test selection is performed efficiently.
    - Can be understood:
        - Change propagation can be understood.
        - Test selection can be understood.
    - Can be used:
        - The change propagation process can be used by the stakeholder.
        - The test selection mechanisms can be used by the stakeholder.

Human Effort

- Learning:
    - Language and methodology Learning effort.
- Model:
    - Model generation:
        - System model generation effort.
        - Requirements model generation effort.
        - Test model generation effort.
    - Model updates:
        - Model update effort.
- Methodology:
    - Methodology usage:
        - Effort saved through change propagation.
        - Effort saved through test selection.
        - Overall effort savings.

Completeness

- All relevant entities in HOMES can be included in the model.
- All security properties can be included in test model.
- All change requirements can be applied to the test model.

Usability

- Test evolution usability:
    - All models can be handled by an integrated tool.
    - Model element states are easy to observe and edit.
    - Test types are easy to observe and edit.

    - Associations between requirements, tests, and services are easy to observe and edit.
    - Changes in test states are notified clearly.
- Test selection usability:
    - Test selection queries are easy to run.

A detailed explanation of each sub-criterion and the results obtained are provided in the following sections.

Effective Usage

The criteria of effective usage determine whether the language and methodology defined for this artefact can be successfully applied to the HOMES scenario, and the level of effort required for it. This effort is then compared to that of a manual approach to test evolution, to estimate the time savings obtained.

Applicability

Applicability has been separately evaluated for the two artefact components: the language (a meta-model for expressing tests and connecting model elements) and the methodology (a process to manage synchronization of tests and system during test evolution). Three aspects of the language were taken into consideration. The first one was its ability of the model to represent all relevant elements in the HOMES scenario, including the system, its requirements, and the tests. In second place, the degree to which this language can be understood by interested parties; we analysed this for the meta-model itself, as well as the model representations used, and the state machines associated to model elements. Finally, we examined the use of this language for generating models and updating them.

Regarding the methodology, we evaluated four categories of characteristics. The first two dealt with its main functionalities, which are managing the propagation of changes in the system and the selection of tests suites matching a set of changes. Within these functionalities, we examined the coverage of changes that could be propagated, as well as the variety of filter conditions that could be applied to test selection, and the efficiency of the selection process. The other two categories were the comprehensibility and usage of this methodology. Both of categories were analysed separately for change propagation and test selection.

HOMES System can be represented

As part of the validation walkthrough, a system model was created according to the TTS meta-model, providing an accurate representation of the system in the HOMES scenario. Since the system model defined by the TTS meta-model is actually a conventional UML component diagram, any service-oriented system should be easy to represent this way.

HOMES Requirements can be represented

During the walkthrough, a subset of the requirements from HOMES, including both functional requirements and functional requirements, were modelled according to the TTS meta-model. This provided an accurate representation of these requirements. Though the full requirement set was not used due to time constraints, we concluded that all of the requirements from the scenario could be represented this way with similar accuracy.

The representation of choice for this requirements model is a requirement hierarchy, where every requirement is composed of a name, an identifier and a brief textual description, and security requirements are associated to functional requirements as constraints to them. This is a representation that can easily be obtained from other common methods of representing requirements (such as use case descriptions), and which can be applied to all kinds of functional and security requirements in a service-oriented system.

HOMES Tests can be represented

In the walkthrough, several tests related to security and functional requirements in the HOMES scenario were modelled according to the TTS meta-model. The test models provided an accurate representation of these tests. The main representation of each test model is called *test story,* and consists on a series of service calls between test actors, and a set of assertions associated to these calls. In SecureChange, test stories are depicted in the form of UML sequence diagrams. The model of a test also includes test data, which is usually shown on a separate table. Though it was not possible to create test stories for a large number of tests due to time constraints, for the tests covered in the exercise we found that it was possible to generate suitable descriptions using test stories. We did not find any case where the language limitations prevented us from showing part of a test on a test story. Likewise, test data tables are straightforward representations which should be applicable to any kind of test, though size considerations might make them difficult or impossible to display in certain scenarios.

Meta model can be understood

In order to run the walkthrough, TID experts on the home gateway domain needed to study the meta-model defined for the test evolution methodology. It was concluded that this meta-model is intuitive and easy to understand for domain experts.

The meta-model has three clearly defined parts: the system model (containing both the actual system and the deployed infrastructure), the test model, and the requirements model. The internal structure of each sub-model shows all relevant components, and matches common representations used in software engineering. The relationships between model elements are consistent with reality, and can be followed intuitively.

One aspect of the meta-model that had to be studied more carefully was the relationship between functional requirements, security requirements, and tests, since it was linked to the particular style of security testing used in this methodology.

Model representations can be understood

The domain experts were shown the model representations used in the TTS methodology. The representations used for the system model and requirements model could be immediately understood with no prior knowledge of the methodology, whereas the test model representation (based on test stories) required familiarity with the methodology in order to be understood. In the TTS language, the system model is represented using a conventional UML component diagram, which is easily understood by most developers. Likewise, the requirement model shows a hierarchy of requirements defined by a name, an identifier, and a short text description – also fairly intuitive. We did come across some language elements, like assertions in the test model sequence diagram, which our experts were not familiar with, even though they appear in the UML 2 standard. That said, we found that once some basic concepts about the model and the structure of tests were explained, this representation of test stories could be understood without problems.

State machines can be understood

As part of the walkthrough, the state machines associated to requirements, services and tests in the TTS model were explained. We found that the model states and transitions had been defined in an intuitive way, and that the way state transitions triggered changes in other system elements could be understood without much effort.

Models can be generated by the stakeholder

Based on our experiences during the walkthrough, we estimate that the stakeholder would be able to generate models for system, requirements and tests, following the TTS meta-model, and with limited assistance from the researchers. As it has been mentioned in previous sections, the system and requirements models show very little deviation from conventional practices, so the stakeholder should be able to generate them without help. By contrast, test models based on test stories are likely to be a new concept for the stakeholder, which may require clarification or assistance when the methodology is first adopted. After acquiring some experience with the model, the stakeholder should be ready to use it in complete independence.

Models can be updated by the stakeholder

One of the tasks performed in the walkthrough consisted in introducing changes in different parts of the model to show how they would propagate. Changing the model after it has been defined is a straightforward activity which the stakeholder can perform in complete independence.

Once the full model for system, requirements and tests has been created, introducing a change on it, whether it is the addition, removal or modification or an element, should be an easy thing to do for the stakeholder, even without assistance from the researchers. Note that this refers specifically to the introduction of a new change in the model; the propagation of said change is covered in a separate section.

All relevant model changes are propagated to tests

Upon close examination of the meta-model and associated state machines, we have verified that the methodology is capable of propagating all kinds of relevant changes in the model towards the tests.

The state machine transitions track model changes such as additions, modifications and removals in either the system model, the requirements model, or the test model. Model elements are interrelated, and changes are propagated across elements, so that the state of the model should always accurately reflect whether a test is up to date.

Test selection criteria can define a variety of test suites

After examining the range of available filter criteria for automated test selection, we have verified that it is possible to select tests according to a wide range of conditions, allowing for a fine-grained configuration of test suites.

The OCL-based test selection mechanism can return a subset of the existing sets based on simple criteria such as test type (regression, evolution, stagnation), but it can also consider the state of other model artefacts. This allows selecting tests based on their association with a certain service, functional requirement or security requirement, or on a combination such conditions. This basically covers any variable in the model which might be relevant in the selection of a test.

Test selection is performed efficiently

One of the steps of the walkthrough included selecting a series of tests that matched certain conditions. We found that the definition of test selection criteria can be performed easily through short scripts, and that the execution of said scripts returns the desired results almost immediately.

A typical OCL query to generate a test suite from the range of available tests takes only a few (3-4) lines of code. More complex queries can be longer but should rarely involve more than half a dozen lines, since each condition can be expressed in a single line. Running the query doesn't take longer than a couple of seconds, at worst.

Change propagation can be understood

The walkthrough showed the process of propagating a change in the model using this methodology, in a clear and easy to follow way. We have reached the conclusion that change propagation can be well understood by the stakeholder, particularly after a practical demonstration.

The mechanism used to propagate changes in the model by means of triggers in state machines is not excessively complex, and one can get a good impression of how it works after seeing it in action with one or two examples.

One complexity that we found associated with this propagation of changes through state machines was that, while individual state changes and their associated triggers were not hard to follow, a change could often propagate across multiple system elements, and tracing it by hand could become a cumbersome task. This means that having access to a dedicated tool that keeps track of changes across the model is strongly recommended in order to use this methodology.

Test selection can be understood

Our experience during the walkthrough shows that the most basic form of test selection (e.g. using test type) can be immediately understood. Further refinements on test selection criteria such as finding tests associated with a requirement or service require some familiarity with the methodology and language, but aren't otherwise particularly challenging to understand.

The change propagation process can be used by the stakeholder

After using the change propagation methodology, we believe that the stakeholder can follow this process independently, provided that adequate automated tools are available.

The existence of automated tools to keep track of the various state machines and change triggers is a critical requirement to implement the methodology on any non-trivial system.

For the walkthrough, we were able to perform the changes by hand because only a limited amount of change operations and model artefacts were considered, but in a real environment this manual approach would be unfeasible.

That said, once the more mechanical steps and state data tracking are taken care of, change propagation is intuitive, and does not present significant challenges to the stakeholder.

The test selection mechanisms can be used by the stakeholder

After trying the test selection mechanisms during the walkthrough, we have concluded that the stakeholder can use these mechanisms in complete independence, provided that a modelling tool with OCL support is available. Since test selection is based on OCL queries over the model artefacts, the use of modelling software compatible with OCL is mandatory. This should not pose a problem, since such modelling tools exist in the market.

With an adequate modelling tool, the test selection process is reduced to defining simple OCL queries and running them, which can be easily managed by the stakeholder, even without assistance from the researchers.

Human Effort

The level of human effort required by this artefact has been evaluated as three separate categories: the learning effort, the effort associated with the usage of the model, and the effort associated with the application of the methodology. Of these, the model-related effort has been divided into generation effort for system, requirements, and test models, and effort of model updates. Finally, the methodology-related effort has been divided into saved effort due to change propagation, saved effort due to test selection, and overall saved effort considering the methodology as a whole.

Learning effort

We observed that it takes between 4 and 6 hours of study for a developer to be familiar enough with the language and methodology to start using it effectively. This includes a few practical exercises, and assumes previous experience with the UML language and modelling tools.

System model generation effort

We expect the effort for system model generation with this methodology to be very low. In practice, most software projects should already have a system model with a very similar structure to the one defined by the TTS meta-model, and this existing model could be reused here with minor adaptations.

Requirements model generation effort

The effort to generate the requirements model (including both functional requirements and security requirements) should be low. At worst, we should expect the system requirements to be defined as a written document. The requirements model used here is relatively simple, with each requirement defined by a name, an identifier, and a short text description, as well as its associations towards other requirements. Generating this model from the written document is a straightforward task, consisting mostly on copying names and descriptions to the modelling tool.

Test model generation effort

We estimate that most of the effort of implementing the test evolution methodology will lie in the task of generating test models. Unlike the models for the system and requirements, we cannot expect to have an existing test model that can be reused – the model will need to be created specifically for use with this methodology. The most costly task of model generation is the creation of a test story for each individual test. With the sequence diagram representation used in SecureChange, we expect that this can take between half an hour and two hours, depending on the complexity of the test. By contrast, defining associations between a test and the system services and requirements shouldn't take more than a few minutes per test. These effort estimations describe the worst case scenario, which assumes the creation of test models for tests that are already designed and implemented. This can happen, for example, when adopting the language and methodology for a project after it has started. However, it is also possible to follow a model-based test creation process, where generating a test story is one step in the design of a test. While the time to generate a test model does not change, the effort can be partially compensated by reductions in design time for the actual tests.

Model update effort

Keeping the models for system, requirements and tests up to date requires some effort, which has to be added to the cost of generating the initial models. In general, the update cost of each model is proportional to its generation cost. System model updates taking little or no extra effort (in the assumption that an updated system model is required whether or not the test generation methodology is used), whereas updates of the requirements model takes up a higher but still small amount of effort, and test model updates involve a significant time investment.

Effort saved through change propagation

The change propagation process saves effort (compared with the manual approach) each time a change is introduced in the system, provided that the system is not trivial and that there is a moderate number of tests. This effort savings grows with the complexity of the system and the number of tests. Under the manual approach, whenever a change is introduced in the system or its requirements, the developers need to run a series of checks.

For a change in a system service:

- Check unit tests that are assigned to the changed element.
- Check for any test that can include calls (whether direct or indirect) to the changed element.

For a change in a requirement:

- Check tests specifically assigned to the use case of that requirement.

Check for any test that can be related to that requirement. With no explicit test model, it is still easy to find the unit tests assigned to a class, or the tests assigned to a requirement use case; it should be possible to identify these tests in a matter of minutes. However, there is no clear way to determine which tests have a direct or indirect relationship to a changed system artefact or requirement. One has to go over the list of existing tests and revise their code (to find calls to a changed element) or their specification (to find if they test a functional or security requirement). This process is prone to errors, and grows in effort and complexity with the amount of services, requirements and tests in a system. By contrast, the test evolution methodology propagates any change in the model, marking the affected tests quickly, and with no additional effort other than updating the model – since the propagation process should be mostly automated. It also has the advantage of being much less prone to errors than the manual approach. Though it is difficult to give hard figures for the time savings introduced by the methodology, we can estimate that:

- The methodology saves effort with each change operation. This will happen as long as the cost of updating models is lower than the cost of looking for affected tests by hand – which will be true unless working with a very simple system with very few tests.
- The effort saving increases with system complexity and the number of tests.

Overall, the effort savings for change propagation will depend on the frequency of changes in the system, the complexity of the system, and the number of existing tests.

Effort saved through test selection

The test selection process saves effort (compared with the manual approach) each time we select a set of tests for a test suite according to a non-trivial filter condition. This effort savings grows with the number of tests in the system. Under the manual approach, selection of a test suite involves going over the list of existing tests, and determining which ones should be included in the suite, on a case by case basis. The complexity of this task depends on the granularity of the filter conditions:

- If we just want to filter by test type (evolution, regression, stagnation), selecting a suite is almost immediate. It is possible to have tests organized by type (e.g. by having them in separate folders for each type) and easily take all tests of the chosen type for a suite.
- If we want to filter tests calling a specific service or associated to a specific requirement, we may need to go over the whole list of tests and check their code and specification. The cost of this operation is proportional to the number of tests.

Overall, the effort savings for test selection will depend on the frequency with which one needs to select tests for a suite (which usually coincides with the frequency of changes in the system), with the number of existing tests, and with the granularity of the filter conditions we want for the test suite.

Overall effort savings

The critical measure to evaluate the success of the test evolution methodology is the overall amount of effort compared to the manual approach, after accounting for the cost of model generation and updates, change propagation, and test selection. The following factors need to be taken into account when comparing the effort required by the test evolution methodology and the manual approach:

- There is a significant initial overhead for the methodology, due to model generation.
- There is a small overhead for the methodology for each change in the system, due to model updates.
- The methodology saves effort for each change due to change propagation. This savings increase with system complexity and the number of tests.
- The methodology saves effort for each generated test suite due to the automated test selection. This savings increase with the number of tests. We can assume that test suites will need to be generated whenever the system changes.

The evolution will result in saved effort whenever the total savings due to change propagation and test suite selection exceed the initial cost of model generation. In general, this will occur in complex systems with many tests, if they are subject to many changes.

Specific Industrial Criteria

Two specific industrial criteria have been considered for this artefact in the HOMES scenario: completeness, the ability to include all relevant entities in the system, and usability, ease of use for the methodology by the stakeholder.

Completeness

In order to evaluate the completeness criteria, we have considered separately how it applied to the model of the HOMES scenario, to the security properties, and to the change requirements under study.

All relevant entities in HOMES can be included in the model

We have determined that the system, requirements and tests used in the HOMES scenario can be fully represented using the defined meta-model. This has been covered in the section describing the applicability criteria. For a detailed explanation, see the discussion of the following criteria:

- HOMES System can be represented.
- HOMES Requirements can be represented.
- HOMES Tests can be represented.

All security properties can be included in test model

The following security properties have been the focus of the work in the HOMES scenario: secure extensibility, policy enforcement, resilience to trust changes, and security expandability. Each of them can be represented in the model as a security requirement associated with one or more functional requirements. The security properties of policy enforcement and security expandability were specifically covered for this artefact, though all of them are representable in the model.

All change requirements can be applied to the test model

Two main change scenarios have been considered for HOMES, called Core Security Module Update and Bundle Lifecycle Operations. We have verified that the test evolution methodology can be successfully applied to both of them. The change requirement called Core Security Module Update deals with changes to critical security services within the system. To apply the test evolution methodology on such changes, the system model needs to be updated to reflect the final versions of the modified security services. Likewise, the change may involve updates to security requirements, which will need to be applied to the requirements model. After updating the models, the changes can be propagated as usual to determine which tests are affected by the change. The change requirement called Bundle Lifecycle Operations refers to the change of trust relationships for certain third party service providers. Using the test evolution methodology, this means changing the security requirements associated with certain services, to reflect the new level of trust assigned to that provider. Once these updates are included in the requirements model, the methodology can be applied to propagate the changes and determine the affected tests.

Usability

Practical application of the test evolution methodology requires using dedicated software tools to assist the developer in the creation and maintenance of models, during change propagation, and while selecting test suites. The usability criteria relates to the ease of use of this software. It has not been possible to evaluate this criterion during the validation exercise due to lack of availability of these tools. Instead, in this section we describe the properties that we have found necessary for the usability of a tool assisting the test evolution methodology. In addition, we discuss the expected complexities in implementing said properties.

We identified six properties as necessary for the usability of the methodology: having all models handled by a single tool, easily observing and editing of model element states, test types, and associations between elements, clear notifications of test state changes, and easily running of test selection queries. In addition to these methodology-specific requirements, the functionalities of a general purpose modelling tool should also be available.

All models can be handled by an integrated tool.

The ability to manage the system, requirements and test models in a single integrated tool is required for the usability of the methodology, since using multiple tools in parallel is cumbersome and requires additional effort for synchronizing the different models. Achieving this requirement should be possible for any modelling tool. UIB has had successful experiences achieving this with MagicDraw and a methodology-specific profile.

Model element states are easy to observe and edit.

One key parameter in the methodology model is the state associated with certain elements. Being able to read and write these states should be a very common operation for users. This issue depends on the GUI implementation; ideally, element states should be readable while looking at a general view of the model, or at worst be available through a single click. As for state editing, this should be carried out and traced automatically.

Test types are easy to observe and edit.

The type of a test (evolution, regression, stagnation) is another commonly accessed parameter in the methodology. Again, this property is GUI-dependent. Test types should be readable when looking at a test in the model, or be available through a single click. Changes in test type should be carried out and traced automatically by the tool.

Associations between requirements, tests, and services are easy to observe and edit.

The methodology defines a series of associations between requirements, tests, and services, which users need to be able to browse and edit with ease. The tool should be able to provide a list of all associations for a given model element by selecting that element and making a few clicks. Likewise, there should be a way to obtain a list with all associations of a given type (such as test-requirements assignments) in the model.

Changes in test states are notified clearly

When the test evolution methodology is applied, a change in the model is propagated, resulting in changes of state and type for a series of tests. The tool should provide clearly visible notifications whenever a test becomes non-executable, or when a requirement stops being under test, so that the user can fix the situation as soon as possible.

Any implementation of the methodology should be able to display test state at any time, and allow users to query for recent changes. In addition, in order to best address this issue, the types of changes mentioned above should be automatically displayed in a GUI element associated with model errors or pending tasks.

Test selection queries are easy to run.

The selection of tests to generate a test suite is a common operation that should not be time-consuming. Optimal usability during test selection would require having an integrated OCL interpreter within the modelling tool to assist in the OCL script generation.

VALIDATION RESULTS

An overview of the validation results is shown in Table 8. The results can be summarized as follows. The applicability criteria for the HOMES scenario is favourable, in that the stakeholder can apply the language and methodology almost independently – we estimate that a small degree of assistance from researchers may be needed initially for model generation, while the stakeholder gets used to generating test stories. In addition, the use of dedicated tools to assist with the methodology should be a requirement for all non-trivial scenarios.

As for the human effort criteria, we have determined that the methodology can result in effort savings as long as certain conditions are met. Specifically, complex projects with large numbers of tests that are expected to suffer frequent changes are the most favourable scenario for the methodology. The development of a home gateway architecture such as the one represented by the HOMES scenario would meet those requirements. For scenarios with fewer tests, infrequent changes, or lower complexity, the methodology will yield lower savings, or even result in greater overall effort – so its application will need to be evaluated on a case-by-case basis.

Regarding the completeness criteria, we concluded that all security-related entities in the HOMES scenario can be included in the language and methodology.

Finally, it was not possible to evaluate the usability criteria due to lack of access to software tools on which to measure this usability. However, we have defined a set of properties that will have to be met to achieve good usability, and provided guidelines to implement them.

Table 8 Telling Test stories (TTS) validation results

Criteria	Evaluation
Applicability	**Stakeholder can apply with assistance**
HOMES System can be represented	Yes
HOMES Requirements can be represented	Yes
HOMES Tests can be represented	Yes
Meta model can be understood	Yes
Model representations can be understood	Yes
State machines can be understood	Yes
Models can be generated by the stakeholder	Yes, with some assistance
Models can be updated by the stakeholder	Yes
All relevant model changes are propagated to tests	Yes
Test selection criteria can define a variety of test suites	Yes
Test selection is performed efficiently	Yes
Change propagation can be understood	Yes
Test selection can be understood	Yes
The change propagation process can be used by the stakeholder	Yes, with automated tools
The test selection mechanisms can be used by the stakeholder	Yes, with modelling tool
Human Effort	**Can save effort under certain conditions**
Learning effort	4-6 hours
System model generation effort	Low
Requirements model generation effort	Low
Test model generation effort	Moderate (~1 hour/test)
Model update effort	Low
Effort saved through change propagation	Depends on system complexity, # of tests, # of system changes
Effort saved through test selection	Depends on filter conditions, # of tests, # of system changes
Overall effort savings	Good for complex systems with many tests, frequent changes
Completeness	**All relevant entities can be included**
All relevant entities in HOMES can be included in the model	Yes
All security properties can be included in test model	Yes
All change requirements can be applied to the test model	Yes
Usability	**Not evaluated**

Industrial considerations

An important consideration when applying the test evolution methodology in an industrial environment is the availability of a software tool that supports it. Currently, UIBK works on the methodology using the MagicDraw modelling tool with a profile for the methodology to manage the models, along with the MoVE academic prototype to support the transitions of state machines. A stakeholder looking to implement the methodology should be prepared to adopt these tools or, alternately, work on the adaptation of its modelling tools of choice.

Another factor to take into account is at which stage of a project the methodology is to be adopted. The greatest efficiency will be achieved when applying the language and process from the beginning of a project, since the test model generation effort can be leveraged if a model-based testing approach is used. Adopting the methodology on a project already in development or even maintenance is also possible, but will involve significant effort, as the models will have to be generated while the system is subject to changes.

4 POPS CASE STUDY

This section describes the validation activities and results of the SecureChange artefacts for the POPS case study. This case study involves the software of an UICC card made of an OS, a Java card platform that executes applications (applets), a set of applets and a Globalplatform layer responsible of the card content management. The global scenario of this case study concerns a change of the software embedded on the card that results from adding a new application on the card or updating the platform layer due to an evolution of the corresponding specification. We are then in the context of **software update** as change requirement. The overall objective is to provide means that will facilitate the assessment of those changes with respect to specific security properties.

4.1 Validation Organization and Conduct

The life cycle of an UICC card involves several actors: the developer of the software (the card manufacturer), the developer of the application, the end-user (the card holder) and the card issuer. In the scenario of mobile payment considered in the SecureChange project, the card issuer is the mobile operator. Therefore the validation activities will simulate the role of these actors:

- The developer when he has to develop a new application.
- The card manufacturer when he has to update the card software platform.
- The card issuer when he has to load a new application on the card.

The card lifecycle phases covered by this scenario are application development, card software validation and applet post-issuance loading. For each role above, the SecureChange project provides means (artefacts) to be used to check or to assess the impact of the change. The application developer will use a static analyzer tool, from the **WP6**, to check specific security properties for its application. The Card manufacturer that updates its card software will formalize its change using artefact from the **WP4** and will validate the change using artefact from the **WP7** and finally the card issuer will use artefacts from the **WP6** to “accept” a new application on its card.

The validation activities consist in playing the role of these actors for using these artefacts and evaluate them in realistic industrial contexts. For each artefact, generally a specific tool, the security engineer takes the role of the developer, and figures out a wide usage in the R&D. The people we have involved in the validation activities have several kinds of background and expertise (security engineer, PHD formal methods specialist, tool developer, etc) in order to have the most representative sample of a generic R&D population. This validation is intended to confirm the feasibility studies described in D1.2 and possibly provides recommendations.

4.2 High-level objectives

The change requirements considered in the SecureChange project for smart cards case study have been taken from the concrete need in the day to day life of the security engineer. Nevertheless, the strategy developed in the project does not take into account the specific change of card software when the card is already in the field and that requires generally patching the card after a bug or a defect is discovered in post-issuance. This kind of change requires specific mechanisms that must be planned at card construction and are not deployed for all the cards. Therefore, we preferred to tackle the classical use case looking for methodologies and tools that allow avoiding this kind of situation.

The **change requirements** that we consider for an UICC card are **the software update** that results from the development and/or loading of a new application on the card and the update of the software platform due to an **evolution of the specification**.

The high-level objective is to prove/demonstrate/test that:

- The new application preserves (does not break) the consistency of the existing and implemented security policies.
- The update of the Globalplatform implementation vs. the new version of the specifications preserves.

For that, WP3, WP4, WP6 and WP7 collaborate to preserve **information protection** and **deny of service** properties. More precisely, WP6 and WP4 provide tools and associated modelling and verification techniques to check that the "new" application to be loaded on the card verifies the information flow properties. The WP7 and WP3 will provide test suites and traceability techniques to check that a new implementation with respect to an evolution of the specification of the underlying platform respects the **information access control** properties.

4.3 WP6 Verification

The following section describes the validation of the artefacts provided by the WP6 for verification. Two approaches are proposed, an off-card verification used during the development phase and on-board verification of an application on the device after its loading. Therefore, the main validation criteria are ***usability*** and ***scalability*** for the off-card tool and ***scalability*** and ***integration*** for the tools to be integrated into the card platform.

4.3.1 Development-time Verification of JC Applets

VALIDATION SCENARIOS and EXERCISES

The development-time verification allows dealing with the software update as a change requirement. The scenario for this kind of change concerns the developer or the product builder that is developing (or receiving from an application provider) a new application to be loaded on the smart card (this scenario may also concern the validation manager that will assess the application). The developer is then the secure software designer that has to ensure a set of security properties for its application. The main property considered is **denial of service**: this robustness property is refined (concretized) into properties to be checked on the source code, i.e.:

- Absence of run-time exceptions.
- Absence of infinite loop.
- Memory bound, avoiding memory overflow and memory access especially update operations due to the durability of the EEPROM and the Flash.

The exercise is that the developer will check the properties above using the Verifast tool. The tool allows using the developed source code of the applet and the properties are expressed using annotations (kind of comments). The two properties considered are the *absence of runtime exceptions* and *infinite loops*, the last one, memory consumption, has been delayed according to the feasibility studies (cf. D1.2). The validation criteria have been defined using this scenario and conducting the exercise using several applets.

VALIDATION CRITERIA

The validation exercise consists in using the tool during the design and development phase of the application. The validation criteria defined are:

- Scalability and performance.
- Modelling & expressiveness.
- Proof capability.
- User-friendly interface.
- Integration in the industrial development process.

VALIDATION RESULTS

Scalability & performance

Code size

This criterion is about the size of the application that could be tackled by the tool, and generally the Java Card applications obey to a set of constraints.
The tool was tested on an applet "phone book" of ~ 2000 lines of code for which the annotation process took about six hours. However, much of that time was spent trying to understand the error messages raised by the tool when the annotations were incomplete or incorrect. But these disadvantages could be minimized by the automatic generation of some annotations (see Degree of Automation) and the understanding of the errors messages will be facilitated by a regular use of the tool. Nevertheless, the debugging phase of an application code is an important step that is generally re-used for a family of applications.

Modularity of the verification

The global verification process (i.e. annotation + automatic verification) is not modular. Even if the automatic verification itself is modular, the user will not necessary provide every class and interface of the source. This means that the used APIs must be annotated before starting the verification process. For methods that are not included in the audit process (in our case, methods that do not contain loops and cannot raise a NullPointerException), we still need to add a "minimal" contract so that the tool runs, e.g.:

```
//@ requires true;
//@ ensures true;
```

Speed of the verification process

This criterion may impact the usability of the tool in order to be integrated in the development life cycle of an application. The automatic verification process itself is fast and negligible in time compared to the annotation process (and even more compared to the development time). Although it is only a small part of the global verification process, it is one of the advantages of the Verifast tool.

Change tolerance

This criterion evaluates the impact of a code change on the annotations or on the model describing the application. With the VeriFast tool, like in all code verifiers, a change in the source code may require a change of all the annotations impacted by this change, because if the annotation does not correspond anymore to the actual behaviour of the application, an error will be raised during the verification.

Properties Modelling

This criterion evaluates the expressiveness of the language to model the properties. In particular, we evaluate the "pollution" of the code by the addition of annotations. The amount of annotations must not exceed a certain percentage of the code. With respect to the properties targeted in this exercise (no NullPointerException & no infinite loop), the annotations take about 100 lines of code (5% of the code). This does not take into account the annotations added in the APIs. In fact, the more properties there are to prove, the more the annotations will grow. The amount of annotations may constitute, for a full functional verification, the same size than the source code being analyzed.

This does not constitute necessary a disadvantage if we consider the completeness of the set of properties to be checked. As the annotations are treated as comments by the compiler, we can consider the set of annotations as the formal model of the application and reuse it for a family of applications.

Proof capability

Degree of Automation

The proof of the properties expressed by the annotations is completely automatic and does not require any interaction from the user. However some of the annotations could probably be automatically generated by the tool (like, for example, the "object validity" predicate, i.e. the object and its fields are allocated in the memory after the constructor has been called).

User interaction

This criterion concerns the ability of the tool to provide hints to complete the proof/to debug the annotation. When an error occurs during the verification (for example a syntax error in an annotation or a missing hypothesis) the interface directly points towards the error location in the source code and displays the corresponding error message. The nature of the error is explained by the error message and in some cases, some generic "fix hints" are provided, not specific to the current code. In other cases, no hint is provided at all. As previously explained, the feedback given back to the developer is important when we are verifying security properties. The Verifast tool could be improved in this promising direction.

User-friendly interface

The required knowledge and expertise of the developer required to use efficiently the tool:

- The basics of Java Card but as the exercise are meant for the JC secure code developer, this requirement is obsolete.
- The specification of the verified code when the objective is the formal verification of the functional properties.

- Knowledge of separation logic, pre-conditions, post-conditions, grammar and keywords of the annotation language to be able to write JML-style annotations. This is necessary to debug/modify contracts when the proof does not terminate. This will also depends on the quality of the feedback provided to the user.

This last requirement is more restrictive as the developer is not necessary a Formal Methods expert. But generally in the security field, the high educational quality of the security engineer allows to learn rapidly the missing knowledge.

Integration in the industrial development process

The criterion represents a summary of the validation exercise, e.g. identifying the missing features to fully integrate the tool in the industrial application development process.

Improve the API management

The provided APIs miss extensive annotations for every basic Java Card API and commonly used APIs (e.g. sim.*, usim.*, uicc.*). Since all the APIs must be annotated before running the tool on an applet, the user is required to have the source code of every API used in his applet. It is not the case generally, for some APIs the developer only has the .jar and .exp files. The tool could assume that, for every un-annotated method, this method has the minimal contract:

```
//@ requires true;
//@ ensures true;
```

This would enable the user to add any missing API (like a proprietary API or an external API) without having to annotate all its methods. An automatic generation of this minimal annotation by the VeriFast tool would allow an incremental annotation process.

Allow an incremental use of the tool

This feature concerns the ability to run the tool on an applet before having annotated his whole source code and APIs. An example of an incremental annotation process may be:

1) Run the tool with no annotation and examine the result.
2) Generate the minimal annotations with the tool.
3) Add the necessary annotations to prove the absence of NullPointerException.
4) Add the necessary annotations to prove the absence of infinite loops.
5) Add the necessary annotations to prove the functional correctness.

In the current state of VeriFast, the step 1 is not possible, and the step 2 must be done by hand which is very time-consuming.

Enhance the tool's documentation and output

The documentation of the annotation language should be greatly improved. The reference manual currently contains a simple description of the language grammar; however there is neither description nor any example in the manual. Moreover, during the annotation process of a Java Card applet, many predicates must be used that are already defined in the APIs such as "system()", "current_applet(...)" or "array_slice(...)". These predicates are not documented anywhere, the user must look into the examples of Java Card annotated applets contained in the tool to see how and when they are used. The output of the tool should also be improved when:

- The proof does not terminate: the "fix hints" could include examples of annotated code triggering the error and how to fix it for the most frequent cases.

- The proof terminates: the only output of the tool when the proof terminates is a status bar saying "0 errors found". There is no way for the user to see how the tool managed to prove his annotations. This could be problematic, specifically for security validation, because it may correspond to nothing has been done by the tool and still outputs "0 errors found", or there could be an error in the tool's implementation. For a developer or an evaluator to trust VeriFast, a detailed output of the proof scenario which correctness can be manually checked is mandatory.

4.3.2 On-Device Verification

VALIDATION SCENARIOS and EXERCISES

The on-board verification is a security mean that addresses the **software update** change requirement and the target security property is the **information protection** through information flow control. The validation scenario consists in taking the role of the card issuer that will decide the loading of an applet on its card if and only if the applet does not break the security policy of the card.

For that, the SecureChange project developed software as a security component of the card platform that will verify/check that the application respects the security policy before being allowed to be installed and executed. It is one of the most challenging techniques for smart cards due to the resources constrained and aggressive performance features for this type of devices.

The scenario involves several actors: the application developer that will develop the applet and writes down its contract, the card issuer that is the owner of the security policy implemented on the card, against which the application will be checked and the card manufacturer that integrates this new on-device checker as one of the component of the platform.

Two approaches have been developed in the SecureChange project: The first approach consists in checking that the applet, after being loaded on the card (byte-code format) and before its installation (linking), respects given **information flow control policy**.

The second approach is a **security-by-contract** approach: The methodology is based on "contract" technique. Each Java Card applet comes with a contract that describes which services it needs from the other applets and which services it proposes to the other applets. This methodology is based on two "tools": a claim checker and a policy checker. The security-by-contract approach is particularly challenging as several attempts in the past have been done without success with respect to the scalability criteria. The targeted security properties are:

- No illegal access to a service: to avoid collusion between applications when using the services provided by other applications.
- Non-interference to avoid illegal information flows between applications.
- Global control of interactions: no illegal sequence (of method calls).

The exercise consists in integrating the tools as a component of the card platform. For that a specific UICC implementation has been chosen and we developed the necessary code to let this new component to collaborate with the other components.

VALIDATION CRITERIA

In both approaches, "tool" (piece of software) will be added to the card platform that will run on the card. Therefore the criteria against which the tools to be integrated will be evaluated are essentially:

- Footprint.
- RAM consumption.

Those tools will check an application that embeds a contract in its code. Therefore, an additional validation criterion is how the policy will be integrated to the applet's code. More precisely:

- How the policy is expressed.
- How the policy is attached to the bytecode of the applet.
- Overhead on the additional code size to be stored on card.

The feasibility studies conducted (described in D1.2) showed that due to time and resource constraint, some implementation on card platform is not possible. In that case, the criteria will be estimated by extrapolating the results obtained on a PC implementation of the methods and tools. When the PC implementation is not available, then only the theoretical complexity of the approach is evaluated.

VALIDATION RESULTS

Security by Contract

The SxC tool, developed by UNITN, is made of:

1. A Java Card applet that stores the on-card security policy (i.e. the policy of all loaded applets) and manages the update of this policy if a new applet is loaded or an existing applet is deleted.
2. Two C modules:
    a. ClaimChecker that checks if the security policy claimed by an applet is consistent with its binary.
    b. SxCInstaller that checks if this policy is consistent with the card security policy.

To evaluate those components, a dedicated library (apiobc) has been developed by GTO to provide the API services to SxC modules. The SxC tool has been integrated and tested on the PC simulator of a GTO platform. The tool has also been compiled on an IC architecture to measure its exact footprint.

Footprint on the platform

The footprint corresponds to the space occupied by the tool (API included). The footprint also includes the on-card security policy that must be stored permanently in the card. The footprint of these components is measured on an industrial IC:

- `PolicyApplet`: 4 Kbytes.
- `API`: 856 bytes.
- `SxCInstaller`: 1004 bytes.

- `ClaimChecker`: 976 bytes.
- On-card security policy: this component depends on the number of applets and the number of services participated into the security policy.

  The size is expressed by the following formula:

  `NumApplets*(2+3*NumApplets+2*NumServices)`

  where `NumApplets, NumServices` are the numbers of applets and services.

NVM consumption

The NVM (non-volatile memory) is used to allocate C structures and persistent Java arrays. In SxC, PolicyApplet allocates two buffers of 135 bytes and 255 bytes respectively. ClaimChecker and SxCInstaller do not consume any NVM because all structures are temporary in RAM.

RAM consumption

RAM is used to allocate transient Java arrays and local C variables. PolicyApplet does not use any transient array. The local variables of ClaimChecker and SxCInstaller consume less than 100 bytes.

Overhead (space and time)

The **space** overhead includes the added space to the standard applet in order to express the claimed security policy. Typically, that corresponds to the Contract custom component. The size of this component depends on the complexity of the security policy by the following formula:

```
12 + 2*num_provide +15*num_calls + (15+2*NumApplets)*secrules
```

where `num_provide` is the number of services provided by the applet; `num_call` is the number of services this applet calls; secrules is number of access rules to service. On the specific applet that we use, this size does not exceed 10% of the original CAP file.

The **time measurement** has been done on a PC simulator and hence, its absolute value is not representative. Overall, on applet loading, SxC time overhead is around 15% (of the total time needed to load and link a new applet).

Methods and tools to define the security policy

There is no defined language to describe the security policy. Instead a GUI (graphical User Interface) is provided. The GUI allows the definition of the policy on the services implemented into an applet. The policy is then attached to the original CAP file as its Contract component. The defined policy can be exported to a binary file for future use.

However, using the current GUI to define a complex security policy is not very easy. The usability of the GUI needs to be improved. In particular, the internal tokens of the services should be replaced by the use of full-qualified names. Furthermore, displaying the current policy of an applet is a useful feature that should be added to the GUI.

EVE_TCF

EVE-TCF developed by INRIA-lille partner implements the transitive information flow verification. *EVE-TCF is not yet tested on a GTO platform due to a licensing issue.* We have however compiled it with the source-code of the platform to ensure that its architecture and development are compatible with the platform. EVE-TCF is composed of:

1. A Java Card applet (IFCInstallerApplet) that stores the on-card security policy (i.e. the policy of all loaded applets) and manages the update of this policy if a new applet is loaded or an existing applet is deleted.
2. A C module (verif) that checks the policy of the newly loaded applet with respect to the on-card security policy.

EVE-TCF uses the same dedicated API library to access GTO platform components. It means that no additional development is required for integration.

Footprint on the platform

The Java applet has a binary size of 3363 bytes. Because the tool is not compiled on an IC architecture (on-target compilation), it is not possible to measure the footprint of its C component. We estimated this footprint by an extrapolation approach, based on the size of object file when compiled on a PC. Actually, the size of *veriof.obj* is the same than the *SxCInstaller.obj* one and *ClaimChecker.obj (see Footprint on the platform)*. As a result, the footprint of *verif.c* on an IC is more or less that of SxCInstaller and ClaimChekcer (around 2000 bytes). The API footprint is the same as that used for SxC (i.e. 856 bytes). On-card security policy: this component depends on the number of security domains, the number of packages in these security domains, the number of classes in these packages and the number of the methods in these classes. The size is expressed by the following formula:

```
10*NumDomain + (3+(5*NumMethods+5)*NumClasses)*NumPackages)
```

On the card platform of this exercise (2 security domains), the on-card security policy takes less than 5% of the size of the total CAP files.

NVM consumption

The NVM (non-volatile memory) is used to allocate C structures and persistent Java arrays. In SxC, IFCInstallerApplet uses 255 bytes for its policy buffer. Verif.c does not consume any NVM because all data structures are in RAM. RAM is used to allocate transient Java arrays and local C variables. IFCInstallerApplet does not use any transient array. The local variables of Verif.c consume less than 100 bytes.

RAM consumption

RAM is used to allocate transient Java arrays and local C variables. IFCInstallerApplet does not use any transient array. The local variables of Verif.c consume less than 100 bytes.

Overhead (space and time)

The space overhead includes the added space to the standard applet in order to express the claimed security policy. Typically, that corresponds to the TCF custom component. The size of this component is the same as that used for on-card security policy. *The time measurement is not done on the simulator due to licensing issue.*

Methods and tools to define the security policy

EVE-TCF defines a dedicated language for describing the security policy. This language is powerful and is very adequate to this task. The policy can be written in a text file and then given to the EVE-TCF converter that transforms it into the TCF component of the CAP file.

The use of this tool chain is easy (e.g. aliasing and full-qualified names are supported) and suitable for managing complex security policy.

Global policy and non-interference

The verification of **global policy** and **non-interference** is not implemented and we have evaluated them using the technical report D6.6 provided by the INRIA-Lille. We only concentrate on the on-card policy footprint and the space overhead on the CAP file. The other performance aspects are only available on an implementation.

Global policy verification

For the POPS use-case, the estimated overhead on the CAP file (i.e. the size of the custom component) is 285+106N where N=n(n-1)/8, n being the number of states of the automaton that defines the forbidden call sequences in the global policy. The technical report provides the different values of this size depending on the value of n (from 1 to 10). For the average-case (n=5), the space overhead is about 30% of the CAP file (603 bytes on 2000). The size of the on-card policy is 70+15N. For the average case, this footprint is around 10% of the CAP file (115bytes on 2000). The conclusion from this evaluation is that, the footprint and the space overhead are still reasonable in practice.

Non-interference verification

For the POPS use-case, the estimated overhead on the CAP file (i.e. the size of the custom component) is more than 200% (4488 bytes on 2000 and 2127 bytes on 1000).

The size of the on-card policy is 941+N (for loading both applets). In other words, the footprint is at least 30% of the CAP files (941 byte on 3000).

The conclusion from this evaluation is that, the footprint and in particular the space overhead does not advocate the practical use of the non-interference verification for smart cards.

4.3.3 Conclusion on the Verification

The verification artefacts provided by the project to face with the problematic of preserving the security in case of software change in two ways, the off card verifiers like verifast, that requires the source code of the applet, could be deployed to the services providers (application developer) to check that their application (or new version) before its deployment to the card issuers databases. When the source code is not available, e.g. the card issuer receives only the cap files of the applications from the service providers, the on board checkers deployed into the cards will allow us to verify the compliance of the application with the security policy of the family of targeted cards.

4.3.3.1 Off-card Verification

The approach of using the source code for the verification of security properties is probably the most promising Formal methods techniques to be used in the mobile applications. This is due to the fact that it could be fully integrated in the process of the secure code development (without an additional step of producing a formal model of the specification). In this process, a significant amount of time is dedicated to the validation, using code review and security audit.

The security experts are interested by such a security validation tool for two main reasons: they can adapt the rules to be checked according to their expertise, state of the art, or the policy of the card issuers (their customers), secondly they can re-validate the application with respect to a **change**, an additional property to be checked or an update of the application. With the verifast tool, the speed of the verification is an appreciable advantage, but the feedback to the user in case of error must be improved, e.g. with more information provided. Useful information allowing the improvement of the code is a valuable feature because the step of debugging the code is crucial.

With respect to the common criteria certification, the off-card verification of the security properties of an applet could be used for the EAL7 evaluation (highest level requiring formal modelling and verification). This fits in the research activities on the benefits of static analysis tools to provide evidences for the evaluation. For example, the set of annotations that constitutes the formal model of the application may be the formal model of the design of the application (e.g. ADV_TDS). In that case, the informal correspondence that must be provided between the formal model and the code source (JC in our case) will be provided implicitly by the tool like Verifast. Another potential use of this kind of tool is its use for the validation of “basic” applets (applet that do not require security certification) before being allowed to be loaded on a certified product.

4.3.3.2 On-device Verification

The performance evaluation results show the two verification tools SxC (for verifying the interaction between applications) and EVE-TCF (for verifying transitive information flow) can be both embedded inside the conventional smart cards. Indeed, the persistent footprint of each tool is roughly 6KB plus a small amount of memory reserved for storing security policy. This footprint is relevant with respect to some 256KB of persistent memory of current smart cards. For RAM consumption, each verification does not use more than 100 bytes. This is compared favourably to roughly 5KB of RAM contained in the current smart cards. Moreover, one may still optimize the implementation by using a pre-defined RAM buffer (255 bytes) provided by the platform and hence avoid using any additional non-volatile memory. The overhead in terms of applet loading time (15%) can also be optimized using some specific design features of the platform.

The SecureChange project demonstrates once more the feasibility of the on-board verification, considered as the most challenging technology for resources constrained devices like smart cards. In the context of open cards that must accept the loading of any applications in the field (post issuance), the ability to perform the checks on-board is a market facilitator.

The solutions proposed by the project are very promising for treating safely the change due to the ability to custom the security policy with respect to the targeted cards. For example, besides the firewall security policy that ensures the isolation properties at runtime, we may want to refine the isolation policy with respect to the card issuer security requirements, with respect to its contracts with the service providers.

EVE-TCF technology for transitive information flow verification may be already used as the validation results provides reasonable figures about footprint and overhead while being usable without specific expertise. This technology allowing the verification of illegal access to the services is very interesting in the context of open cards that will host different applications from different sectors.

For the non-interference and the global policy properties, that are more complex by nature, the technology is not suitable for devices like smart cards with strong constraints on the size. But it is a powerful technology that could be used for larger devices such as mobile phone, for which we will have the same requirements of applications loading in the field with the strong security constraints.

With respect to Common Criteria evaluation of a card integrating the on-device verification, those on-board checkers will be evaluated as part of the product. Therefore this technology is not here to facilitate the Common Criteria evaluation itself.

4.4 WP4 Model Design

4.4.1 UMLchange

VALIDATION SCENARIOS and EXERCISES

The scenario concerns the platform developer that will use a security mean allowing the formalization of a **change** due to an **evolution of the specifications** of the GlobalPlatform component (card Manager) of the card. More precisely, the platform developer (or the security engineer), during the design phase, wants to check whether the new implementation still verifies the security properties. For that, he will use a model of the behaviour of the card manager according to Globalplatform specifications. The feasibility study first concerned the use of UMLsec for the verification of the confidentiality of transmitted data in a communication protocol. The conclusion for this exercise was that the use of UMLsec is efficient for specific but small specifications (cf. D1.2). The second exercise concerns the modelling of a subset of GlobalPlatform that allow focusing on security properties about the card life-cycle, and more precisely **life-cycle consistency**. For that, we evaluated UMLchange, a tool for modelling changes in UML notations. UMLchange works on the top of several UML editors. The version that we used during the evaluation is TOPCASED. Basically, UMLchange allows a user to define the change constructors as UML stereotypes. UMLchange also provides proof tools (as plug-ins) to ensure that any change respects the original model security policies. The validation will concentrate on the use of the semi-formal language based on UML, with the hypothesis that the security engineer has a basic knowledge and expertise using this language.

VALIDATION CRITERIA

The validation criteria that will be used are:

- Usability of the UMLseCh methodology (stereotype).
- Scalability.
- Ability to express security protocol elements and properties.

VALIDATION RESULTS

Usability

UMLchange is built upon well-known UML GUIs such as TOPCASED: this feature allows the UML engineers to get acquainted very easily with the tool. The proof of SecureChange is completely automatic using the plug-ins: this is a valuable feature for UML engineers. However, several improvements can be done on the interface of the tool:

- The grammar for describing the changes is error-prone: a context-sensitive assistance would be very helpful here to avoid issues due to typos.
- The error messages are not really helpful for identifying the issues: these messages should be more informative.

An experienced user may also write a new proof tool (plug-in) to prove the security of the changes. These plug-ins are developed in Java and are necessary to get confidence on the changes, but they have not been evaluated during this exercise. Although we do not have information on the difficulty of the plug-in development, let us note that this task can be prohibitive for industrial projects because UML engineers may not necessarily have the required expertise. Therefore, a framework for plug-ins development may be helpful. Otherwise, we need a large library of pre-defined plug-ins.

Scalability

Built upon TOPCASED, UMLchange can face with industrial-size modelling projects. Extensibility is definitely an advantage of UMLchange because new stereotypes can be defined to express various kinds of changes. Pre-defined stereotypes can also be provided for a specific domain, like the stereotypes defined for Global Platform's card life-cycle management to express changes resulting from the evolution of GlobalPlatform specification.

From a security point of view, a question is raised here: to which extent we trust these plug-ins? What happen if there are bug? A solution may be that these plug-ins generate a proof trace that will be certified by an external prover.

Ability to express GP security elements and properties

The evaluation has been concentrated on the **card life-cycle** management that corresponds to concrete and **local security properties**. It appears that most of the classic security policies defined using UML, can be easily handled by UMLchange. The modelling of more high-level and generic properties such as the integrity (non-interference) property of the security domains in the UICC configuration seems to be more complex. From a security evaluation point of view (e.g. Common Criteria), it is required that any changes is traced during the product life-time (change request, request accepted, change specified, implemented, change verified/validated, etc). Also change should be specified and validated by different people. These features are particularly mandatory if UMLchange is used for a Common Criteria certified products.

4.5 WP7 Testing

4.5.1 Model-based Testing Tool

VALIDATION SCENARIOS and EXERCISES

The artefact provided by the WP7 is a model-based testing tool that allows facing with the **specification evolution** as a change requirement. This change requirement is the most common use case for change for the platform part of the embedded software of o a smart card. The scenario consists in taking the role of a validation engineer that wants to minimize the validation activity of a new version of the software due to a change of specification. A set of security properties have been validating model-based generated test suites on a specific version of the platform (implementing the GlobalPlatform 2.1 specifications). The objective is to re-evaluate these properties on the next version of the implementation (implementing the GlobalPlatform 2.2 specifications). The target **security properties** are related to the **card lifecycle** consistency, and to the hierarchies of Security domains:

- Applet and card life cycle:
    - Whenever the card is in the TERMINATED state, it should not be possible to revert to another state.
    - It should not be possible for an application that does not have the Card Terminate privilege to switch the card lifecycle state to TERMINATED.
- The consistency of the Security domains hierarchy with respect to the privileges: the properties are related to the Authorized Management (AM) privilege of Security Domains (SD): Ensure that for any execution, it can never happen that two (or more) SDS with AM are on the same branch of the hierarchy.
- Properties related to the secure channel capabilities of the SD: ensuring that whenever a SD is moved across the hierarchy, the relevant authentications and accesses to secure channels are cleaned accordingly.

The tool is built upon a software component called EvoTest. EvoTest leverages upon Smartesting's model-based testing tool TestDesigner by extending it with three SecureChange software components:

- A security testing component: the schema-based test generator.
- A component for selective test generation method: SetGAM.
- A component for publishing the evolving test results: SmartPublisher.

For the purpose of the evaluation exercise these components have been used in conjunction with a specific UML/OCL model for Global Platform 2.2, and a model adapted for Global Platform 2.1.

VALIDATION CRITERIA

The validation criteria, according to the feasibility studies conducted against the usability, scalability and usability and discussed in the D1.2, will concern the model, the schema-based test generator, and eventually the SetGAM + SmartPublisher package. The validation criteria are then organized around:

- Testing Model.
- Testing component.
- Evolution Testing component.

VALIDATION RESULTS

Test model

For the evaluation exercise, two models are used:

- A full model for Global Platform 2.2.
- A model for Global Platform 2.1, that essentially differs from the first one on the life-cycle management part.

These models are defined using UML for the static part (description of the global state of Global Platform), and OCL for the dynamic part (corresponding to the legal transitions induced by the execution of Global Platform commands). These models are built inside IBM RSA (Rational Software Architect), extended with custom extensions provided by Smartesting for the edition of OCL pre and post-conditions and export of the model to the test generation engine. The use of this software suite is mandatory for examining the model's internals and usage within the EvoTest environment. Each Global Platform command is modelled as an UML operation attached to the Card class, and equipped with a pre and a post condition. The model is to be used for generating tests, so the semantics of these conditions is as follows: a sequence of alternating model states and model operations is called *a legal test case* for this model if and only if:

- the state is the initial state (as specified in the model),
- for each integer :
  - the pre-condition for operation is true in state ,
  - the post-condition for operation is true in state .

Each branch of each condition may then be labelled with a specific comment (@REQ) that allows giving a name to the corresponding behaviour of the system. The role of the generation engine is then to generate a minimal set of legal test cases such that a given set of behaviours (specified a set of @REQ) is achieved. This set may then be used as a test suite for the validation of these behavioural requirements.

Scalability
The model in itself is quite large. Its aims at covering the whole Global Platform specification (up to some abstractions), and therefore the task of maintaining it across evolutions is not a negligible. Provided that one has the sufficient level of knowledge (evaluated in section Usability, Required Experience), maintaining the model across specification evolutions requires deep understanding of its internals. The time required obviously depends on the scope of the change (as it is detailed in section Usability, Level of details).

Usability

Level of details
The Global Platform model describes the Global Platform specification by specifying all Global Platform specific APDUs (up to the abstractions listed in section Relevance). Moreover, each operation is correctly documented, and relates to the APDU (or sequence of APDUs) to be sent to the card. In the same line, each abstract value's meaning is correctly documented, and their names are often self-explanatory. Therefore, the model exhibits the right level of detail for the generation of executable tests, using a suitable adaptation layer.

Required expertise
Maintaining the model across specification evolutions requires a good knowledge of the following:

- UML modelling, in particular class diagrams, object diagrams, state diagrams, for modifying static aspects of the model (i.e. the type of Global Platform states) and the initial state.

- OCL language in order to modify the dynamic aspects of the model (i.e. pre- and post-conditions attached to the operations).
- Understanding of the test generation process, in particular the semantics of the pre- and post-conditions.
- Some experience with the IBM RSA tool (based upon Eclipse) and the SmarTesting extensions (in particular the model animator).
- Understanding of the model's internals.

These results are to be mitigated with the impact of the evolution change on the model. Indeed, the model-based testing approach has the huge benefice of keeping the amount of modifications to the model roughly proportional to the amount of modifications in the specification, provided that the model is sound and readable. It is the case for this model, which is cleanly structured and reasonably well documented. In case of minor changes to the specification, make the corresponding modification of the model does not require expertise in the fields mentioned above, especially when the modification requires only rewriting a specific pre- or post-condition, for which the OCL code is already present. On the other hand, larger modifications, such as the ones that imply the modification of the class diagram or adding of a brand new operation requires a much deeper knowledge.

Relevance
The model of Global Platform 2.2 for SecureChange is a faithful representation of the whole specification, up to a certain number of abstractions:

- The cryptographic primitives are abstract, and defects in their use are simulated through specific parameters, allowing us to test for situations such as invalid key length, incorrect MAC, incorrect signature and so on.
- The application loading process is abstract, through the use of a one-shot "load" primitive. Issues with sequencing of load commands, incorrect DAP blocks, length of blocks, etc may be simulated using specific additional parameters.
- All kinds or array parsing processes are abstract, including:
    - TLV-encoded data fields.
    - AIDs validation.
    - Keys validation.

  This is done through the use of dedicated parameters for the various parts of the data fields and various parsing errors that may happen, and through the use of enumerated data types (for instance for the AIDs).

The Global Platform specification is written in such a way that it includes both low-level aspects and high-level aspects. For instance, the whole appendix E of the specification describes all basic cryptographic bricks for the secure communications (SCP'02), and is abstract in the model. This would make this model unsuitable for general security testing. Nevertheless, the model has the right level of detail for security testing in the scope that has been defined for the POPS case study. More precisely, all operations related to the query and modifications of the life-cycle state of the card are accurately modelled, and the level of detail is close enough to the specification. Moreover, all life-cycle related checks in all Global Platform commands are completely and correctly modelled.

Security testing component
The Schema-Based Test Generator (SBTG) extends the test-generation functionalities of TestDesigner by allowing a security expert to express security-related scenarios using the artefacts from the UML/OCL model. The SBTG consists in:

- An Eclipse-based test schema editor integrated with IBM RSA, which allows us to enter properties using the Schema-based language defined in deliverable 7.3.
- A modified test generation engine that takes into account the test schemas.

The main objective of the SBTG is to allow expressing properties that reflect real-life critical scenarios that may be lacking from a pure functional test suite. Using it, it becomes possible to reach a specific state of the system/model using a more intricate path than the shortest one. This is particularly useful for lifecycle-related tests, where security testing may involve trying to build up vulnerability by applying critical side-effects on the global state of the system.

Scalability
Since EvoTest is to be considered as a research prototype, this part of the evaluation is not relevant.

Usability
This criterion is used to evaluate the level of knowledge of the model's internals needed to express the properties. The schema language allows expressing security properties using universal quantifiers that allow guiding the generation process by iterating through the operations and behaviours defined in the model. The quantifiers may be nested, and the language allows to express that the call to an operation shall be repeated (at least once, or any number of times). The user is also given the ability to filter the generated tests by using OCL predicates inside the formulas of the language. The language itself is sound and well defined, and uses a very intuitive syntax. The expression of any arbitrary property nevertheless requires a good knowledge of the model in order to express the relevant OCL predicates. Deep knowledge of the OCL language itself is not required, since the predicates will consist of a simple equality used to check the value of a part of the model's state. Nevertheless, although the language itself is very easy to learn, it seems mandatory to at least understand the class diagram of the model to express non-trivial properties.

Relevance
The language seems expressive enough to accurately describe any interesting security test intention. It is pertinent in the context of life-cycle or card content management related security testing for Global Platform. The ability to query the model in its own language while quantifying over operations and iterating them properly fills the gap between pure functional testing and security testing.

Evolution testing component
This component is made of:

- The SeTGaM selective test generation tool, which allows using previous test suites to automatically classify the generated tests with respect to their state and relevance according to the specification changes.
- The SmartPublisher component, that allows to reflect these successive generations and store the evolving status of the tests in an incremental test repository.

Using SeTGaM, the generated tests (according to the schemas defined using SBTG) are classified in three test suites:

- Evolution test suite, that contain tests targeting new requirements that were absent from the former test model.
- Regression test suite, which contains tests that were not impacted by the evolution of the model.
- Stagnation test suite, which contains tests that are invalid with the new version of the model.

There is also a deletion test suite, whose purpose is to serve as a garbage can for the test engineer, who can tag tests in the stagnation test suite that should not be considered for future evolutions.

Usability

Level of expertise
There is no specific expertise required, provided that the model and the security properties are already defined. SeTGaM in itself is a push-button tool. It is nonetheless mandatory to understand precisely how the test suites are constructed, and therefore to have a complete understanding of the content of deliverable 7.3.

User-friendliness
The SeTGaM component is available in the IBM RSA tool as a frame that allows selecting two successive versions of a model, and launching the generation and classification process. The results are presented synthetically using two graphs, giving information about the volume of each test suite, and the number of tests for each status. It is also possible to get for each status the list of test having this status, and for each of these tests the sequence of operations to be executed. This graphical interface itself is kept very simple, and in this sense is very usable. It is nevertheless to be considered as a research prototype, and as such exhibits some minor problems:

- The need for a naming convention for the test suites could be relaxed by using specific interface elements.
- The global interactions between SeTGaM, SBTG and TestDesigner are not always clear; it would be useful to have more guidance on the global workflow for using the whole EvoTest component.

Relevance
The building of these test suites is based on the automatic computation of the status of a test, which has some relevance in an industrial context. A test may have the following status:

- New tests, corresponding to new behaviours introduced by the new version. Since each such test necessarily activates a portion of fresh new code, they are the ones to be tested first and more thoroughly.
- Outdated tests, corresponding to behaviours that have been crossed out of the new versions. These ones cannot, by definition, be reproduced on the new version of the product. They may be used in some case to ascertain that some explicitly removed

behaviour is completely unreachable, as is often the case for "maintenance updates" of a specification.

- Un-impacted and re-executable tests. While the distinction between these two statuses is quite subtle from our point of view, and not really relevant for our validation activity, these tests as a whole constitute the basis of the regression campaign.
- Updated and adapted tests, which correspond to behaviours existing in both versions of the specification, but have to be reached in a different way. The production of these tests is certainly the most time-consuming activity in manual testing.

Therefore, this component allows organizing the testing activity, by prioritizing the tests to be run. The automatic generation ensures behavioural coverage (so negative testing and bounds testing is properly achieved), and the automatic classification clearly defines the regression test suite. Moreover, this approach allows linking tests that have essentially the same objective, but may be expressed in completely different ways. It is a task that may not always be achieved by manual testing, but it is still extremely useful for the purpose of maintaining a constant quality of the test suite across time and evolutions.

4.5.2 Conclusion on the Model-Based Testing

Several experiences have been made using model-based testing for smart cards software. Generally, the major drawback highlighted by the validation teams is the time spent for modelling and the maintenance of the models in case of specification evolutions. Although the organization of the generated test suites and the traceability are appreciable, the validation engineer prefers modifying the tests suites that the model itself.

The SecureChange project confirms these results on the modelling effort but at the same time demonstrates that, in case of change, it is easier to report a minor modification on the models than investigating the test suites to identify the place for modification. This is an important feature for smart cards platforms. Generally, a smart card manufacturer develops and maintains few platforms (called baselines), traditionally one per market sector. Then several branches are developed corresponding to family of products. This means that the software that constitutes the platforms, such as Globalplatform implementation, will be concerned by specification evolutions or small modification for customization purpose. Therefore, the SecureChange model-based technology will be helpfully to report the changes on the models developed once that on the million of tests that are maintained in the tests benches. This is why the effort must continue to improve the usability of the modelling and we advocate that this technology must be planned and used early enough in the product life cycle. Although expertise in UML / OCL seems to be an important requirement, it is rapidly damped in time as with any programming language. Usually the R&D people are either already familiar with these languages, or have the scientific background needed to be quickly trained.

One of the main advantages of this technology is its use in the context of Common Criteria certification. Generally, if a product has been CC certified, any modification requires at least a “delta” certification. This requires to the developer providing the evaluator with evidences on the impact of the change and in particular, how to perform the testing on the modified product. It is clear that the SETGAM methodology and tool will facilitate this step with the categorization of the test suites and the corresponding reports.

5 VALIDATION CONCLUSIONS

This section provides some concluding validation remarks about the SecureChange solutions. The validation activities highlight that SecureChange results address to a certain extent the lack of support in engineering evolving systems and guaranteeing security properties. The three case studies highlighted how WP artefacts support industrial practices. Moreover, the validation activities allowed us to identify alternative usages for SecureChange solutions. Table 9 shows the validated WP artefacts by each case study. Previous sections report the validation results for each artefact. Overall, SecureChange artefacts provide suitable support to specific engineering activities that concern the modelling and verification of security features with respect to changes. The case studies and the conducted validation activities highlighted how the different artefacts support SecureChange objectives.

Table 9 WP Artefacts validated by case study

Case Study	WP	Artefact
ATM	WP2	Change Driven Security Engineering
		MoVE Tool
	WP3	SeCMER Modelling
		SeCMER Tool
	WP4	Integration of Design Modelling Solutions
	WP5	Risk Assessment Language and Methodology
		Risk Modelling Tool
HOMES	WP2	Security-As-A-Service (SeAAS)
		Change Patterns
	WP6	VeriFast
		Security-by-Contract (SxC)
	WP7	Telling Test Stories (TTS)
POPS	WP4	UMLchange
	WP6	Development-time Verification of JC Applets
		On-Device Verification
	WP7	Model-based Testing Tool

One of the drawbacks is the complexity of the models captured by different SecureChange solutions. It is critical therefore the tool support for each WP artefact. Hence, future work would need to improve the usability of such tools before deploying them into industrial domains. Obviously, training would play a critical role for such deployments. Another critical aspect is the integration of different models. The evaluation activities have shown an extent of integration between different artefacts. However, integration among and traceability across models would need to be supported carefully. Integration and traceability aspects play an important role in supporting an engineering process tailored to deal with changes and security feature, like the one supported by SecureChange. Finally, the validation activities highlighted how the different artefacts support and comply with industrial practices, hence how SecureChange fits current industrial requirements. However, domain experts found SecureChange artefacts of particular relevance for their daily work. However, it is still questionable whether or not SecureChange artefacts can be delivered in industrial contexts in their current versions. It has emerged that it would be necessary to tailor further SecureChange artefacts in order to customise them to specific application domains. However, validation activities highlighted alternative usages (e.g. systematic model-based support of requirements elicitation and risk analysis) identified by domain experts that were not envisaged initially. Domain experts pointed out how relevant activities (e.g. dealing with changes) receive little support by current engineering practices. Therefore, SecureChange artefacts provide various advancements with respect to current industrial practices. Moreover, they would provide valuable support to activities (e.g. brainstorming activities on changes) that are currently dealt unsystematically. In conclusion, the validation of SecureChange has effectively highlighted how delivered technical artefacts address the goal of *ensuring "lifelong" compliance to evolving security, privacy and dependability requirements for long-running evolving software systems*.

References

[1] SecureChange, D1.1 Description of the scenarios and their requirements, 2010.

[2] SecureChange, D1.1.1 Selected Change Requirements and Security Properties, 2010.

[3] SecureChange, D1.2 Applicability of SecureChange Technologies to the Scenarios, 2011.

[4] EUROCONTROL, European Operational Concept Validation Methodology, E-OCVM Version 3.0, Volume I, February 2010.

[5] SESAR JU, WP 14 - SWIM Technical Architecture, Description of Work (DoW), Version 4.0, 2008.

[6] M. S. Lund, B. Solhaug, K. Stølen: Model-Driven Risk Analysis – The CORAS Approach, Springer, 2011.

[7] International Organization for Standardization: ISO 31000 Risk Management – Principles and Guidelines, 2009.

[8] EUROCONTROL safety regulatory requirements (ESARR), ESARR 4 – risk assessment and mitigation in ATM, Edition 1.0, 2001.

[9] SecureChange, D.3.4 Proof-Of-Concept CASE Tool, 2012.

APPENDIX

A. Validation Process

Validation is a generic term has wide usage but with a diversity of interpretations. Even in one area, such as software engineering, it may have different meanings and characteristics. We adopted an Operational Validation, which encompasses aspects of Technical Validation with user-centred evaluation. In our view, Validation is the Process needed to demonstrate how a system, a methodology or a operational procedure can function in real life conditions with the required level of performances, security and operability. This involves checking that technological feasibility and target safety level, cost-efficiency, end-users acceptability are all achieved. The Operational Validation can also be defined as the process of answering the question: ***Are we building the right system?*** In addition, to the Technical Validation and Verification that can be taken as answering the question: ***Are we building the system right?*** We used a systematic and generic approach to Validation, by applying state-of-the-art validation methods, like the European Operational Concept Validation Methodology (E-OCVM) [4], that can be used for all the various contributions and results of the SecureChange project.

We validated the SecureChange artefacts by exposing them to a set of industrial case studies emerging from the three reference domains showing the security characteristics and complexity of the evolving infrastructures the SecureChange framework is expected to manage. We are thus providing a set of real world, industrial relevant prototypes and scenarios based on our application case studies and on the techniques and tools currently available in the project. The industrial evaluation has been used to assess the feasibility and effectiveness of the SecureChange solutions. Each Case Study plays a different role with respect to the validation of the SecureChange solutions. Meanwhile there are complementarities and interplays among them.

Each SecureChange outcome has been both technically and operationally validated, with respect to a particular application domain and security problem. Thus, the Definition of different scenarios within the three different Case Studies addressed the following points:

1. Happening in a real work setting (e.g. realistic procedures, realistic conditions) and proposing realistic situations clearly addressing evolutionary issues.
2. Covering main security, privacy and dependability problems emerged during the analysis phases.
3. Including main resources (Software, Hardware, Liveware) and describing their interaction highlighting criticalities.

Validation Methodology

It is always difficult to demonstrate that Validation objectives of a project are achieved, and for this reason the high level Validation objectives have to be broken down into detailed Validation criteria. The detailed Validation criteria have a direct influence on the more general Validation objectives and, being more detailed, are more easily measurable. This process of decomposition has to be repeated several times resulting in a hierarchical structure of objectives (Tree Model as shown in Figure 52).

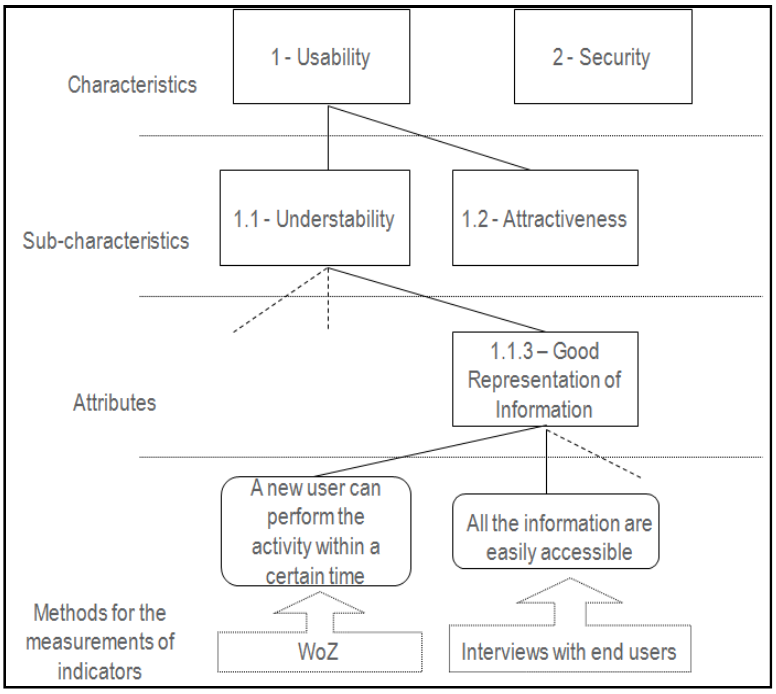

Characteristics
1 - Usability
2 - Security
Sub-characteristics
1.1 - Understability
1.2 - Attractiveness
Attributes
1.1.3 – Good Representation of Information
A new user can perform the activity within a certain time
All the information are easily accessible
Methods for the measurements of indicators
WoZ
Interviews with end users

Figure 52 Decomposition and identification of Validation Criteria

The decomposition of objectives ends with the identification of basic indicators, which represent the 'leaves' of the 'leaf' in the tree model. Note that indicators can be quite diverse. For instance, some indicators can be measurable. Whereas, other indicators might highlight compliance with standards or development processes, adoption of development tools and so on. Indicators will then require different types of evaluations. Figure 52 shows a simple example of how we can identify criteria and determine the trial, by decomposing iteratively the criteria in order to obtain evidences that can be measured or evaluated in a quantitative or qualitative way. Different types of methods can be used to support the Validation:

- **Deterministic**, e.g. formal proof of compliance to a specification, demonstration of Security and Dependability requirements, and so on.
- **Probabilistic**, e.g. quantitative statistical reasoning to establish a numerical level.
- **Qualitative**, e.g. compliance with rules that have an indirect link to the desired criteria (e.g. compliance with standards, staff skills and experience).

Note that the proposed evaluation and validation process is similar to other assessment processes. For instance, system assurance relies on the construction of safety cases for the judgment of the adequacy of system safety.

Indicators can provide information about the lower level of the detailed Validation objectives and they can be evaluated through measures taken during 'experiments' and trials carried out in different Validation Sessions. The evaluation of objectives at a lower level of the hierarchy should allow the evaluation of the objectives on the next level up of the hierarchy. An iterative approach to evaluation will, therefore, move up the hierarchy. In practice, all leaves of the tree can be measured and, therefore, assessed. Their assessment allows the assessment of the 'father', the assessment of the father and the other objectives at the same level allow the assessment of the 'grandfather' and so on.

The measurement of an element through its 'decomposition' into more measurable entities is a common approach in science, and a very similar approach has been successfully used, for example, in software engineering to measure the Quality of Software, or in the Air Traffic Management domain trough development of the European Operational Concept Validation Methodology (E-OCVM) or in Safety Assessment. The main steps in our iterative process, are drawn from the ones proposed in the E-OCVM Methodology, encompassing both operational Validation and technical Validation and Verification, can be summarized as:

1. **Set the evaluation strategy:**
   - Identify the user of the project outcome.
   - Identify the outcome usage and purpose.
   - Identify the general objectives of the Validation.
   - Identify what criteria are to be used.
2. **Determine the trial:**
   - Decompose the criteria iteratively, in order to obtain evidences.
   - Decide how they will be evaluated (e.g. measured and analysed).
   - Set out a plan of how the trial will be conducted.
3. **Conduct the trial:**
   - Go through the various evaluation methods (e.g. tests, formal verifications, simulations, application into case studies, user interviews, expert walkthrough).
4. **Determine the results:**
   - Assess the evaluation results (e.g. analysis of the measurements taken, expert judgements).

On the one hand, the Validation process support also the identification of the Maturity of the SecureChange outcomes and shows a body of evidence that relates to the overall project maturity with respect to the different Validation criteria identified [1][2][3]. On the other hand, scopes and objective of the Validation are likely to mature in line with the advancing maturity of the concept. As the concept becomes more mature, the Validation activity must become more rigorous and realistic. Validation Exercises may be larger and the scope and objectives of these exercises and their objectives becomes more complex and exhaustive.

Figure 53 shows the main phases (V0-V5) forming the E-OCVM and the implementation ones (V6-V7) for the validation and deployment of new operational concepts in ATM [4]. Even if the proposed approach is very general and can be effectively applied to different domains, the identification of the specific Validation Criteria and of the Methods to be used in their assessment, strongly depends on the nature of the particular results under evaluation.

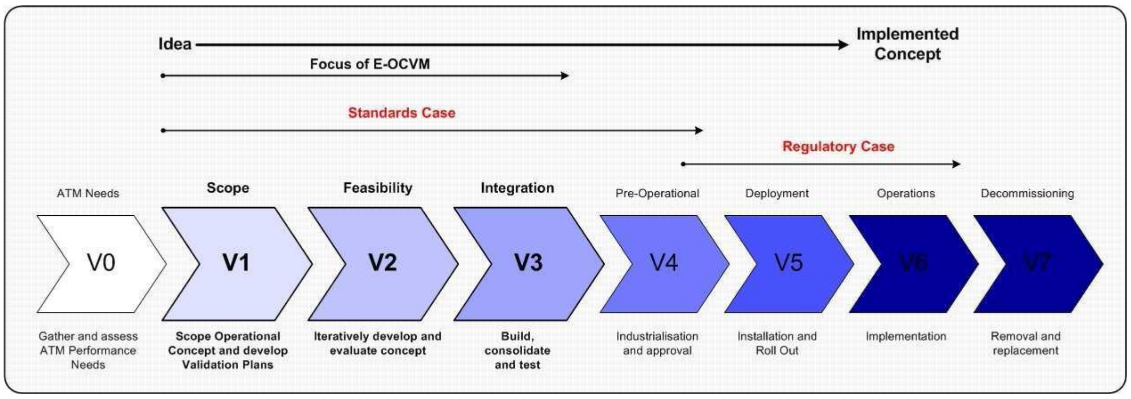

Idea
Implemented Concept
Focus of E-OCVM
Standards Case
Regulatory Case
ATM Needs
Scope
Feasibility
Integration
Pre-Operational
Deployment
Operations
Decommissioning
V0
V1
V2
V3
V4
V5
V6
V7
Gather and assess ATM Performance Needs
Scope Operational Concept and develop Validation Plans
Iteratively develop and evaluate concept
Build, consolidate and test
Industrialisation and approval
Installation and Roll Out
Implementation
Removal and replacement

Figure 53 E-OCVM Operational Concept Validation and Implementation

Data Collection and Analysis

Many different evaluation and assessment methodology can be used for technical validation and verification: intensive testing and quantitative simulations, Formal Verification, Expert Evaluation Techniques, Task Analysis and Direct Observation, Users Feedback Collection, System Data Collection. We hereby provide a small collection of the most common evaluation methods for industrial prototypes that can be implemented by means of scenarios:

Ethnographic approach / contextual enquiry

The ethnographic approach emphasises the understanding of behaviour in context through the participation of the investigator in the situation being studied as an active member of the team of users involved in the situation. It provides a descriptive report, utilising a range of approaches, mainly informal interviews and observational techniques. The ethnographic approach is essentially the traditional systems analysis approach enriched by contact with sociology and social anthropology.

Interviews

Interviews are commonplace techniques where domain experts are asked questions by an interviewer in order to gain domain knowledge. Interviewing is not as simple as it may appear and comes in 3 types: unstructured interviews, semi-structured interviews and structured interviews. The type, detail and validity of data gathered vary with the type of interview and the experience of the interviewer. Interviewing is still the most widely used method of finding out what users want.

Focus groups

A focus group brings together a cross-section of stakeholders in an informal discussion group format. Views are elicited by a facilitator on relevant topics. Meetings can be taped for later analysis. Focus group is useful early in requirements specification. It helps to identify issues which may need to be tackled and provides a multi-faceted perspective on them.

Wizard of Oz

This approach involves a user interacting with a computer system which is actually operated by a hidden developer - referred to as the ‘wizard’. The wizard processes input from a user and simulates system output. During this process the user is led to believe that they are interacting directly with the system. This form of prototyping is beneficial early on in the design cycle and provides a means of studying a user's expectations and requirements. The approach is particularly suited to exploring design possibilities in systems which are demanding to implement.

Rapid prototyping (software or hardware based)
This method is concerned with developing different proposed concepts through software or hardware prototypes, and evaluating them. In general the process is termed 'rapid' prototyping. The development of a simulation or prototype of the future system can be very helpful, allowing users to visualize the system, and provide feedback on it. Thus it can be used to clarify user requirements options. Rapid prototyping is described as a computer-based method which aims to reduce the iterative development cycle. Interactive prototypes are developed which can be quickly replaced or changed in line with design feedback. This feedback may be derived from colleagues or from the experiences of users as they work with the prototype to accomplish set tasks.

Storyboarding
Storyboards are sequences of images which demonstrate the relationship between individual screens and actions within a system. A typical storyboard will contain a number of images depicting features such as menus, dialogue boxes and windows. The formation of these screen representations into a sequence conveys further information regarding the structure, functionality and navigation options available within an intended system. The storyboard can be shown to colleagues in a design team as well as potential users, which allows others to visualise the composition and scope of an intended interface and offer critical feedback. This method can be used early in the design cycle where the use of storyboards supports the exploration of design possibilities and the early verification of user requirements.

Expert walkthrough
A walkthrough is a process of going step by step through a system design getting reactions from relevant staff, typically users or experts role-playing the part of users. Normally one or two members of the design team will guide the walkthrough, while one or more users will comment as the walkthrough proceeds. This technique is most often used where there is a relatively unstable prototype or a written procedural specification.

B. ATM Validation Plan

WP	Artifact	Contact	Version	Date of Availability	Starting Date	Action	Description
WP2	**Change driven security engineering**	UIB	n/a	n./a.	June 2011	Workshop with ATM Experts	Technical workshop to refine and complete the evaluation of WP2 process, already started in Y2 by means of direct application to the ATM Case Sudy.
	MoVE Tool support	UIB	v 0.9.x	31 March 2011	September 2011	Tool Live Demo during Workshop	Live and interactive Demo of the MoVE tool. Some simple modeling activities carried out by ATM experts with the support of WP2 technical partners. Direct observations by validation experts. Feedback collection trough questionnaires and semi-strucutred interviews.
WP3	**SeCMER conceptual model**	UTN	v 3.19	31 January 2011	April 2011	Methodology Evaluation	Application of the SeCMER conceptual Model to the ATM Case Study. Model review and refinement of the WP3 artefact to adapt to ATMs.
					June 2011	Workshop with ATM Experts	Technical workshop to refine and complete the evaluation of WP3 conceptual model.
	SeCMER case tool	UTN	v 2.0	9 May 2011	September 2011	ToolLive Demo during Workshop	Live and interactive Demo of the SeCMER case tool. Some simple modeling activities carried out by ATM experts with the support of WP3 technical partners. Direct observations by validation experts. Feedback collection trough questionnaires and semi-strucutred interviews for tool improvememtn and customisation.
			v 3.0	9 May 2011	September 2011	Workshop with ATM Experts	Live and interactive Demo of the final version of the SeCMER case tool.
WP4	**Prof of Concept Integration of design modelling solutions**	THA	D4.4 v1.0	31 January 2011	September 2011	Methodology Evaluation	Direct application of the Proof of Concept Integration of Modelling Solutions to the ATM Case Study. Evaluation of completeness, effectiveness, dmain suitability and user

							acceptability of the integrated modelling solutions.
			D4.4 v2.0	31 October 2011	September 2011	Workshop with ATM Experts	Technical and operational workshop to refine and complete the evaluation of D4.4 integrated modelling solutions.
WP5	**Risk Assessment Method**	SINTEF	n./a.	31 January 2011	June 2011	Methodology Evaluation by Expert Walktrough	Workshop with DBL Safety and Security ATM Experts to present, analyse and review the WP5 Risk Assessment Methodology. Possible foreseen exploitation in the SESAR Programme.
					December 2011	Questionnaires	Preparation and distribution to a wider audience of ATM stakeholders of a SecureChange Risk Assessment Methodology and Tool description in order to collect more feedback about the applicability and effectiveness of WP5 arteacts in the ATM domain
	Risk Modeling Language	SINTEF	n./a.	31 January 2011	June 2011	Workshop with ATM Experts	Technical and operational workshop to evaluate the completeness, expressibility and flexibility of the Risk Modelling Language.
	Prototype Risk Modeling Tool	SINTEF	D5.4	31 January 2011	September 2011	Tool Live Demo during Workshop	Live and interactive Demo of the WP5 Prototypee tool. Some simple modeling activities carried out by ATM experts with the support of WP5 technical partners. Direct observations by validation experts. Feedback collection trough questionnaires and semi-strucutred interviews to refine and improve the tool.
			D5.5	31 january 2012 (beta version)	September 2011	Tool Live Demo during Workshop	Live and interactive Demo of the final version of the WP5 Prototype tool case tool.

C. SeCMER Conceptual Model

The SecMER conceptual model (Figure 54) identifies a set of core concepts and relationships.

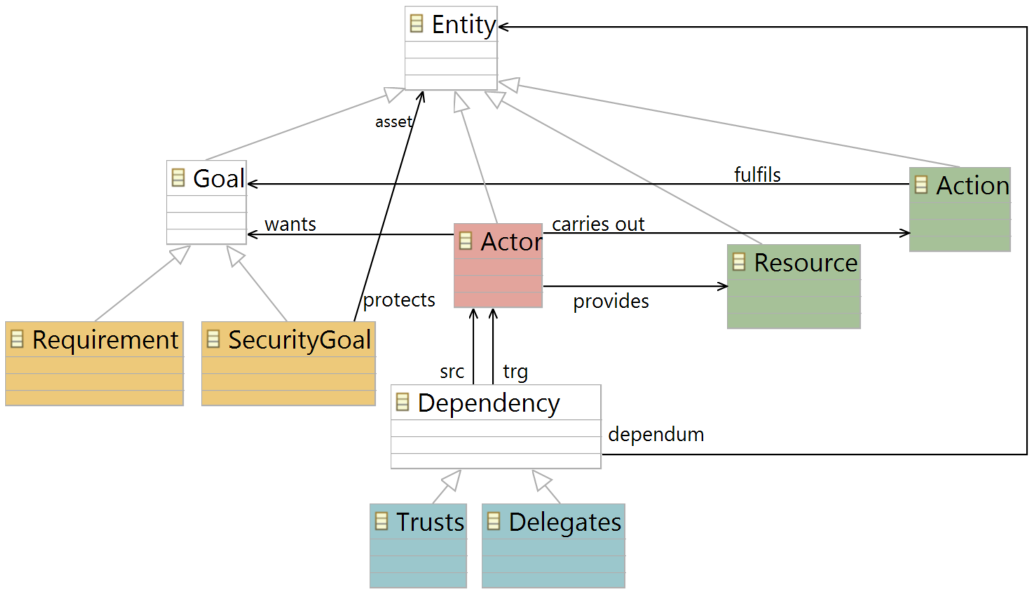


Figure 54 SeCMER conceptual model

- **Actor:** an entity that can act and intend to want or desire.
- **Action:** an entity performed by an actor, which can generate events, and can have preconditions and post-conditions.
- **Resource:** an entity without intention or behaviour.
- **Asset:** an entity of value that needs to be protected.
- **Goal:** a proposition an actor wants to make true.
- **Requirement:** is a refinement of a goal.
- **Security Goal:** a goal which prevents harm to an asset.
- **Trust:** is a relationship between two actors over a dependum. It specifies the belief of the trustor that the trustee won't misuse the dependum.
- **Delegation:** is a relationship between two actors over a dependum. It specifies the passage of responsibilities between two actors.
- **Protects:** is a relationship between a security goal and asset. It specifies that a security property needs to be satisfied for the specific asset.
- **Consumes:** is a relationship from an action to a resource denoting that the process consumes the resource.
- **Produces:** is a relationship from an action to a resource denoting that the process generates the resource.

Since there is no visual syntax specific for SeCMER's concepts, we will use the SI* visual syntax to illustrate the instantiation of the concepts. Table 10 shows the mapping between SeCMER's concepts and SI* concepts while Figure 55 shows the graphical notation for the SI* concepts.

Table 10 Mapping of concepts between SeCMER and SI*

SecMER concepts and relationships	SI* concepts and relationships
Actor	Actor
Action	Task
Resource	Resource
Asset	Goal, Resource, Task
Goal	Goal
Security Goal	Soft Goal
Trust relationship	Trust relationship
Delegation relationship	Delegation relationship
Protects	----
Consumes	Means-end
Produces	Means-end

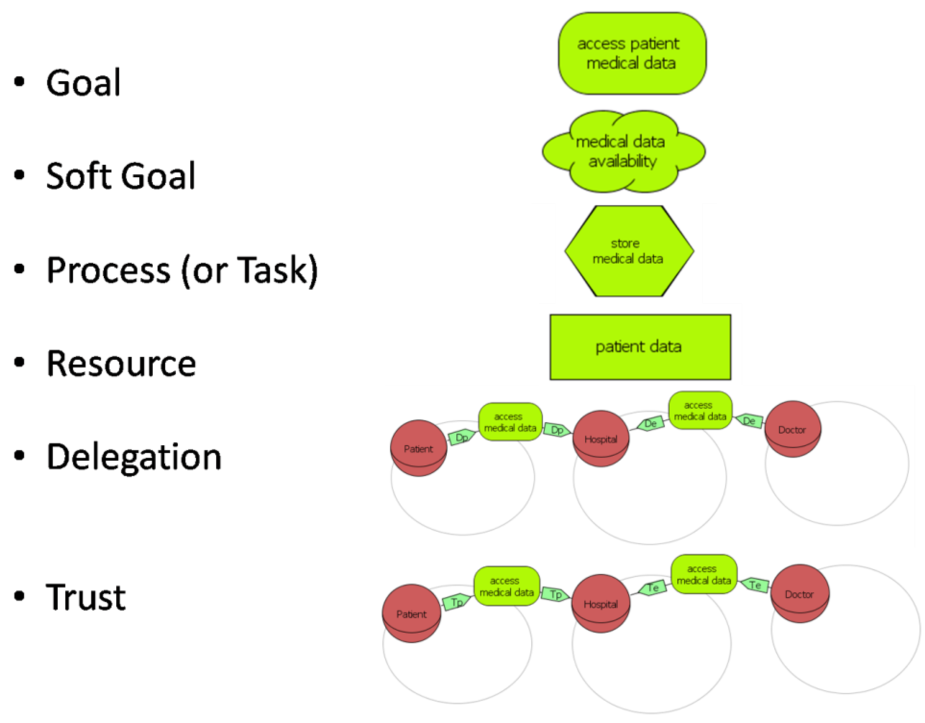


Figure 55 Graphical representation of SI* concepts

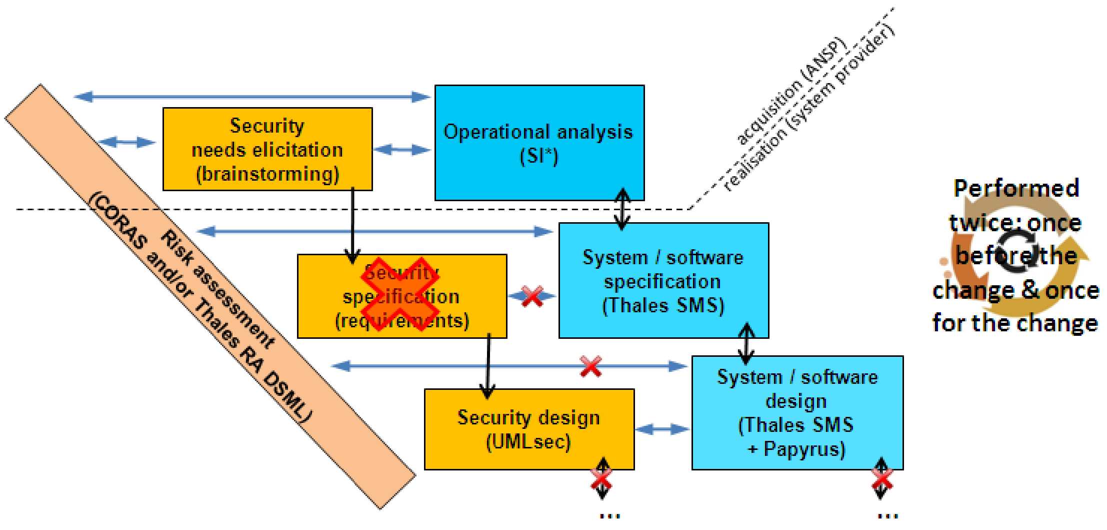
D. CORAS Definitions

- **Asset:** Something to which a party assigns value and hence for which the party requires protection.
- **Consequence:** The impact of an unwanted incident on an asset in terms of harm or reduced asset value.
- **Likelihood:** The frequency or probability of something to occur.
- **Party:** An organization, company, person, group or other body on whose behalf a risk analysis is conducted.
- **Risk:** The likelihood of an unwanted incident and its consequence for a specific asset.
- **Risk level:** The level or value of a risk as derived from its likelihood and consequence.
- **Threat:** A potential cause of an unwanted incident.
- **Treatment:** An appropriate measure to reduce risk level.
- **Unwanted incident:** An event that harms or reduces the value of an asset.
- **Vulnerability:** A weakness, flaw or deficiency that opens for, or may be exploited by, a threat to cause harm to or reduce the value of an asset.

E. ATM/WP4 Model Design

Case Study Walkthrough

This ATM case study walkthrough presents a number of modelling steps following the industrial version of the system engineering process, as described in D4.4b. This walkthrough allows for the validation of the methods and tools used at each step, but also, and most importantly, the links between the different engineering steps in terms of traceability, consistency and other relevant factors. This appendix shows some of the modelling steps of the ATM case-study walkthrough. The scope of the ATM case study walkthrough is pictured below (Figure 56).



Security needs elicitation (brainstorming)
Operational analysis (SI*)
acquisition (ANSP)
realisation (system provider)
Risk assessment (CORAS and/or Thales RA DSML)
Security specification (requirements)
System / software specification (Thales SMS)
Performed twice: once before the change & once for the change
Security design (UMLsec)
System / software design (Thales SMS + Papyrus)
...
...

Figure 56 Scope of the ATM case-study walkthrough

The following text explains the steps and then provides the schedule.

The 1st step is the operational analysis of the system "as is", which is normally done by the Air Navigation Service Provider, taking into account all constraints imposed by the regulator. At this level, the description deals with goals, missions and/or capacities, and resources. For SecureChange, we have used Si*.

The 2nd step is the security need elicitation for the system "as is". This step uses the modelling result of step 1 as input. There is no tool in SecureChange that covers this. It was therefore performed through brainstorming sessions between DBL, THA and SINTEF.

The 3rd step is the risk assessment performed at the operational level, for the system "as is". This step uses the results of steps 1 and 2 as input. Tool support was provided by CORAS.

The 4th step is the system / software specification by the industrial system provider contracted by the ANSP. This specification should relate only to the part of the system that is being contracted. It therefore takes as input a small part of the result of step 1. In our case, the scope of the contracted system was defined as a complete ATC centre, without

AMAN. Tooling was provided by the Thales SOA Modelling Suite (SMS) because the latter is now the modelling environment with which the Thales risk assessment DSML is integrated.

The 5th step is the security risk assessment performed at the system (meaning equipment) level by the industrial system provider. This risk assessment takes as input the results of step 4 (i.e. the description of the equipment in terms of process, components and services) and of step 3 (i.e. the high-level security requirements imposed by the ANSP on the subcontracted system).

We have decided to skip step n°6, which would have been the engineering of the solutions to cope with the identified risks. The main reasons for this decision are that SecureChange does not provide any specific technology to cope with that, and also because this step can be performed by iterating on step 4.

The 7th step is the detailed design. In our case-study, we focussed on a very "small" part of the ATC centre. The main objective here was to feed the formal verification of the security properties of step n°8. Therefore, the compulsory choice is a detailed design model using UML / EMF, namely Papyrus. Again, modelling here was kept minimal because the SecureChange project does not provide any specific technology to cope with step.

The 8th step is the formal verification of the security properties using UMLsec. This step takes as input the results of step n°7.

Finally, we have decided not to perform step 9, which would have been the risk assessment at the detailed design level. The main reasons for this decision are that the UML model produced in step n°7 will be simplistic (only adapted by UMLsec evaluation) and that we did not expect any additional lessons learnt with respect to the risk assessment step already performed in steps n°3 and 5.

These 9 steps close the 1st iteration, describing the system before the change. For the 2nd iteration, we performed all the 9 steps again, introducing the AMAN change at each step. Again steps 6 and 9 were skipped.

The table below recalls the work share, specifying the main tools used, the responsible partner and the approximate dates of provision, considering that the responsible actor was responsible for both iterations, once for the system "as is" without the AMAN, and once with the system "to be" with the AMAN.

Iteration	Step n°	Step name	Responsible actor	Tool	Type (sub-steps)	Actors	Provision date
1	1	Operational analysis	DBL	Si*	Offline modelling	DBL	June 2011
					Model consolidation (by TelCo)	DBL + UNITN + THA	July 2011
					Model upgrade & validation	DBL + UNITN + THA	July 2011
1	2	Security need elicitation	DBL + THA	Brainstorming	Offline modelling	DBL	July 2011
					Model consolidation (by TelCo)	DBL + THA + UNITN	July 2011
					Model validation	Security expert (THA or	July 2011

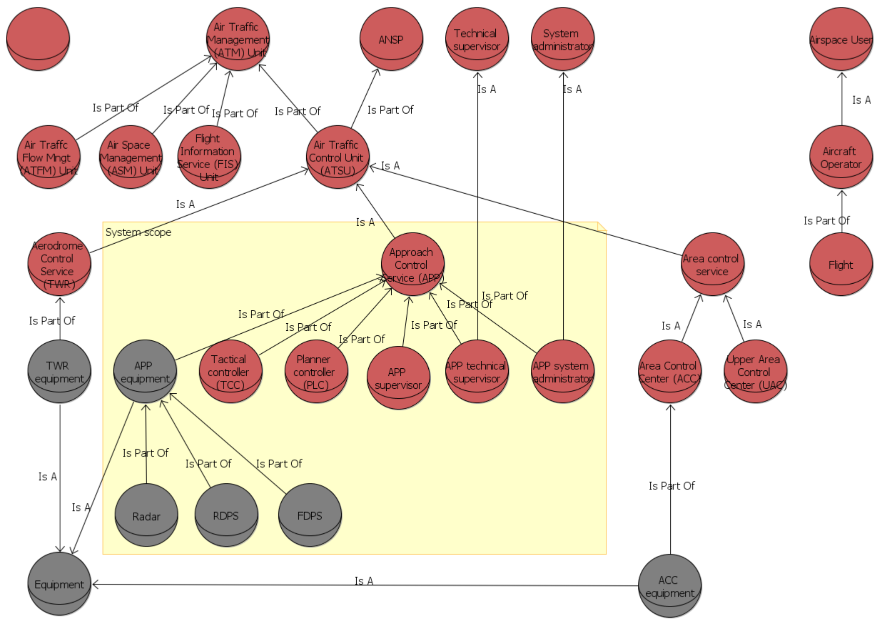
Iteration	Step n°	Step name	Responsible actor	Tool	Type (sub-steps)	Actors	Provision date
						DBL?)	
1	3	Risk assessment performed at the operational level	SIN	CORAS	Offline modelling	SIN	July 2011
					Model consolidation (by TelCo)	SIN + DBL	August 2011
					Model validation	Security expert (DBL + THA)	Septembre 2011
1	4	System/software specification	THA	Thales SMS	Offline modelling	THA	August 2011
					Model consolidation (by TelCo)	THA	Septembre 2011
					Model validation	DBL	Octobre 2011
1	5	Risk assessment performed at the system/software architectural level	DBL + THA	Thales RA DSML	Offline modelling	THA	Septembre 2011
					Model consolidation (by TelCo)	THA+DBL	Octobre 2011
					Model validation	Security expert (SIN)	Octobre 2011
1	6	Security specification	-	-	-	-	-
1	7	System /software detailed design	THA	Papyrus	Offline modelling	THA	Septembre 2011
					Model consolidation (by TelCo)	THA+DBL	Octobre 2011
					Model validation	Archi. expert (THA)	Octobre 2011
1	8	Security design (formal verification)	THA	UMLsec	Offline modelling	THA	Septembre 2011
					Model consolidation (by TelCo)	THA + TUD	Septembre 2011
					Model validation	Archi. expert (THA)	Octobre 2011
1	9	Risk assessment performed at the detailed design level	-	-			
1	**-**	**Debriefing iteration 1**	**ALL**	**-**	**F2F meeting?**		Octobre 2011
2	1	Operational analysis	DBL	Si*	Offline modelling	DBL	17/10/11
					Model consolidation (by TelCo)	DBL + UNITN + THA	24/10/11
					Model validation	DBL (ENAV?)	26/10/11
2	2	Security need elicitation	DBL + THA	Aniketos STML	Offline modelling	DBL	24/10/11
					Model consolidation (by TelCo)	DBL + THA + UNITN	31/10/11
					Model validation	Security expert	02/11/11

Iteration	Step n°	Step name	Responsible actor	Tool	Type (sub-steps)	Actors	Provision date
						(THA or DBL?)	
2	3	Risk assessment performed at the operational level	SIN	CORAS	Offline modelling	SIN	31/10/11
					Model consolidation (by TelCo)	SIN + THA	07/11/11
					Model validation	Security expert (THA)	09/11/11
2	4	System /software specification	THA	Thales SMS	Offline modelling	THA	07/11/11
					Model consolidation (by TelCo)	THA+TUD	14/11/11
					Model validation	DBL	16/11/11
2	5	Risk assessment performed at the system /software architectural level	DBL + THA	Thales RA DSML	Offline modelling	THA	14/11/11
					Model consolidation (by TelCo)	THA+DBL	21/11/11
					Model validation	Security expert (SIN)	23/11/11
2	6	Security specification	-	-			21/11/11
2	7	System /software detailed design	THA	Papyrus	Offline modelling	THA	21/11/11
					Model consolidation (by TelCo)	THA+TUD	28/11/11
					Model validation	Archi. expert (THA)	30/11/11
2	8	Security design (formal verification)	TUD	UMLsec	Offline modelling	THA	28/11/11
					Model consolidation (by TelCo)	TUD+THA	05/12/11
					Model validation	Archi. expert (THA)	07/12/11
2	9	Risk assessment performed at the detailed design level	-	-			
2	**-**	**Debriefing iterations 1 & 2**	**ALL**	**-**			12/12/11

Iteration n°1: The system "as is"

*Step n°1: Operational analysis using Si**

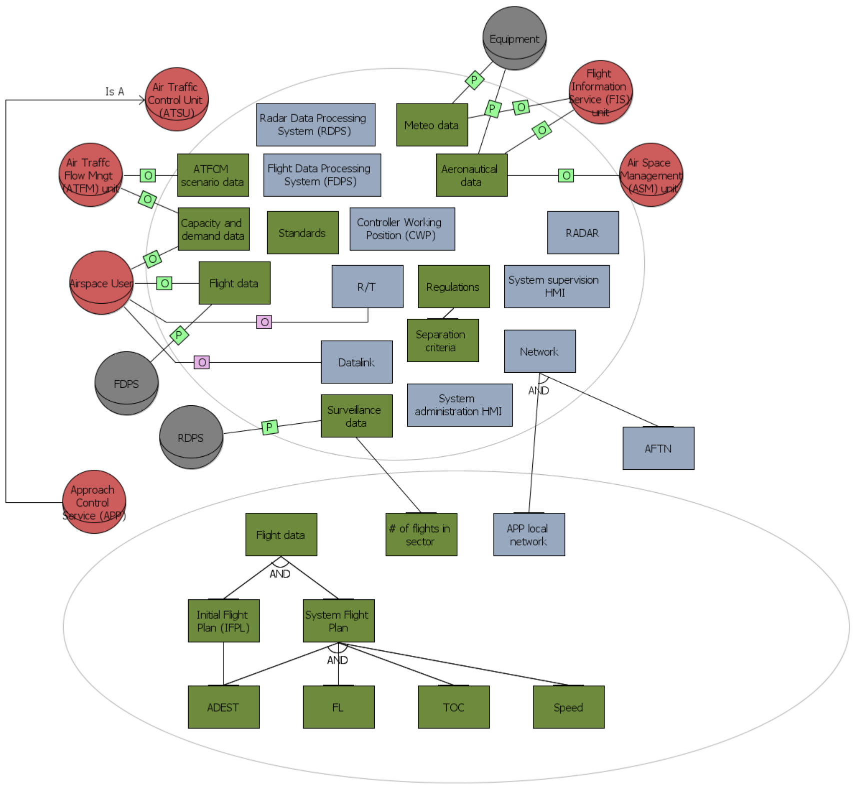
Below are presented a number of operational diagrams realised with Si* for the ATM case-study. Each of these diagrams has a specific focus: the actors (cf. Figure 57), the resources (cf. Figure 58), the overall ATM (cf. Figure 59), arrival sequencing (cf. Figure 60) and equipment (cf. Figure 61).



Air Traffic Management (ATM) Unit
ANSP
Technical supervisor
System administrator
Airspace User
Air Traffc Flow Mngt (ATFM) Unit
Air Space Management (ASM) Unit
Flight Information Service (FIS) Unit
Air Traffic Control Unit (ATSU)
Aircraft Operator
Flight
Aerodrome Control Service (TWR)
System scope
Approach Control Service (APP)
Area control service
TWR equipment
APP equipment
Tactical controller (TCC)
Planner controller (PLC)
APP supervisor
APP technical supervisor
APP system administrator
Area Control Center (ACC)
Upper Area Control Center (UAC)
Radar
RDPS
FDPS
Equipment
ACC equipment
Is Part Of
Is A

Figure 57 All the actors

Beyond presenting all the actors, Figure 57 also presents the scope of the contracted system, which will be designed during step n°4. It is to be noted that equipment is modelled as an actor when it realises some goals. Equipment is also modelled as a resource in terms of physical equipment. In Figure 58, both intangible resources and tangible resources are modelled.



Equipment
Is A
Air Traffic Control Unit (ATSU)
Radar Data Processing System (RDPS)
Meteo data
Flight Information Service (FIS) unit
Air Traffc Flow Mngt (ATFM) unit
ATFCM scenario data
Flight Data Processing System (FDPS)
Aeronautical data
Air Space Management (ASM) unit
Capacity and demand data
Standards
Controller Working Position (CWP)
RADAR
Airspace User
Flight data
R/T
Regulations
System supervision HMI
Separation criteria
Network
FDPS
Datalink
AND
RDPS
Surveillance data
System administration HMI
AFTN
Approach Control Service (APP)
Flight data
# of flights in sector
APP local network
Initial Flight Plan (IFPL)
System Flight Plan
ADEST
FL
TOC
Speed
P
O

Figure 58 All the resources

For the overall goal modelling, a number of "Golden Rules" were applied, of which:

- Golden Rule n°1: there should be no sub-goals without an upper-goal.
- Golden Rule n°2: goals are a state of affair, not activities; therefore goals are best expressed in the past tense. Nouns are used occasionally for continuous goals, i.e. goals that are never accomplished, e.g. ATC service.

Figure 59 was not meant to be "readable" in this document. It is provided to show the complexity of the case-study and the scalability capacities of Si*.

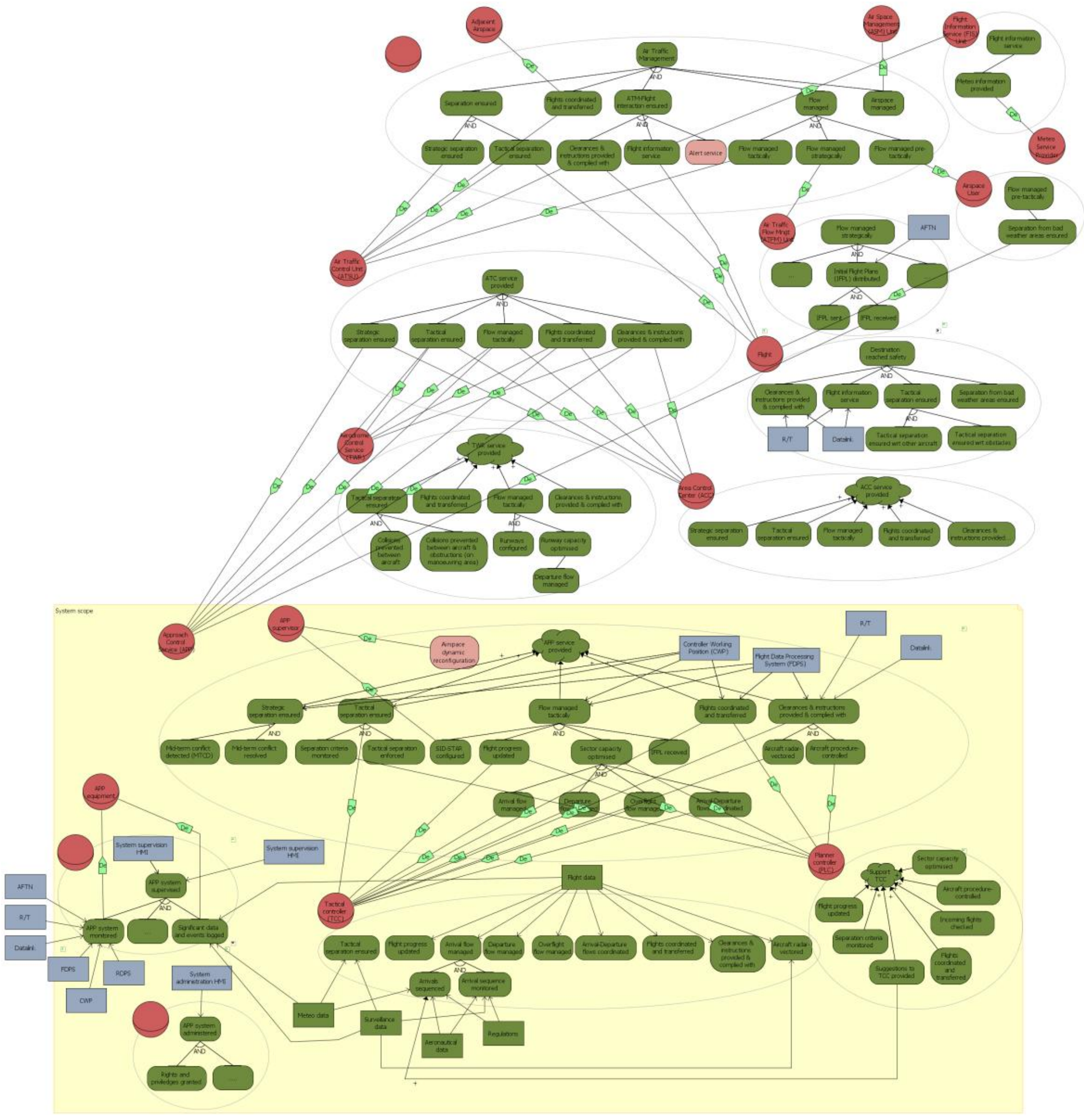


Figure 59 Overall ATM view

To enhance the readability of the goal decomposition, the arrival sequencing goal was decomposed in a separate diagram (Figure 60). The tool does not provide consistency assurance between the overall goal modelling diagram (Figure 59) and the focused arrival sequencing diagram (Figure 60).

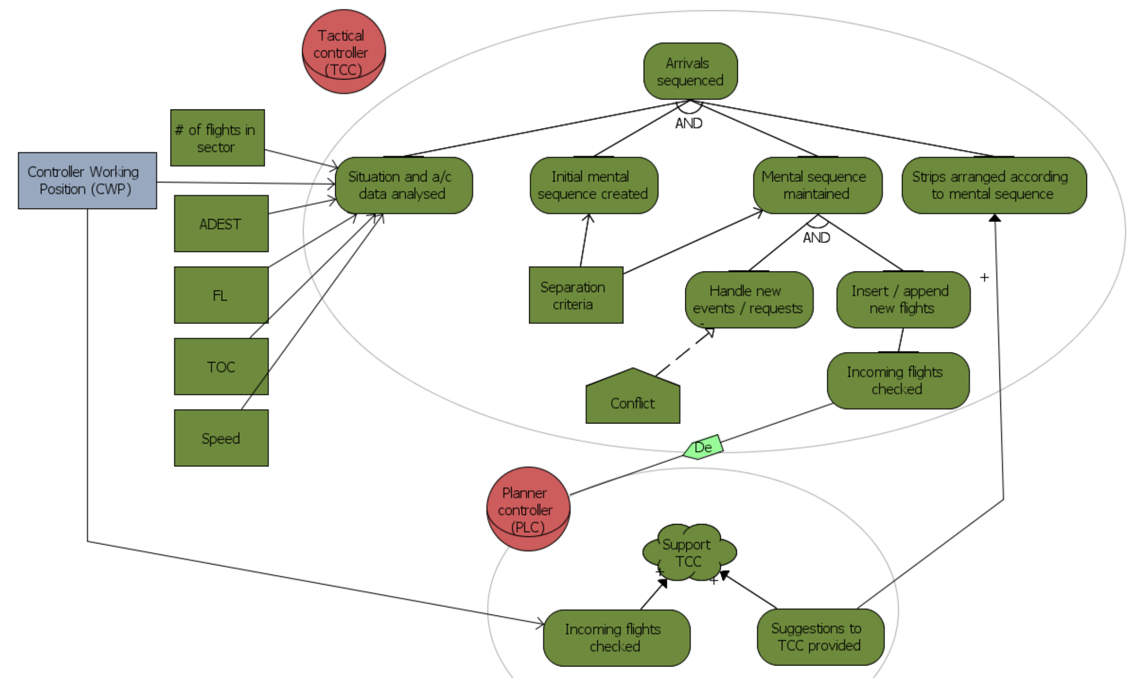

Tactical controller (TCC)
Arrivals sequenced
AND
# of flights in sector
Controller Working Position (CWP)
Situation and a/c data analysed
Initial mental sequence created
Mental sequence maintained
Strips arranged according to mental sequence
ADEST
FL
TOC
Speed
Separation criteria
AND
Handle new events / requests
Insert / append new flights
+
Conflict
Incoming flights checked
De
Planner controller (PLC)
Support TCC
Incoming flights checked
Suggestions to TCC provided

Figure 60 Focus on arrival sequencing

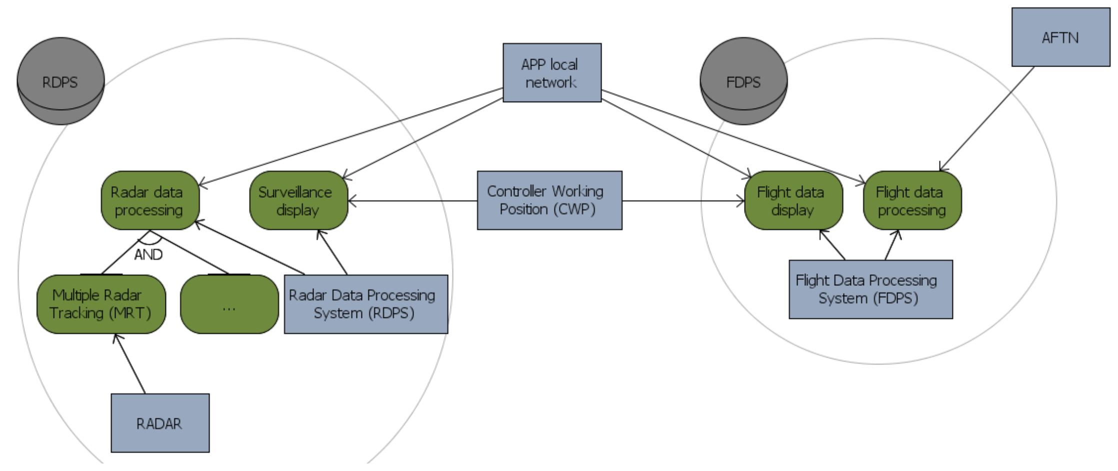

RDPS
APP local network
FDPS
AFTN
Radar data processing
Surveillance display
Controller Working Position (CWP)
Flight data display
Flight data processing
AND
Multiple Radar Tracking (MRT)
...
Radar Data Processing System (RDPS)
Flight Data Processing System (FDPS)
RADAR

Figure 61 Focus on equipment

Likewise, the diagram focused on the equipment enhances the readability of the model, but suffers from the absence of consistency assurance with the other diagrams. The ATM operational modelling was stopped when the model was deemed sufficient to perform the operational-level risk assessment.

Step n°2: Security needs elicitation

Following a number of discussions, inquiry and brainstorming sessions between the involved partners, the following 5 security needs were agreed upon:

- identification and authentication,
- need-to-know/ least-privilege,
- auditing support,

- robustness / checking of external messages,
- secured physical access to equipment, e.g. HMIs, cabinets.

Care was taken to focus only on security needs, not on safety. Each security need is further discussed below. Identification and authentication is enforced just by means of check points at the entrance of the buildings. There is no other kind of IT supported identification and authentication (e.g. no login at the controller working position).

Need-to-know/least-privilege implies having a fine understanding of the roles. On the operational side, the main roles are: Supervisor(s), Air Traffic Controllers (Planner and Tactical), and for the TMA: Coordinator. On the technical side, the main roles are: Technical Supervisor, System Administrator, and Technical Personnel. All the different roles have specific working positions, often located also in separated rooms; the Control Room is always separated from the Technical Room. Access control is mainly enforced by means of physical security (i.e. different locations, locked doors, guards).

Logging for auditing purposes is currently mainly focused on accidents and incidents, rather than on malicious external attacks (i.e. more on safety issues than on security threats). In particular, all the radio-frequency communications and all the CWP logs are recorded.

In the scope of this case-study, we tried to focus on auditing with respect to security issues. With respect to robustness, the communication (in/out) with the external world is mainly based on:

- phone 'point-to-point' communications,
- dedicated radio frequency communications,
- flight plan data; repetitive flight plans (RFPL) may represent a fallback solution in case of failure to comply with robustness,
- coordination data,
- receipt of flight surveillance data.

Finally, the secured physical access to equipment copes with the intrinsic geographically distributed nature of ATC systems. For example, one can stand near a radar without having access to surveillance data, so identification is not necessary, but protecting the radar from physical damage by a malicious actor or accidental event is still required.

Step n°3: Operational-level risk assessment using CORAS

This section presents the risk modelling resulting from the risk assessment that was conducted with respect to the Si* models and the identified security needs in the ATM case study. The Si* models are the output from the 1st step of the work plan, whereas the security needs were elicited during the 2nd step. The risk assessment was conducted as the 3rd step at the operational level, and the risk models were made using CORAS threat diagrams. The risk assessment scope is the APP service provisioning asset, which is a sub-part of the complete Si* models. Risk assessment includes risk identification, risk estimation and risk evaluation. The risk identification results are shown first.

The risk estimation involves the estimation of likelihoods and consequences for the identified unwanted incidents. The risk evaluation involves determining the risk levels based on the likelihood and consequence estimation, and comparing the results against the predefined risk evaluation criteria. Scales coming from a EUROCONTROL risk assessment document were reused in this case-study.

Risk identification

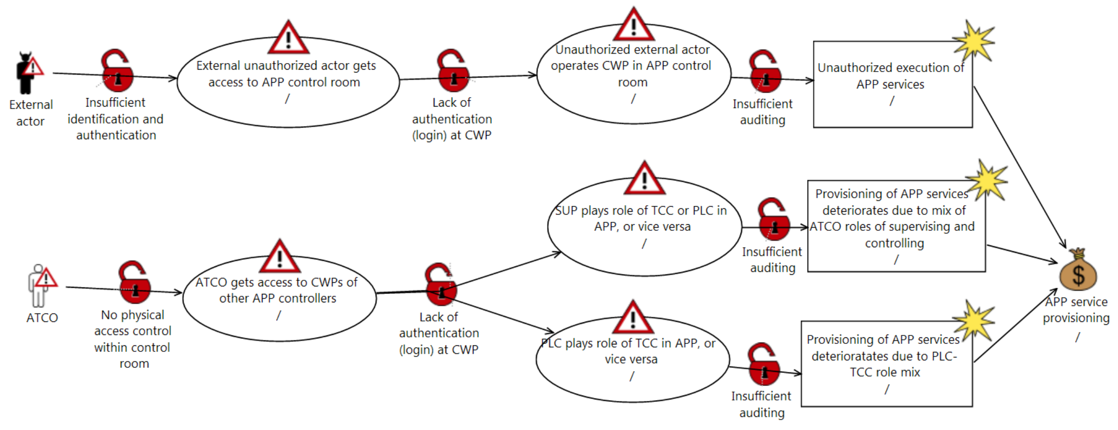

External actor
Insufficient identification and authentication
External unauthorized actor gets access to APP control room
Lack of authentication (login) at CWP
Unauthorized external actor operates CWP in APP control room
Insufficient auditing
Unauthorized execution of APP services
SUP plays role of TCC or PLC in APP, or vice versa
Insufficient auditing
Provisioning of APP services deteriorates due to mix of ATCO roles of supervising and controlling
ATCO
No physical access control within control room
ATCO gets access to CWPs of other APP controllers
Lack of authentication (login) at CWP
PLC plays role of TCC in APP, or vice versa
Insufficient auditing
Provisioning of APP services deterioratates due to PLC-TCC role mix
APP service provisioning

Figure 62 Identification and authentication, physical access and auditing

Figure 62 shows unwanted incidents related to insufficient identification and/or authentication in the APP control room. The diagram also addresses secured physical access. The potential security problem related to insufficient auditing is modelled by the vulnerabilities preceding the identified unwanted incidents. In this case, lack of auditing may contribute to the likelihood of the unwanted incidents, partly because actors may be reluctant to initiate incidents in case they know they are audited, and partly because lack of auditing makes it harder to identify adequate preventive means for incidents that reoccurs. Notice that *External actor* is modelled as a deliberate threat, i.e. someone with malicious intents. The *ATCO* is modelled as an accidental threat, i.e. it is assumed that he/she is not acting maliciously.

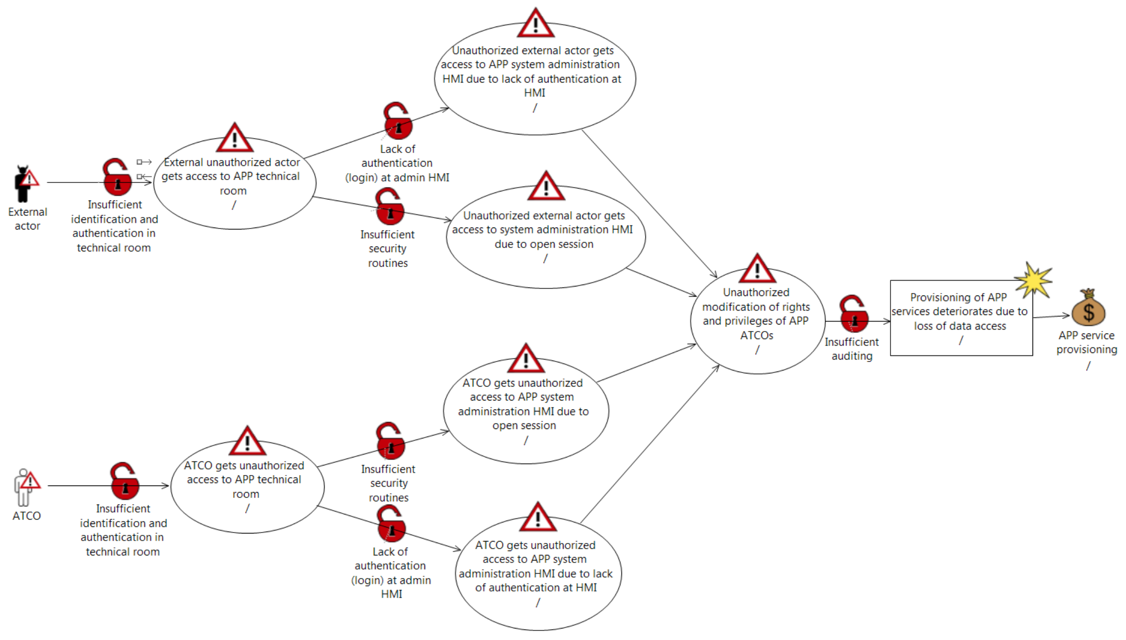

Unauthorized external actor gets access to APP system administration HMI due to lack of authentication at HMI
External actor
Insufficient identification and authentication in technical room
External unauthorized actor gets access to APP technical room
Lack of authentication (login) at admin HMI
Insufficient security routines
Unauthorized external actor gets access to system administration HMI due to open session
Unauthorized modification of rights and privileges of APP ATCOs
Insufficient auditing
Provisioning of APP services deteriorates due to loss of data access
APP service provisioning
ATCO gets unauthorized access to APP system administration HMI due to open session
ATCO
Insufficient identification and authentication in technical room
ATCO gets unauthorized access to APP technical room
Insufficient security routines
Lack of authentication (login) at admin HMI
ATCO gets unauthorized access to APP system administration HMI due to lack of authentication at HMI

Figure 63 Identification, authentication and auditing, technical room

Figure 63 shows an unwanted incident related to insufficient identification and/or authentication in the technical room. Notice that the threat *External actor* is modelled as deliberate, whereas the *ATCO* is not. It is hence assumed that the ATCO does not intend to act maliciously, but rather goes beyond his/her area of responsibility and authority. The potential problem of insufficient auditing is addressed similar to the previous threat diagram in Figure 62.

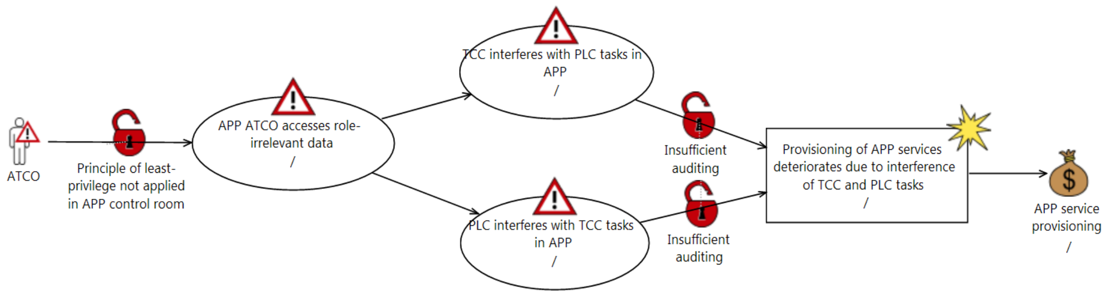

ATCO
Principle of least-privilege not applied in APP control room
APP ATCO accesses role-irrelevant data /
TCC interferes with PLC tasks in APP /
PLC interferes with TCC tasks in APP /
Insufficient auditing
Insufficient auditing
Provisioning of APP services deteriorates due to interference of TCC and PLC tasks /
APP service provisioning /

Figure 64 Least privilege and auditing

Figure 64 shows an unwanted incident related to lack of the principle of least privilege in the APP control room. Auditing is as before modelled as a vulnerability that may contribute to the likelihood of the identified unwanted incident.

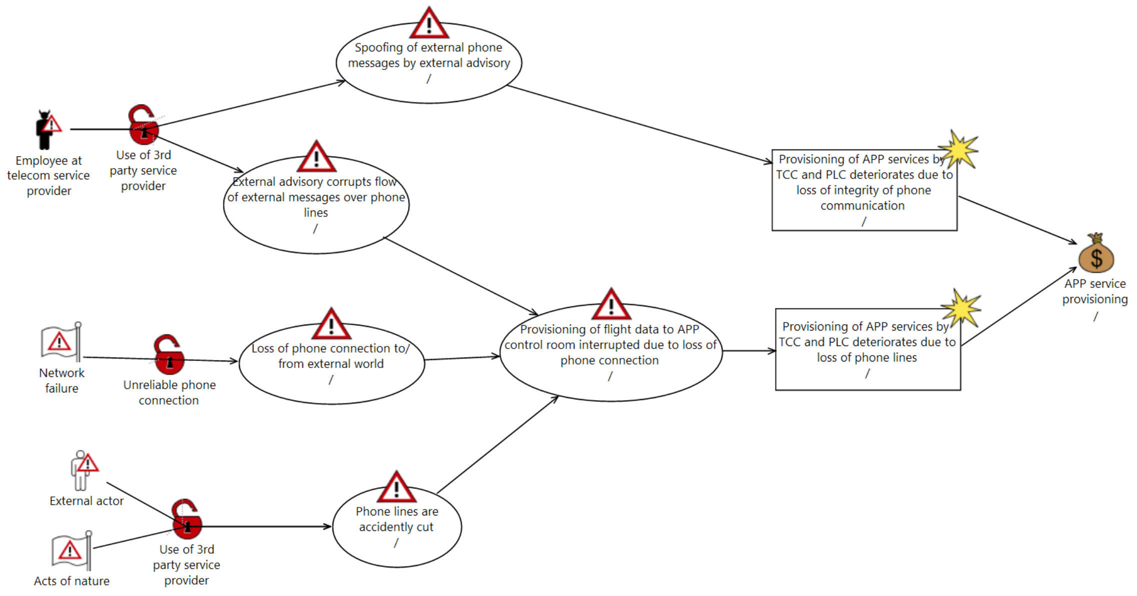
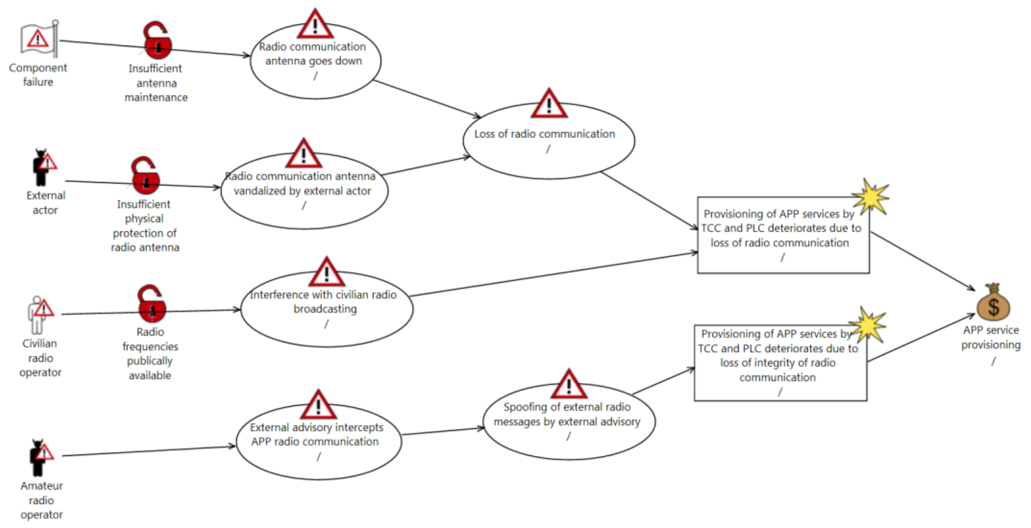

Spoofing of external phone messages by external advisory /
Employee at telecom service provider
Use of 3rd party service provider
External advisory corrupts flow of external messages over phone lines /
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of phone communication /
APP service provisioning /
Network failure
Unreliable phone connection
Loss of phone connection to/ from external world /
Provisioning of flight data to APP control room interrupted due to loss of phone connection /
Provisioning of APP services by TCC and PLC deteriorates due to loss of phone lines /
External actor
Acts of nature
Use of 3rd party service provider
Phone lines are accidently cut /

Figure 65 Robustness wrt external messages, phone lines

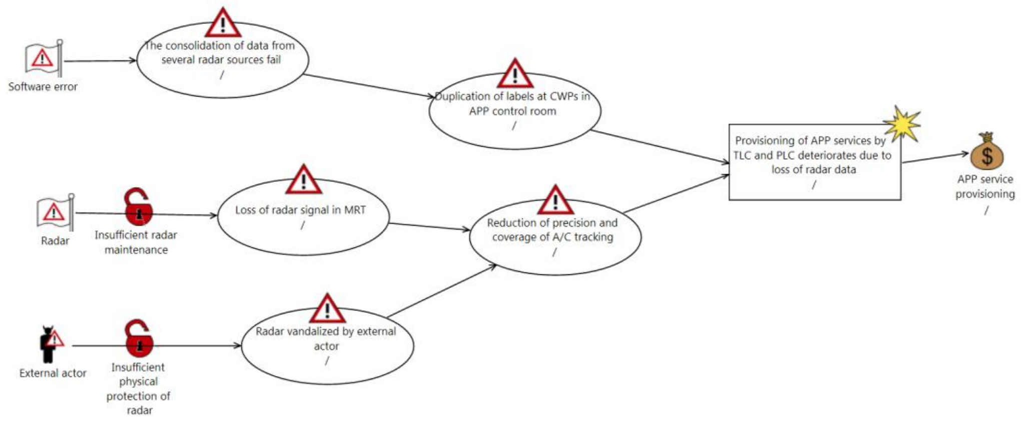
Figure 65 shows two unwanted incidents related to robustness, with respect to external messages transmitted by point-to-point phone lines. Secured physical access to equipment (phone lines) is also relevant.



Component failure
Insufficient antenna maintenance
Radio communication antenna goes down
/
Loss of radio communication
/
External actor
Insufficient physical protection of radio antenna
Radio communication antenna vandalized by external actor
/
Provisioning of APP services by TCC and PLC deteriorates due to loss of radio communication
/
Civilian radio operator
Radio frequencies publically available
Interference with civilian radio broadcasting
/
APP service provisioning
/
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of radio communication
/
Amateur radio operator
External advisory intercepts APP radio communication
/
Spoofing of external radio messages by external advisory
/

Figure 66 Robustness with respect to external messages, radio

Figure 66 shows two unwanted incidents related to robustness of external messages transmitted by radio. Secured physical access to equipment (radio antenna) is also addressed.



Software error
The consolidation of data from several radar sources fail
/
Duplication of labels at CWPs in APP control room
/
Provisioning of APP services by TLC and PLC deteriorates due to loss of radar data
/
APP service provisioning
/
Radar
Insufficient radar maintenance
Loss of radar signal in MRT
/
Reduction of precision and coverage of A/C tracking
/
External actor
Insufficient physical protection of radar
Radar vandalized by external actor
/

Figure 67 Robustness with respect to external messages, radar

Figure 67 shows an unwanted incident related to the robustness of external messages transmitted by radar. Secured physical access to equipment (radar) is also addressed.

Likelihoods, consequences and risk levels

Each pair of an unwanted incident threatening a supporting asset, and a feared event on a primary asset constitutes a **risk**. In order to determine the risk level, we need to determine the likelihood and consequence of each unwanted incident. **Likelihood** is the frequency or probability for something to occur, whereas a **consequence** is the impact of an unwanted incident on an asset in terms of harm or reduced asset value. The likelihood scale is based on EUROCONTROL documents, where likelihood is defined as follows: *The extent to which an event is expected in a given time scale. In security risk analysis this factor may be uncertain, and is often described in qualitative terms*. The likelihood scale of four levels is given in Table 11.

Table 11 Likelihood scale

Level	Description
1 Frequent	Several incidents per year. This includes very frequent incidents (e.g. Many per day).
2 Occasional	An Incident is likely in 10 years. At most a small number of incidents may be expected in any year.
3 Possible	Incidence cannot be estimated; however, it is realistic to anticipate the event.
4 Rare	The incident can be discounted.

The consequence scale is based on the same source, where impact (consequence) is defined as follows: *The unwanted consequence of a security incident; the impact may be qualified in financial, opportunity, efficiency, safety or any other relevant business or ATM operational terms*. The impact scale of five levels is given in Table 12 with their interpretation in safety, business and reputation.

Table 12 Impact (consequence scale)

	IMPACT		
Level	**Safety**	**Business**	**Reputation**
1 Very High	Large scale loss of life		
2 High	Significant risk of fatality	Long term business damage	Loss of Operating Licence
3 Medium	Increased safety risk	Significant business damage	Litigation or Criminal Conviction
4 Low	Short term safety risk	Significant loss	Public reputation damage
5 Insignificant	Insignificant	Insignificant	Insignificant

Notice, importantly, that the purpose of the description of the impact levels in terms of safety, business and reputation is to give the risk analysis participants reference points so as to understand what degree of impact the various levels intend to describe.

The risk levels are defined by means of a risk matrix, which yields a risk level for each combination of likelihood and impact. The risk matrix is based on the same EUROCONTROL document and given in Table 13.

Table 13 Risk matrix

		Likelihood			
		4 Rare	**3 Possible**	**2 Occasional**	**1 Frequent**
Impact	**1 Very High**	High	Very high	Critical	Critical
	2 High	Negligible	High	Very high	Critical
	3 Medium	Negligible	Tolerable	High	Very high
	4 Low	Negligible	Tolerable	Tolerable	High
	5 Insignificant	Negligible	Negligible	Negligible	High

The risk evaluation criteria define for each risk level whether the risk is acceptable or must be evaluated for possible treatment. The risk evaluation criteria proposed in

Table 14 were approved by the participants before proceeding with the risk estimation.

Table 14 Risk evaluation criteria

Negligible	Acceptable
Tolerable	Acceptable (Should be monitored; should be treated if treatment cost is justifiable)
High	Unacceptable
Very high	Unacceptable
Critical	Unacceptable

Risk estimation
The task that remains is to do the risk estimation. This is conducted directly in the CORAS threat diagrams by annotating each unwanted incident with a likelihood value, and each relation between an unwanted incident and asset with a consequence value. It is also useful to estimate and document likelihoods for each threat scenario. First, this will increase the understanding of the most important sources of risks. Second, assigning likelihoods to threat scenarios may facilitate the likelihood estimation of the unwanted incidents that the threat scenarios lead to. Third, likelihood estimates of threat scenarios yields a better basis for validating the results. It may also be useful to estimate conditional likelihoods to the relations from threat scenarios to other threat scenarios, and from threat scenarios to unwanted incidents. For a scenario/incident A that leads to scenario incident B, a conditional likelihood describes the likelihood that A will lead to B when A occurs. A conditional likelihood is often specified as a probability, i.e. a value in the interval [0,1].

Notice, however, that at this phase of the development lifecycle the system is not yet designed. For the identified unwanted incidents the objective is therefore to estimate their severity in terms of consequences. Given these consequences, likelihoods are assigned to unwanted incidents that ensure acceptable risk levels as a result. In this way the likelihood estimates serve as requirements to the security risks that the system may be exposed to.

In the following, Figure 68 shows the risk estimation related to insufficient identification and/or authentication in the APP control room. It is important to notice that the likelihood that an unauthorized external actor accesses a CWP in the control room is rare, even if the likelihood of an entering to the control room is possible, because the other ATCOs will recognise the external and malicious actor and give an alert. Moreover, we would like to clarify that the 'switch' between TC and PLC role is very frequent and often useful and recommended. Thus, it not always leads to a deterioration of APP services (that, in fact, has a lower likelihood). Figure 69 shows the risk estimation related to insufficient identification and/or authentication in the APP technical room. In the technical room, many different companies employees collaborated together, thus the identification of an external actor is more complicated (leading to a lower likelihood that in the previous mentioned case). Figure 70 shows the risk estimation related to the principle of least privilege. It is important to mention that the redundancy of information and the information exchange among different roles is a common work practice and it has often a good impact on the APP service provisioning, leading to a deterioration of the services due to the interference of tasks just in few cases. Figure 71, Figure 72 and Figure 73 show the risk estimation related to robustness of external messages with respect to phone lines, radio signals and radar data, respectively. The redundancy of communication and surveillance apparatuses reduces in many cases the likelihood of the threat scenarios.

The consequence estimations are documented on the relations between the unwanted incident and the identified assets. The consequences range from *Insignificant* to *Very high* as defined in the impact scale of Table 12. In the diagrams below, each of the documented consequences represents a (rough) aggregate of the impact of the unwanted incidents on safety, business and/or reputation. Because the aggregated consequences alone may be insufficient for understanding how to adequately mitigate unacceptable risks (which depends on the impacted domain), we have in Table 15 specified of the impact of each incident in the respective domains together with the resulting overall aggregate.

Table 15 Qualification of consequence estimates

	Impact			
Incident	**Safety**	**Business**	**Reputation**	**Aggregate**
Unauthorized execution of APP services	High	Medium	High	High
Provisioning of APP services deteriorates due to mix of ATCO roles of supervising and controlling	High	Medium	Low	High
Provisioning of APP services deteriorates due to PLC-TCC role mix	Low	Negligible	Negligible	Medium
Provisioning of APP services deteriorates due to loss of data access	High	Medium	Low	High
Provisioning of APP services deteriorates due to interference of TCC and PLC tasks	Low	Negligible	Negligible	Medium
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of phone communication	High	Medium	Medium	High
Provisioning of APP services by TCC and PLC deteriorates due to loss of phone lines	High	Medium	Medium	High
Provisioning of APP services by TCC and PLC deteriorates due to loss of radio communication	High	Medium	Medium	High
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of radio communication	High	Medium	Medium	High
Provisioning of APP services by TCC and PLC	High	Medium	Medium	High

deteriorates due to loss of radar data				

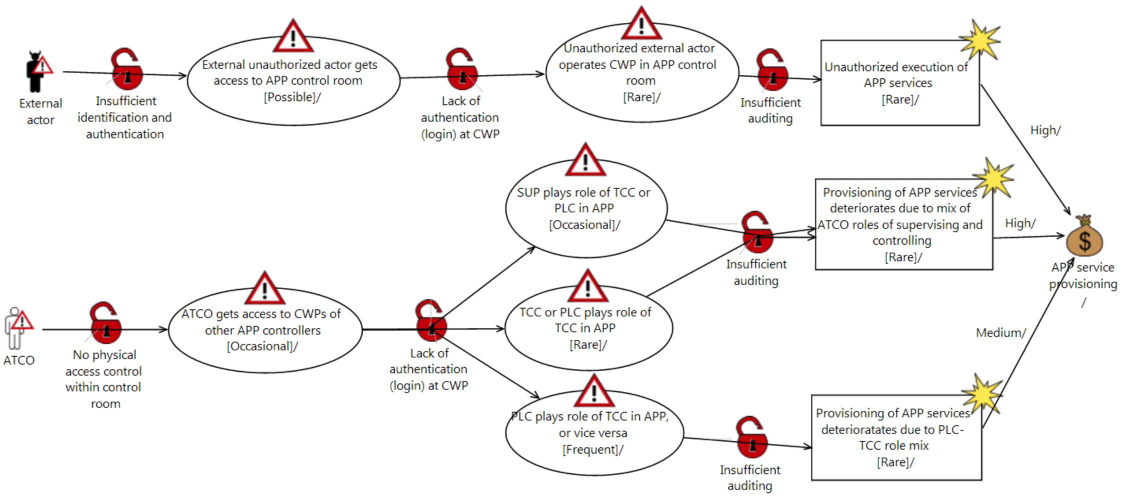

External actor
Insufficient identification and authentication
External unauthorized actor gets access to APP control room [Possible]/
Lack of authentication (login) at CWP
Unauthorized external actor operates CWP in APP control room [Rare]/
Insufficient auditing
Unauthorized execution of APP services [Rare]/
High/
SUP plays role of TCC or PLC in APP [Occasional]/
Insufficient auditing
Provisioning of APP services deteriorates due to mix of ATCO roles of supervising and controlling [Rare]/
High/
APP service provisioning /
ATCO
No physical access control within control room
ATCO gets access to CWPs of other APP controllers [Occasional]/
Lack of authentication (login) at CWP
TCC or PLC plays role of TCC in APP [Rare]/
Medium/
PLC plays role of TCC in APP, or vice versa [Frequent]/
Insufficient auditing
Provisioning of APP services deteriorates due to PLC-TCC role mix [Rare]/

Figure 68 Risk estimation

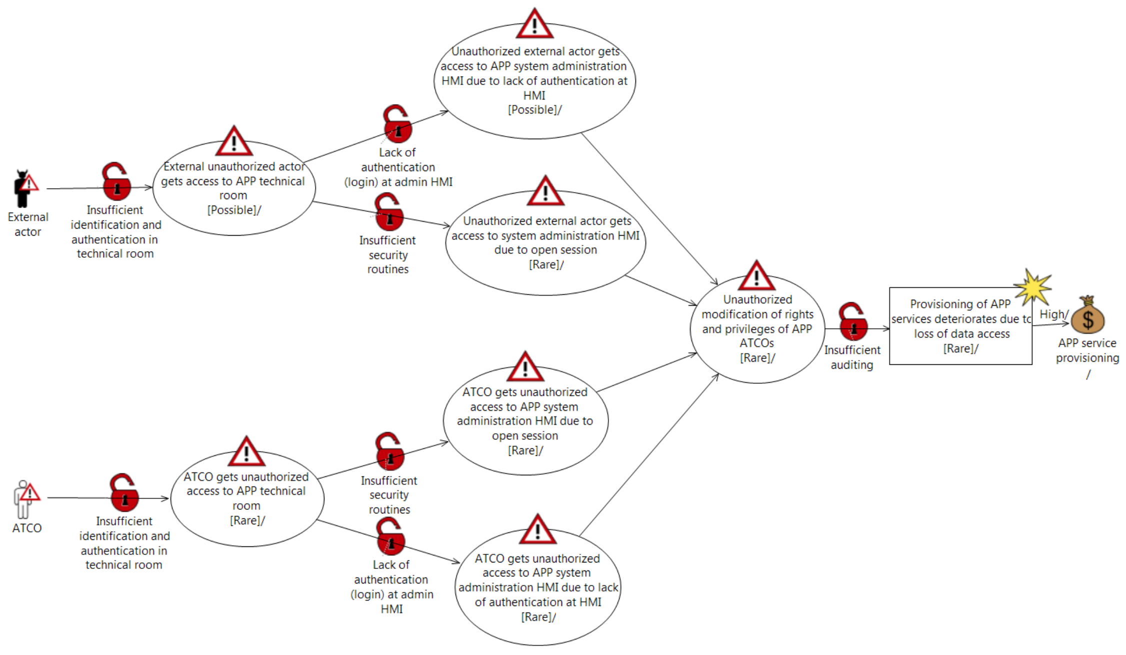

Unauthorized external actor gets access to APP system administration HMI due to lack of authentication at HMI [Possible]/
External actor
Insufficient identification and authentication in technical room
External unauthorized actor gets access to APP technical room [Possible]/
Lack of authentication (login) at admin HMI
Insufficient security routines
Unauthorized external actor gets access to system administration HMI due to open session [Rare]/
Unauthorized modification of rights and privileges of APP ATCOs [Rare]/
Insufficient auditing
Provisioning of APP services deteriorates due to loss of data access [Rare]/
High/
APP service provisioning /
ATCO gets unauthorized access to APP system administration HMI due to open session [Rare]/
ATCO
Insufficient identification and authentication in technical room
ATCO gets unauthorized access to APP technical room [Rare]/
Insufficient security routines
Lack of authentication (login) at admin HMI
ATCO gets unauthorized access to APP system administration HMI due to lack of authentication at HMI [Rare]/

Figure 69 Risk estimation

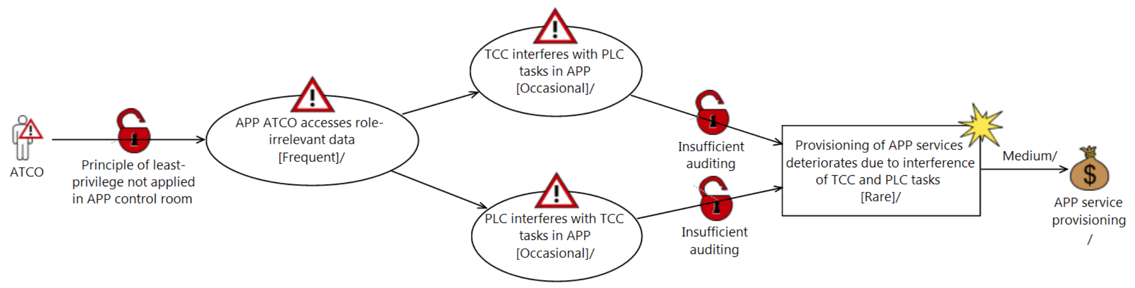
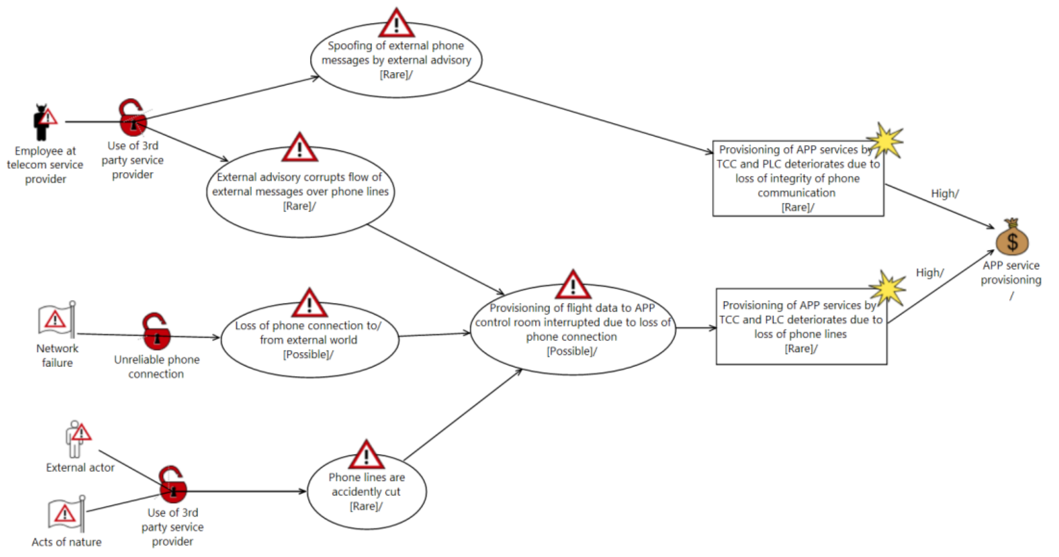
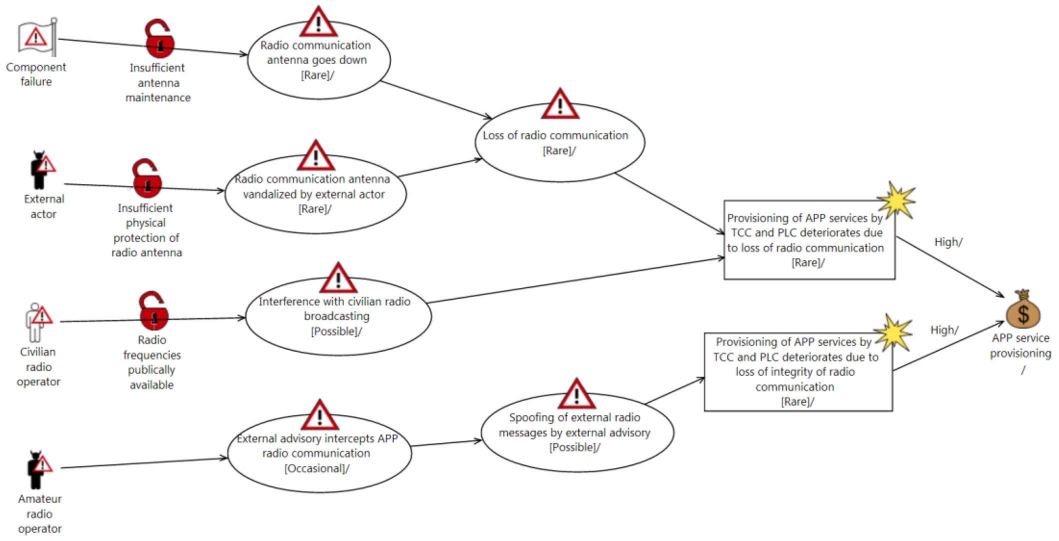

TCC interferes with PLC tasks in APP [Occasional]/
ATCO
Principle of least-privilege not applied in APP control room
APP ATCO accesses role-irrelevant data [Frequent]/
Insufficient auditing
Provisioning of APP services deteriorates due to interference of TCC and PLC tasks [Rare]/
Medium/
APP service provisioning /
PLC interferes with TCC tasks in APP [Occasional]/
Insufficient auditing

Figure 70 Risk estimation



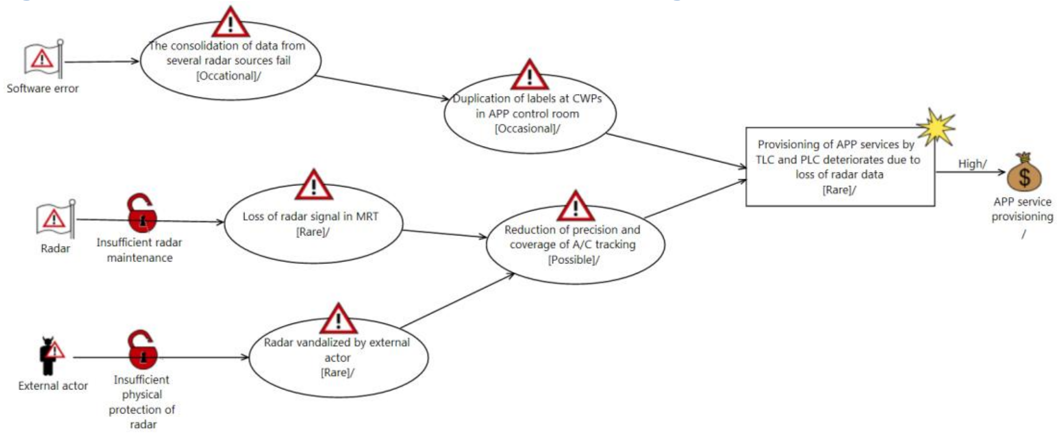
Spoofing of external phone messages by external advisory [Rare]/
Employee at telecom service provider
Use of 3rd party service provider
External advisory corrupts flow of external messages over phone lines [Rare]/
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of phone communication [Rare]/
High/
APP service provisioning /
High/
Network failure
Unreliable phone connection
Loss of phone connection to/ from external world [Possible]/
Provisioning of flight data to APP control room interrupted due to loss of phone connection [Possible]/
Provisioning of APP services by TCC and PLC deteriorates due to loss of phone lines [Rare]/
External actor
Acts of nature
Use of 3rd party service provider
Phone lines are accidently cut [Rare]/

Figure 71 Risk estimation wrt robustness wrt external messages: Phone lines



Component failure
Insufficient antenna maintenance
Radio communication antenna goes down [Rare]/
Loss of radio communication [Rare]/
External actor
Insufficient physical protection of radio antenna
Radio communication antenna vandalized by external actor [Rare]/
Provisioning of APP services by TCC and PLC deteriorates due to loss of radio communication [Rare]/
High/
Civilian radio operator
Radio frequencies publically available
Interference with civilian radio broadcasting [Possible]/
APP service provisioning /
High/
Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of radio communication [Rare]/
Spoofing of external radio messages by external advisory [Possible]/
External advisory intercepts APP radio communication [Occasional]/
Amateur radio operator

Figure 72 Risk estimation wrt robustness wrt external messages: Radio



Software error
The consolidation of data from several radar sources fail [Occational]/
Duplication of labels at CWPs in APP control room [Occasional]/
Provisioning of APP services by TLC and PLC deteriorates due to loss of radar data [Rare]/
High/
APP service provisioning /
Radar
Insufficient radar maintenance
Loss of radar signal in MRT [Rare]/
Reduction of precision and coverage of A/C tracking [Possible]/
External actor
Insufficient physical protection of radar
Radar vandalized by external actor [Rare]/

Figure 73 Risk estimation wrt robustness wrt external messages

Risk Evaluation

The identified and estimated risks are evaluated with respect to the already defined risk evaluation criteria. The results are in the following documented by means of CORAS risk diagrams. These diagrams show all the risks with their risk level, as well as the threats that initiate them and the assets that are harmed.

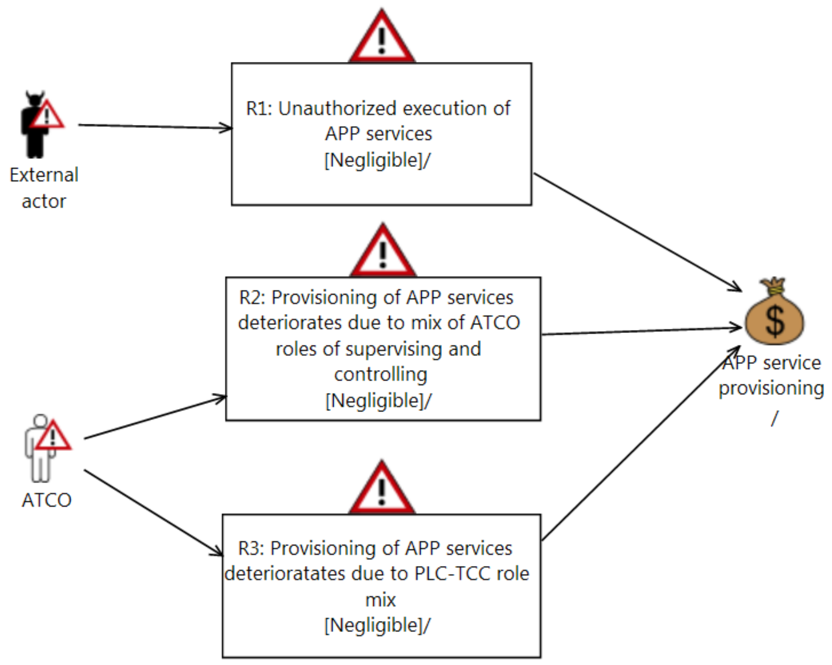

External actor
R1: Unauthorized execution of APP services [Negligible]/
R2: Provisioning of APP services deteriorates due to mix of ATCO roles of supervising and controlling [Negligible]/
ATCO
R3: Provisioning of APP services deterioratates due to PLC-TCC role mix [Negligible]/
APP service provisioning /

Figure 74 Risk evaluation

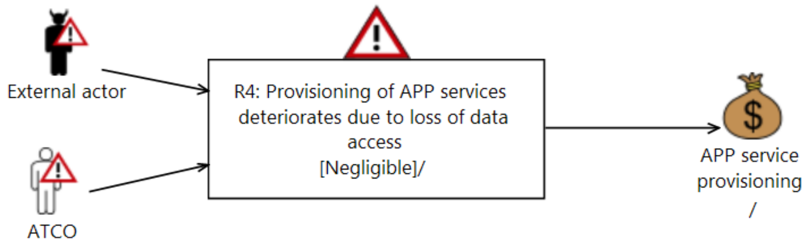

External actor
ATCO
R4: Provisioning of APP services deteriorates due to loss of data access [Negligible]/
APP service provisioning /

Figure 75 Risk evaluation wrt identification and authentication and auditing

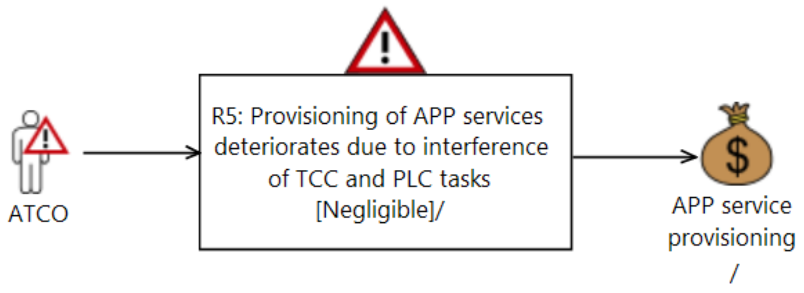
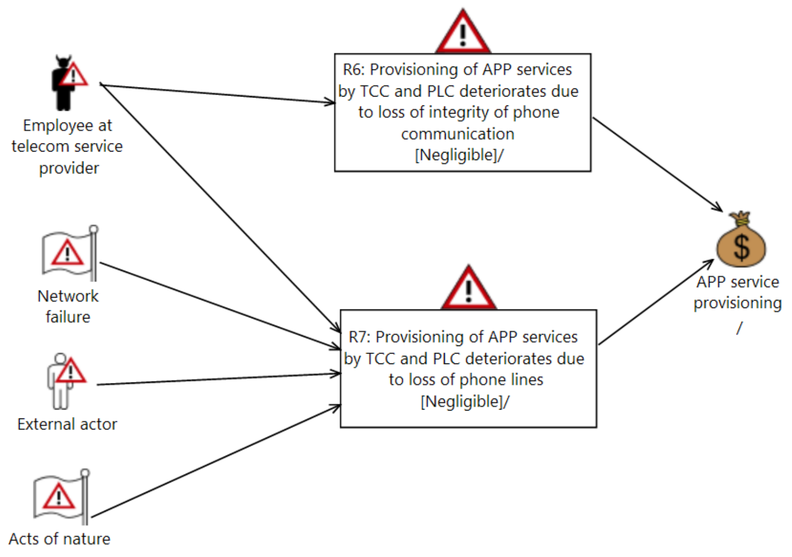
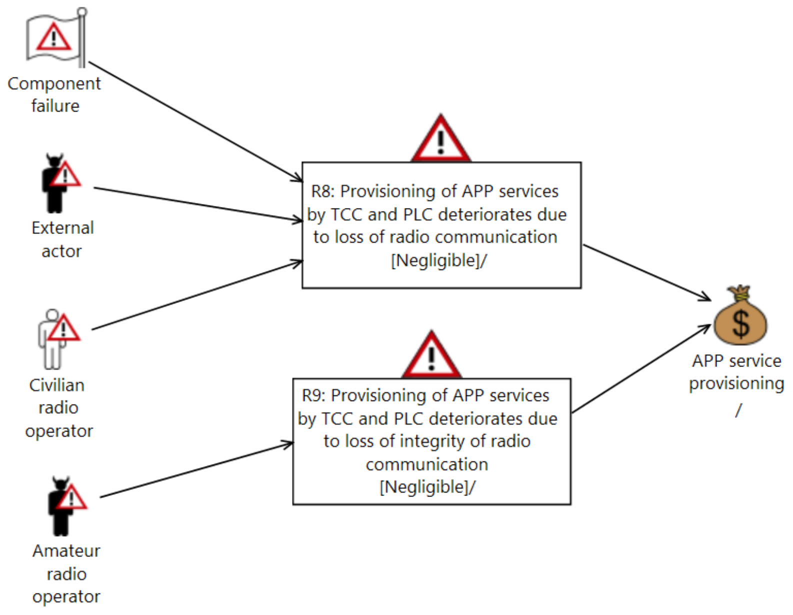

ATCO
R5: Provisioning of APP services deteriorates due to interference of TCC and PLC tasks [Negligible]/
APP service provisioning /

Figure 76 Risk evaluation wrt least privilege and auditing



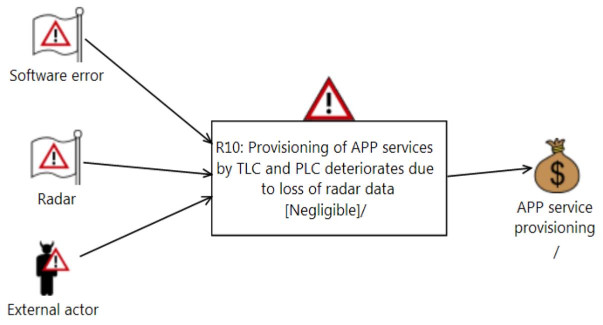
Employee at telecom service provider
Network failure
External actor
Acts of nature
R6: Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of phone communication [Negligible]/
R7: Provisioning of APP services by TCC and PLC deteriorates due to loss of phone lines [Negligible]/
APP service provisioning /

Figure 77 Risk evaluation wrt robustness wrt external messages



Component failure
External actor
Civilian radio operator
Amateur radio operator
R8: Provisioning of APP services by TCC and PLC deteriorates due to loss of radio communication [Negligible]/
R9: Provisioning of APP services by TCC and PLC deteriorates due to loss of integrity of radio communication [Negligible]/
APP service provisioning /

Figure 78 Risk evaluation wrt robustness wrt external messages



Software error
Radar
External actor
R10: Provisioning of APP services by TLC and PLC deteriorates due to loss of radar data [Negligible]/
APP service provisioning /

Figure 79 Risk evaluation wrt external messages

We use the risk matrix to give a summary and overview of the results of the risk evaluation. In Table 16 each of the ten identified risks is plotted into the risk matrix according to the estimated likelihood and consequence. We see that there are three negligible risks, whereas three risks are high and four risks are very high.

Table 16 Risk evaluation overview

		Likelihood			
		4 Rare	**3 Possible**	**2 Occasional**	**1 Frequent**
Impact	**1 Very High**	🟧	🟥	🟥	🟥
	2 High	🟩 R1 R2 R4 R6 R7 R8 R9 R10	🟧	🟥	🟥
	3 Medium	🟩 R3 R5	🟨	🟧	🟥
	4 Low	🟩	🟨	🟨	🟧
	5 Insignificant	🟩	🟩	🟩	🟧

Step n°4: System / software specification using SMS

The Thales SOA Modelling Suite (SMS) is not a work product of SecureChange, but it has been used as a representative workbench to integrated Thales' risk assessment DSML matured in the scope of SecureChange. Thus, it has been used for the system / software specification of the ATM case-study, and is briefly described in what follows The purpose of SMS is to capture the different concerns related to service oriented architecture (SOA) architectures specifications and implementations. It is a research prototype still under development. 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. Using SMS, users can create a project and use the modeller to edit SOA models, eventually generating the appropriate documents. Multiple features are available including, for instance, high level service and message type specification, logical view specification, physical view specification, BPMN 2.0, enterprise integration patterns. For the ATM case-study, a large number of diagrams were modelled using SMS (cf. Figure 80).

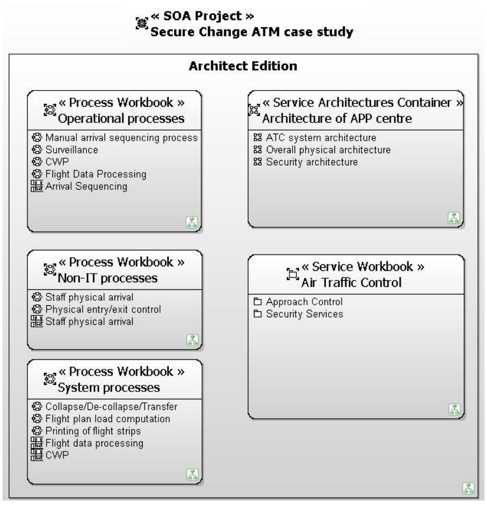


Figure 80 Overall system / software specification using SMS

A collection of diagrams is provided below to show the scope of the coverage. The 1st diagram is a solution for the identification, authentification and auditing security needs, for which a number of risks were raised using CORAS (cf. Figure 62). Using the Business Process Modelling Notation (BPMN) it specifies how a controller (executive or planner) is given access to his working position (cf. Figure 81). Process modelling can also be performed at a more detailed operational level (cf. Figure 82).

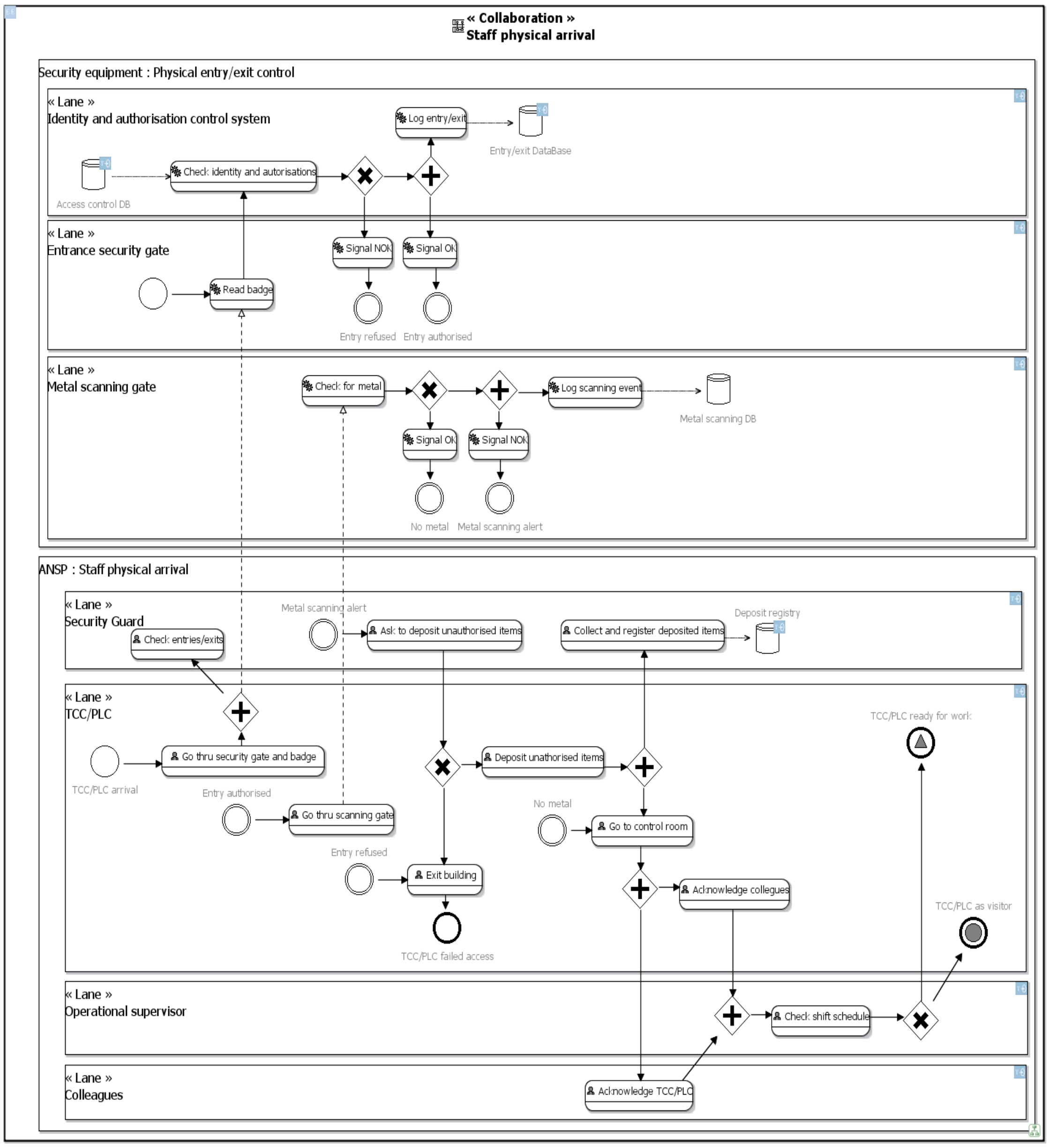


Figure 81 Staff physical arrival process (BPMN)

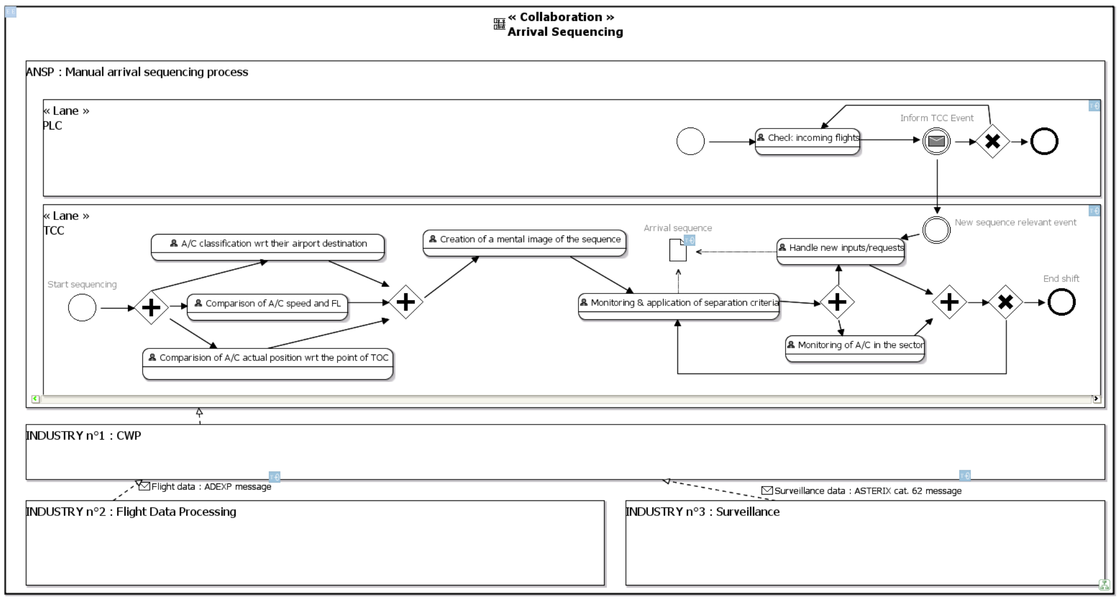

« Collaboration »
Arrival Sequencing
ANSP : Manual arrival sequencing process
« Lane » PLC
Check incoming flights
Inform TCC Event
« Lane » TCC
A/C classification wrt their airport destination
Creation of a mental image of the sequence
Arrival sequence
New sequence relevant event
Handle new inputs/requests
Start sequencing
Comparison of A/C speed and FL
Monitoring & application of separation criteria
End shift
Comparision of A/C actual position wrt the point of TOC
Monitoring of A/C in the sector
INDUSTRY n°1 : CWP
Flight data : ADEXP message
Surveillance data : ASTERIX cat. 62 message
INDUSTRY n°2 : Flight Data Processing
INDUSTRY n°3 : Surveillance

Figure 82 Aircraft arrival sequencing collaboration (BPMN)

Figure 83 shows the specification of the main ATC centre components (FDPS, RDPS, CWP and WAN) and the main communications between these components through service subscriptions.

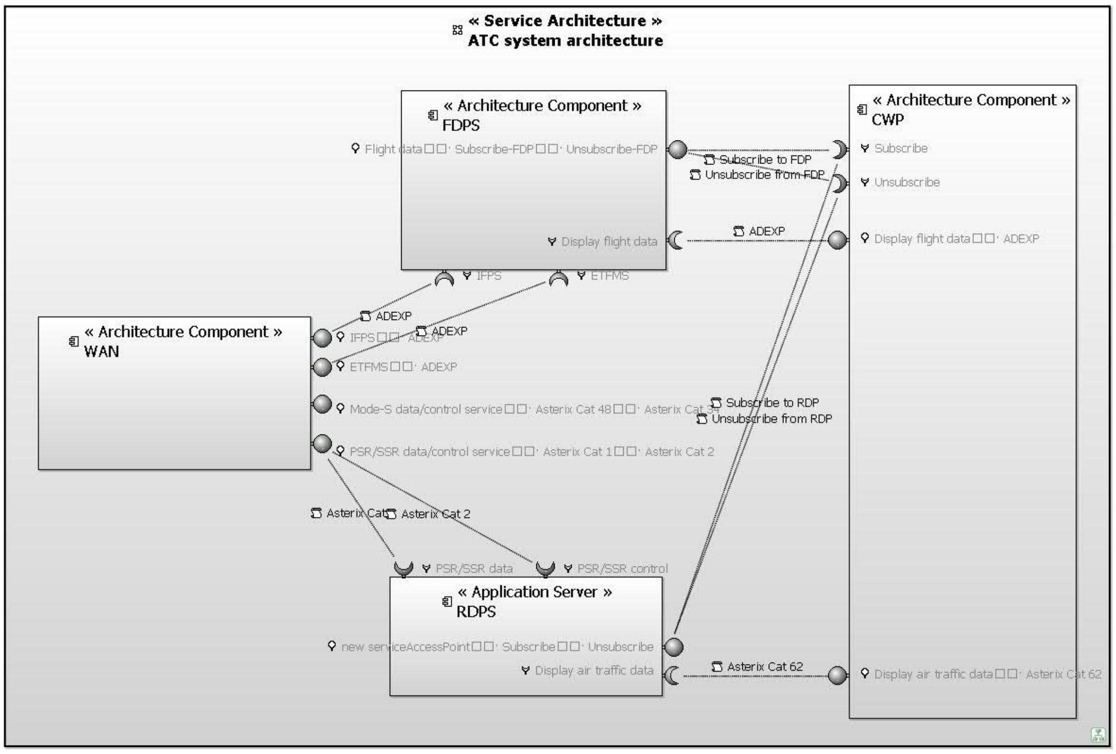

« Service Architecture »
ATC system architecture
« Architecture Component » FDPS
Flight data Subscribe-FDP Unsubscribe-FDP
Display flight data
IFPS
ETFMS
« Architecture Component » CWP
Subscribe
Unsubscribe
Display flight data ADEXP
Subscribe to FDP
Unsubscribe from FDP
ADEXP
« Architecture Component » WAN
ADEXP
IFPS ADEXP
ADEXP
ETFMS ADEXP
Mode-S data/control service Asterix Cat 48 Asterix Cat 34
Subscribe to RDP
Unsubscribe from RDP
PSR/SSR data/control service Asterix Cat 1 Asterix Cat 2
Asterix Cat 1
Asterix Cat 2
PSR/SSR data
PSR/SSR control
« Application Server » RDPS
new serviceAccessPoint Subscribe Unsubscribe
Display air traffic data
Asterix Cat 62
Display air traffic data Asterix Cat 62

Figure 83 ATC system architecture

With SMS, these services are organised in service portfolios. The approach control portfolio contains both a description of information domains (in terms of enumerations, data types and messages) relevant to approach control services, and a description of service domains (cf. Figure 84).

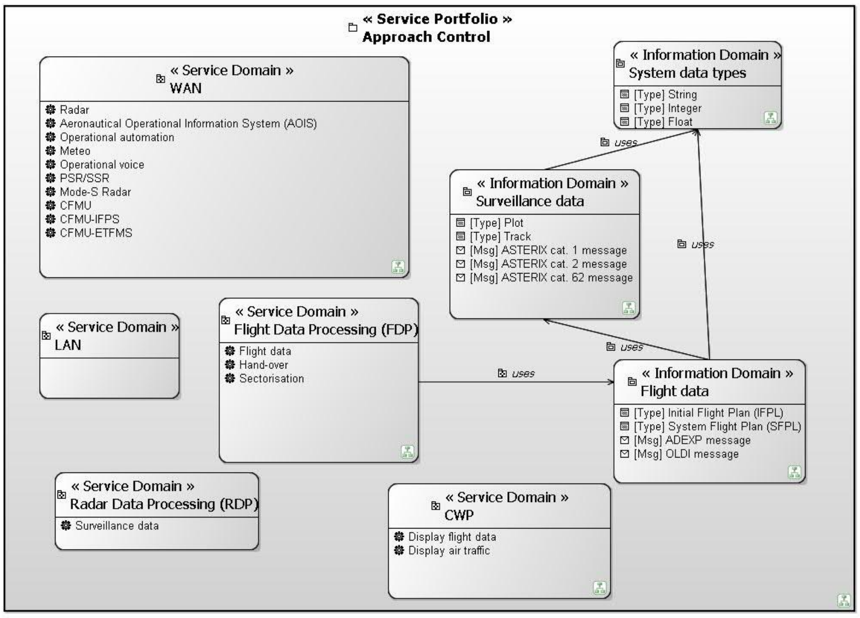
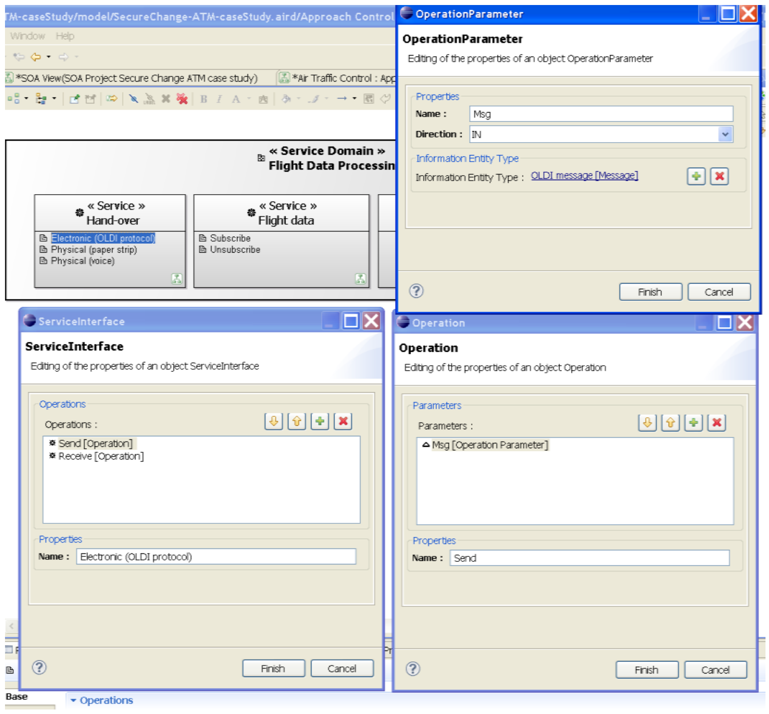


Figure 84 Approach control service portfolio

Each service of a service domain can then be finely specified in terms of service interface, operation and parameters (cf. Figure 85).



OperationParameter

OperationParameter

Editing of the properties of an object OperationParameter

Properties

Name : Msg

Direction : IN

Information Entity Type

Information Entity Type : OLDI message [Message]

ServiceInterface

ServiceInterface

Editing of the properties of an object ServiceInterface

Operations

Operations :

Send [Operation]

Receive [Operation]

Properties

Name : Electronic (OLDI protocol)

Operation

Operation

Editing of the properties of an object Operation

Parameters

Parameters :

Msg [Operation Parameter]

Properties

Name : Send

Figure 85 The electronic hand-over service specification

According to the ATM risk assessment performed by SINTEF with CORAS, a number of unwanted incidents were identified with:

- a list of threat scenarios leading to these unwanted incidents,
- the severity level of each unwanted incident.

At the phase of the development lifecycle at which the CORAS risk assessment study was performed, the system is not yet designed, so it is impossible to “assess” the likelihood of the threat scenarios. It is however possible to state likelihoods as constraints on the threat scenarios so as to make the identified risks acceptable, given the severity of the unwanted incidents.

Example: the “Unauthorised execution of APP services” stands out as the most severe unwanted incident, with respect to the “APP service provisioning” goal (i.e. primary asset), with an overall severity rated “Very 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”, 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 requirements and the like represent the main inputs for the system specification phase performed herein.

For example, in Figure 81, the Staff physical arrival collaboration (BPMN) was designed using multiple levels of security checks, to make the likelihoods of the threats as low as possible, within reasonable costs. For other processes, e.g. the Aircraft arrival sequencing collaboration (BPMN), as pictured in Figure 82, there is no security measure, as no risks has been identified here by the CORAS risk assessment study.

Step n°5: System-level risk assessment using Rinforzando
The risk assessment performed here is not redundant with the CORAS risk assessment because this risk assessment takes as input the system design (in terms of system requirements). It is assumed at this stage that the system will be implemented as designed, but this will later need to be checked through qualification actions, e.g. audits. See deliverable D4.4b for in-depth explanations on the security engineering process and change management for this specific engineering activity (deliverable D1.3 takes an operational point of view on the ATM case-study, whereas D4.4b takes a more technical point of view).

The risk assessment is illustrated herein on two essential elements:

- the staff physical arrival process, which is key to at least three of the security needs elicited in Step n°2: Security need elicitation, namely:
    a. identification and authentication,
    b. auditing support,

c. secured physical access to equipment,

- the arrival sequencing process.

The staff physical arrival process risk assessment is pictured in Figure 86. It can be seen that the staff physical arrival process (essential element) is supported by a large number of "targets", some of which are equipments (e.g. access control DB, deposit registry, entry/exit database, metal scanning gate, etc.), whilst others are people (e.g. security guard, controllers, operational supervisor). All these elements come from the mainstream system engineering activity performed using SMS.

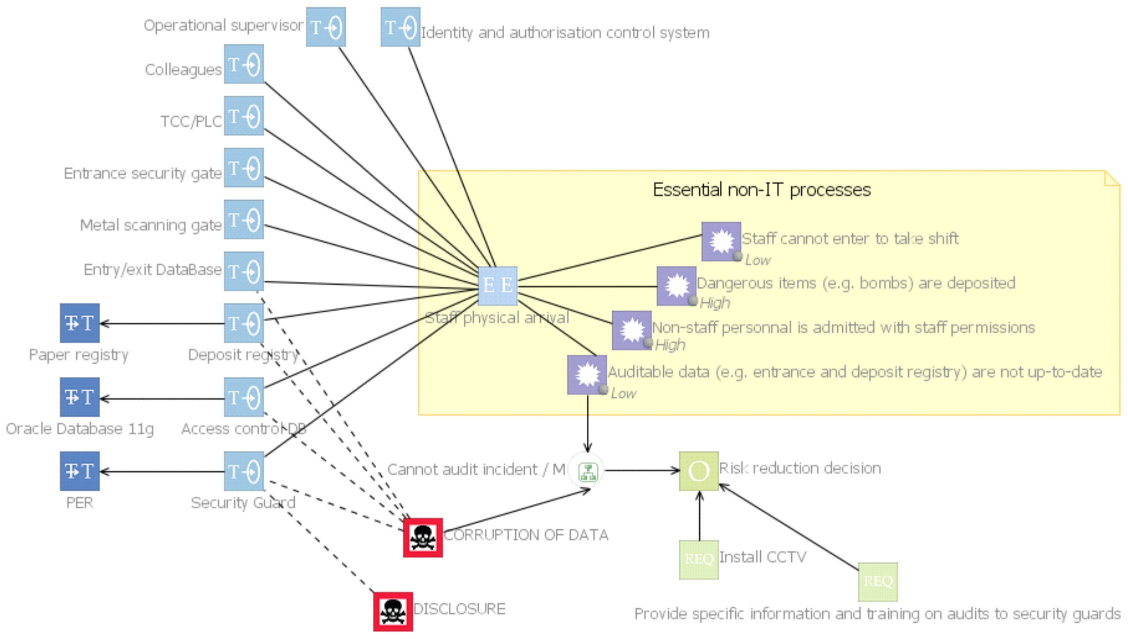

Operational supervisor
Identity and authorisation control system
Colleagues
TCC/PLC
Entrance security gate
Metal scanning gate
Entry/exit DataBase
Paper registry
Deposit registry
Oracle Database 11g
Access control DB
PER
Security Guard
Essential non-IT processes
Staff physical arrival
Staff cannot enter to take shift
Low
Dangerous items (e.g. bombs) are deposited
High
Non-staff personnal is admitted with staff permissions
High
Auditable data (e.g. entrance and deposit registry) are not up-to-date
Low
Cannot audit incident / M
Risk reduction decision
CORRUPTION OF DATA
Install CCTV
DISCLOSURE
Provide specific information and training on audits to security guards

Figure 86 Risk assessment on the staff physical arrival process

Figure 81 shows that when these supporting assets are selected in the mainstream system engineering model, these elements are tagged in the SMS tool with the target symbol (T). The staff physical arrival process is tagged as an essential element (E E).

Thus, the engineers responsible for the mainstream system engineering design are aware in real time that these key-elements are in the risk assessment study. This information may be considered cumbersome, or even confidential. This is why it is also possible to hide or display the risk assessment tags by activating specific layers in the SMS tool: it may be envisaged that specific security policies will give the rights to activate those layers only to specific roles.

Figure 86 shows the damages that can be expected if the staff physical arrival process malfunctions or fails: the staff may not be able to enter to take its shift, dangerous items (e.g. bombs) can be deposited, non-staff personal may be admitted with staff permissions, or auditable data (e.g. entrance and deposit registry) may not be up-to-date. A risk exists when there is a conjunction of threats on targets and feared events (or damages) on essential elements.

Figure 86 shows only two threats: the corruption of data and the disclosure of confidential data. The analysis of the corruption of data threat shows that the security guard is involved in all scenarios (cf. Figure 87), because his complicity is required at all times. Thus, a risk exists with respect to the impossibility of auditing, especially in case of a security incident. This risk was assessed as tolerable, but it was still decided to reduce it: this is modelled as a security objective (O) in Figure 86.

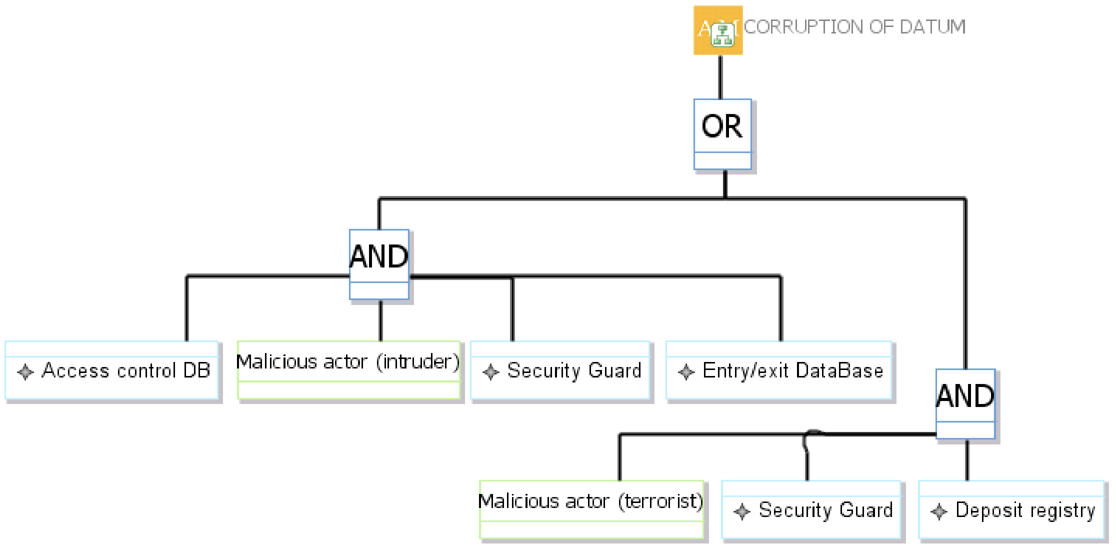


Figure 87 Corruption of data threat scenario

As a result, two security requirements are added: one related to more equipment (i.e. install a CCTV) and one related to a procedure (i.e. provide specific information and training on audits to security guards). With respect to the arrival sequencing process, the risk assessment (cf. Figure 88) shows one damage (i.e. inefficient runway usage – with a “medium” impact), and one threat (i.e. an ATCO intentionally sub-optimises the sequencing – with a “rare” likelihood). The risk is thus negligible, and therefore there is no attached security objective.

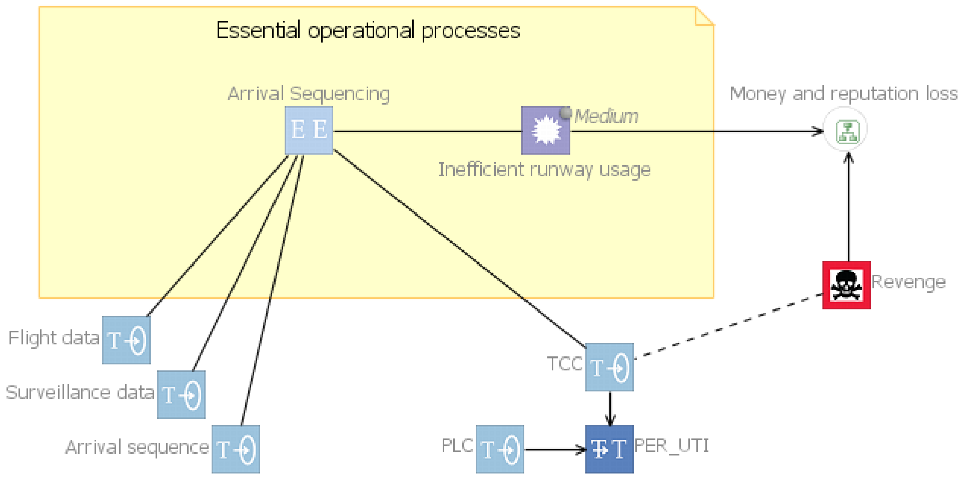


Figure 88 Risk assessment on the arrival sequencing process

Step n°7: Detailed design using Papyrus

This step refines the specification carried out in step 4. For example we developed a UML activity diagram (see Figure 89) refining the arrival for the TCC, which is part of the BPMN process in Figure 81. The diagram was developed with Papyrus UML. Notes above the diagram indicate actors and activities are aligned with the note of their actor.

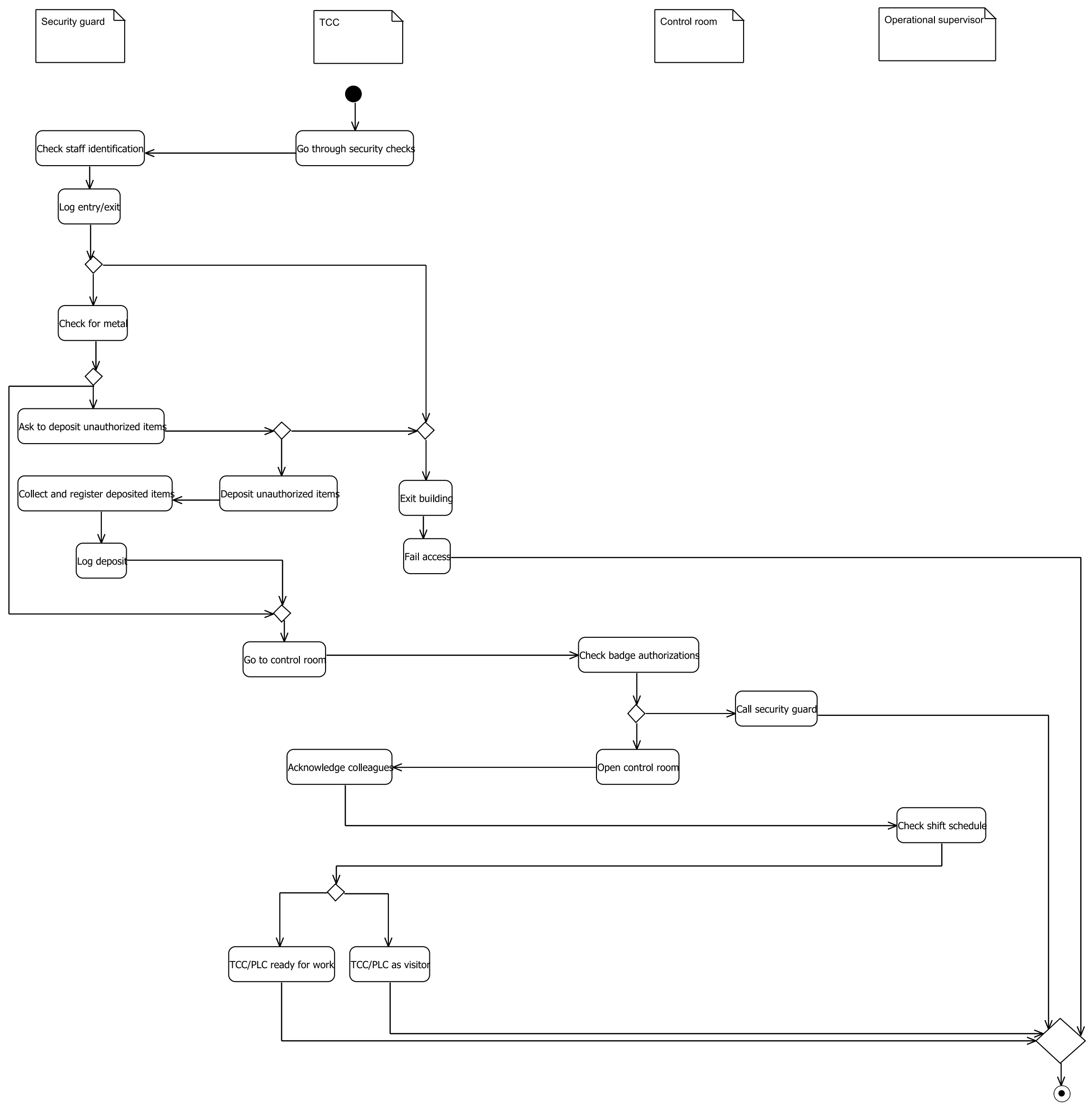


Figure 89 Activity diagram for the physical arrival of the TCC

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. This model can then be analysed using UMLsec.

Step n°8: Formal verification of the security properties using UMLsec

Several security properties of the process in Figure 89 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 demonstrate how several UMLsec analyses can help verify that the process has those properties. Each property is encoded into a conjunction of UMLsec analyses:

Identification means that the TCC is only allowed to work after going through several identification steps, and that the only outcomes of any identification step are to proceed to the next step or to exit the area. This is guaranteed by the conjunction of the following UMLsec analyses:

- Fair exchange with {start = Go through security checks} and {stop = Check staff identification} ensures the TCC will be checked as a member of staff.
- Fair exchange with {start = Check staff identification} and {stop = Check for metal, Exit building} ensures that members of staff will be checked for metal and the others will leave the building.
- Fair exchange with {start = Check for metal} and {stop = Go to control room, Exit building} ensures that members of staff carrying unauthorized items will exit the building while the others will proceed to the control room.
- Fair exchange with {start = Go to control room} and {stop = Check badge authorizations} ensures that any path to the control room is protected by a badge check.
- Fair exchange with {start = Check badge authorizations} and {stop = Open control room, Call security guard} ensures that all staff members attempting to enter the control room will either succeed or be escorted out by a security guard.
- Fair exchange with {start = Open control room} and {stop = Acknowledge colleagues} ensures that all staff members entering the control room will need to be acknowledged by their colleagues.
- Fair exchange with {start = Open control room} and {stop = Check shift schedule} ensures that all staff members entering the control room will need to be checked against the schedule.
- Provable with {cert = TCC/PLC ready for work} each one of {Check staff identification, Check for metal, Check badge authorizations, Acknowledge colleagues, Check shift schedule} for {action} ensures that all the identification steps must have been passed before the TCC is allowed to work.
- Rbac with {"Check staff identification, Check for metal"}⊆{"protected"}, {("Check staff identification" ,"Security guard"),("Check for metal", "Security guard")}⊆{"right"}, and {("Security guard", "Security guard")}⊆{"role"} ensures that only the security guard can check the identification and the presence of metal.
- Rbac with {"Check shift schedule"}⊆{"protected"}, {("Check shift schedule", "Shift manager")}⊆{"right"}, and {("Shift manager", "Operational supervisor")}⊆{"role"} ensures that only the supervisor can check the shift schedule.

Auditing means that some operations must be logged, and that the integrity of the log must be guaranteed. Here is an example for the staff identification:

- Fair exchange with {start = Check identification} and {stop = Log entry/exit} ensures that all staff identifications are logged.
- Provable with {action = Check staff identification} and {cert = Log entry/exit} ensures that logging of an entry or exit may only take place after a staff identification has been performed.
- Rbac with {"Log entry/exit"}⊆{"protected"}, {("Log entry/exit" ,"Security guard")}⊆{"right"}, and {("Security guard", "Security guard")}⊆{"role"} ensures that only the security guard can log entries and exits.

It can be seen here that some seemingly simple security properties can only be completely verified by a combination of checks using UMLsec, and that this combination is not straightforward.

Interation n°2: The system "to be"

This section presents the risk modelling resulting from the risk identification that was conducted as the 3rd step of the 2nd iteration, i.e. after the system changes of the AMAN introduction. This step is conducted with respect to the SI* models of the 1st step of the 2nd iteration, as well as the security needs that were elicited during the 2nd step. At the same time the CORAS risk identification and modelling builds on the CORAS risk models from the 1st iteration updating these according to changes to the risk picture, and also takes into account new knowledge gained from the system specification and design during the 1st iteration. One part of the risk identification and estimation after the changes is to conduct a systematic walkthrough of the risk models before change to determine whether these are affected by the changes either by changes to the risk levels or by parts of the risk picture becoming obsolete. Another part is to identify and assess new risks that may arise. The CORAS risk modelling language for changing risks supports the explicit modelling of such changes. Risk that become obsolete are represented in grey colour, risks that are present both before and after changes are represented by two-layered icons, whereas risks that arise are represented by regular, coloured CORAS icons. Moreover, changes in risk levels are captured by pairs of likelihoods and pairs of consequences for the before-after icons, where the former value of the pair denotes the value before the changes and the latter denotes the value after the changes. The general structure of the risk assessment after change is as for the risk assessment before change. We will therefore not repeat and explain the full process. We rather focus on the changes and how these are handled and modelled.

Security Needs after Change

The security needs are as before changes except for the addition of one security need, namely confidentiality. This is due to a concern about State Flight information that after the AMAN introduction needs to be managed electronically since the data needs to be processed by the AMAN. Before the AMAN it was only the ATCOs that required knowing about this sensitive data. After the AMAN introduction the System Flight Plan (SFPL) is modified to carry this information.

Threat Diagrams after Change

In the following we present the CORAS threat diagrams completed with likelihood and consequence estimates. The asset *APP service provisioning* is still considered, while the new asset *State Flight information* is introduced. The scales for likelihoods and consequences are as before, as are also the risk matrix and risk evaluation criteria. As shown by Figure 90, the unwanted incidents with respect to insufficient identification and/or authentication in the APP control room, as well as secured physical access, are persistent under change. However, the threat scenario of an external unauthorized actor accessing the APP control room is considered to decrease in likelihood as is shown by the shift from *Possible* to *Rare*. This decrease is due to the documentation of the staff physical arrival as part of the system specification using SMS during the 1st iteration. Because *Rare* is the lowest likelihood value the likelihood of the unwanted incident *Unauthorized execution of APP services* does not change, although it can be assumed that it is even lower as before. It is only if the unwanted incident can be completely discharged that it becomes obsolete and removed from the risk model; in that case the likelihood is considered as being zero.

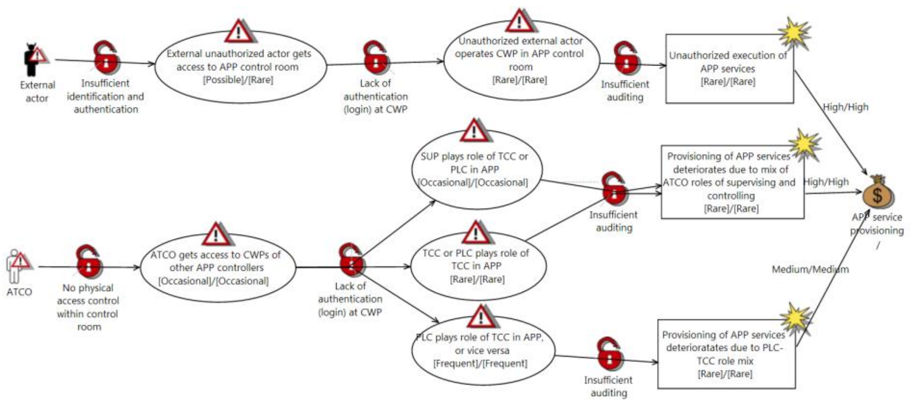

External actor
Insufficient identification and authentication
External unauthorized actor gets access to APP control room [Possible]/[Rare]
Lack of authentication (login) at CWP
Unauthorized external actor operates CWP in APP control room [Rare]/[Rare]
Insufficient auditing
Unauthorized execution of APP services [Rare]/[Rare]
High/High
SUP plays role of TCC or PLC in APP [Occasional]/[Occasional]
Insufficient auditing
Provisioning of APP services deteriorates due to mix of ATCO roles of supervising and controlling [Rare]/[Rare]
High/High
APP service provisioning
ATCO
No physical access control within control room
ATCO gets access to CWPs of other APP controllers [Occasional]/[Occasional]
Lack of authentication (login) at CWP
TCC or PLC plays role of TCC in APP [Rare]/[Rare]
Medium/Medium
PLC plays role of TCC in APP, or vice versa [Frequent]/[Frequent]
Insufficient auditing
Provisioning of APP services deterioratates due to PLC-TCC role mix [Rare]/[Rare]

Figure 90 Identification and authentication, physical access and auditing

Similarly, Figure 91 shows the changes to the risks related to identification and authentication, as well as auditing, regarding the technical room. In this case all threat scenarios related to unauthorized access of external actor are considered as being *Rare*, two scenarios dropping from the likelihood *Possible* before the changes.

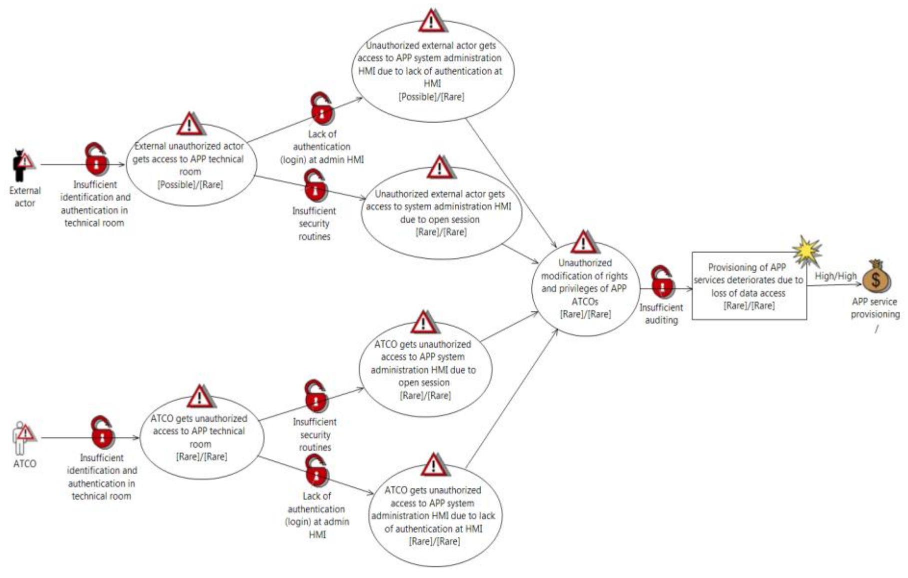

External actor
Insufficient identification and authentication in technical room
External unauthorized actor gets access to APP technical room [Possible]/[Rare]
Lack of authentication (login) at admin HMI
Unauthorized external actor gets access to APP system administration HMI due to lack of authentication at HMI [Possible]/[Rare]
Insufficient security routines
Unauthorized external actor gets access to system administration HMI due to open session [Rare]/[Rare]
Unauthorized modification of rights and privileges of APP ATCOs [Rare]/[Rare]
Insufficient auditing
Provisioning of APP services deteriorates due to loss of data access [Rare]/[Rare]
High/High
APP service provisioning
ATCO gets unauthorized access to APP system administration HMI due to open session [Rare]/[Rare]
ATCO
Insufficient identification and authentication in technical room
ATCO gets unauthorized access to APP technical room [Rare]/[Rare]
Insufficient security routines
Lack of authentication (login) at admin HMI
ATCO gets unauthorized access to APP system administration HMI due to lack of authentication at HMI [Rare]/[Rare]

Figure 91 Identification and authentication and auditing: Technical room

The remaining risks from the 1st iteration before the changes are considered as not being affected by the changes, and therefore remain the same, both in terms of threat scenarios and likelihood and consequence estimates. We do therefore not show these diagrams again here.

Figure 92 on the other hand shows an unwanted incident that may occur after the changes. Because State Flight information now must be processed by the AMAN, it is managed electronically and carried by the System Flight Plan (SFPL) and may therefore be available to unauthorized personnel via the wide area network (WAN). The identified vulnerabilities are insufficient access control with respect to the WAN, as well as insufficient application of the need-to-know (NtK) principle with respect to the information carried by the SFPL.

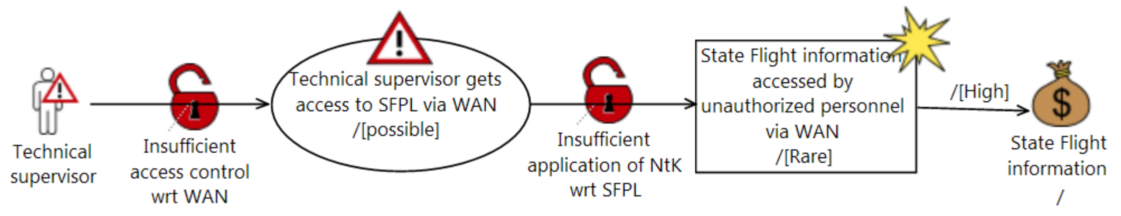

Technical supervisor
Insufficient access control wrt WAN
Technical supervisor gets access to SFPL via WAN /[possible]
Insufficient application of NtK wrt SFPL
State Flight information accessed by unauthorized personnel via WAN /[Rare]
/[High]
State Flight information /

Figure 92 Confidentiality of State Flight information

Risk Evaluation after Change

The evaluation of the identified risks with respect to the risk evaluation criteria are documented by means of CORAS threat diagrams. Because the levels of the risks before the changes are the same also after changes, we do not repeat these risk diagrams here. Figure 93 show the evaluation of the risk that is due to the new unwanted incident that was identified with respect to confidentiality of State Flight information.

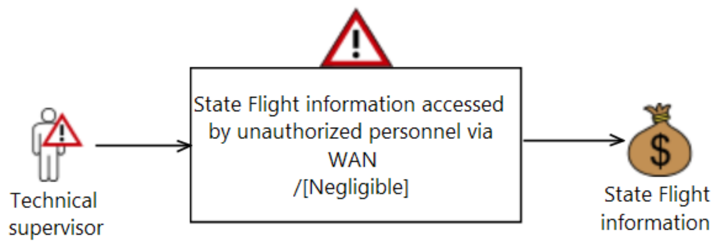

Technical supervisor
State Flight information accessed by unauthorized personnel via WAN /[Negligible]
State Flight information

Figure 93 Risk evaluation

System / software specification using SMS

The main change introduced in iteration n°2 with respect to iteration n°1 is the introduction of an Arrival Manager (AMAN). This mainly affects the arrival sequencing process.

The arrival sequencing in its “as is” version was described in Figure 82. On that figure, it is possible to see that the arrival sequencing is a collaboration between the TCC and the PLC supported by three equipments: a controller working position (CWP), a flight data processing sub-system (FDPS), and a surveillance sub-system. This collaboration (in the BPMN meaning) is completely revised with the introduction of the AMAN.

The AMAN process (Description freely inspired from "Arrival management with required navigation performance and 3D paths", by Aslaug Haraldsdottir, Julien Scharl, Matthew E. Berge, Ewald G. Schoemig, Michael L. Coats, Boeing Commercial Airplanes, presented at the 7th US/Europe ATM R&D Seminar, Barcelona, Spain, July 2-5, 2007) consists of three main tasks (cf. Figure 94):

- The Trajectory Prediction uses a model of aircraft performance to predict the Estimated Time of Arrival (ETA) of each aircraft at the meter fix and at the runway. The perfect trajectory prediction is degraded to represent several key prediction error factors (e.g. wind, temperature, aircraft position).
- The Traffic Scheduling is capable of optimising arrival sequences and schedules in the presence of a discrete delay field, which is a product of the discrete paths and speeds that are used for delay absorption. The Traffic Scheduling is typically executed every 2 minutes and includes all aircraft that fall inside the planning horizon and upstream of the freeze horizon. It produces an arrival schedule that minimizes delay, subject to minimum in-trail spacing requirements at the runway and at each meter fix. The output consists of Scheduled Times of Arrival (STA) for each aircraft in the planning horizon at the runway and at the meter fix.
- The Trajectory Selection. Each STA is either equal to the ETA, if no spacing conflicts are present, or larger than the ETA by the required delay. Associated with each delay value is a particular path and speed combination. The Trajectory Selection task searches through the set of delay values to determine the path and speed combination that implements the required delay.

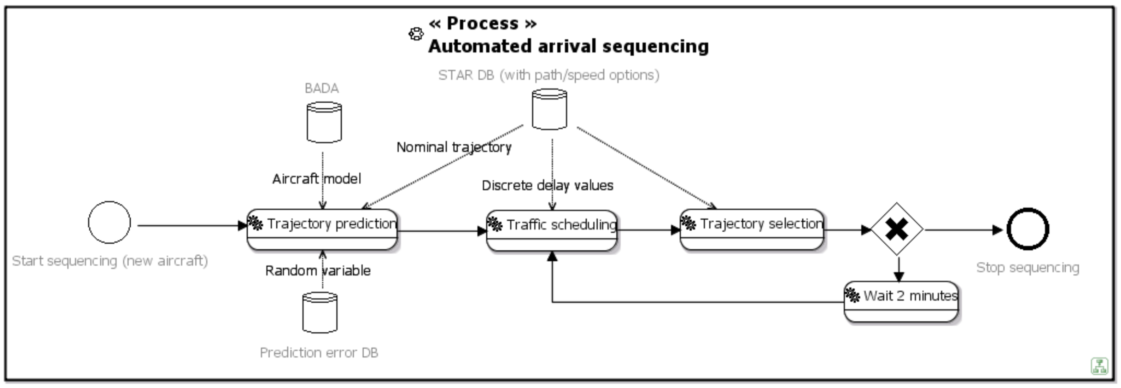


Figure 94 The automated arrival sequencing process

The tasks of the PLC and TCC are of course reorganised to cope with the new AMAN tool. The new collaboration is shown in Figure 95. This time, the TCC and PLC processes are not detailed, as the focus is on the AMAN and its interactions and because the SI* models have already highlighted the main operational changes for the personnel.

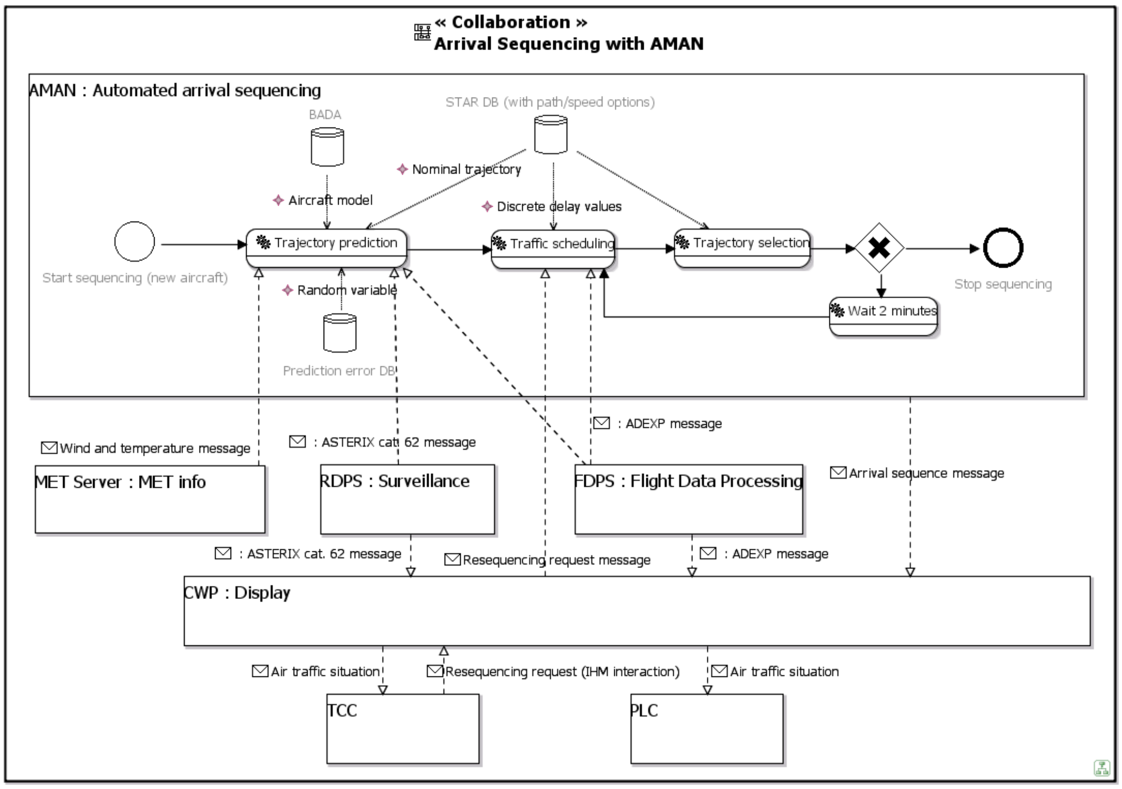
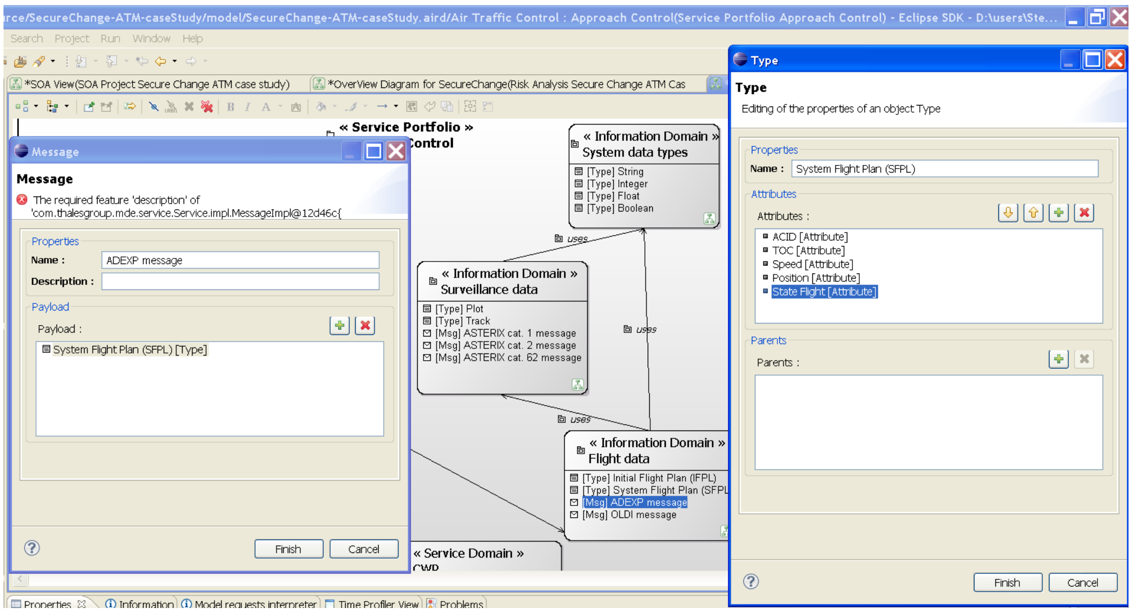


Figure 95 The automated arrival sequencing collaboration

To optimise the traffic scheduling, the AMAN requires having aircraft priorities, in particular with respect to State Flights. This has major consequences on the overall architecture. Let us first make a couple of assumptions on the "as is" system:

- The "State Flight" information was not managed electronically, as only the ATCOs required knowing about this sensitive data. Since the data is to be processed by the AMAN in the "to be" system, the System Flight Plan (SFPL) is now modified to carry this information.
- The HMI provided in the technical supervision room was identical to the controller working position (CWP), giving the technical supervisor an exact replica of what was happening in the control room. To preserve the confidentiality of the "State Flight" information, a new HMI called "Technical Supervision HMI" is now derived from the CWP, which is a downgraded version of the CWP in which confidential data is not shown.

Overall, the AMAN introduction has major impacts on the service architecture and component architecture.



Message

The required feature 'description' of 'com.thalesgroup.mde.service.Service.impl.MessageImpl@12d46c{

Properties

Name : ADEXP message

Description :

Payload

Payload :

System Flight Plan (SFPL) [Type]

Type

Editing of the properties of an object Type

Properties

Name : System Flight Plan (SFPL)

Attributes

Attributes :

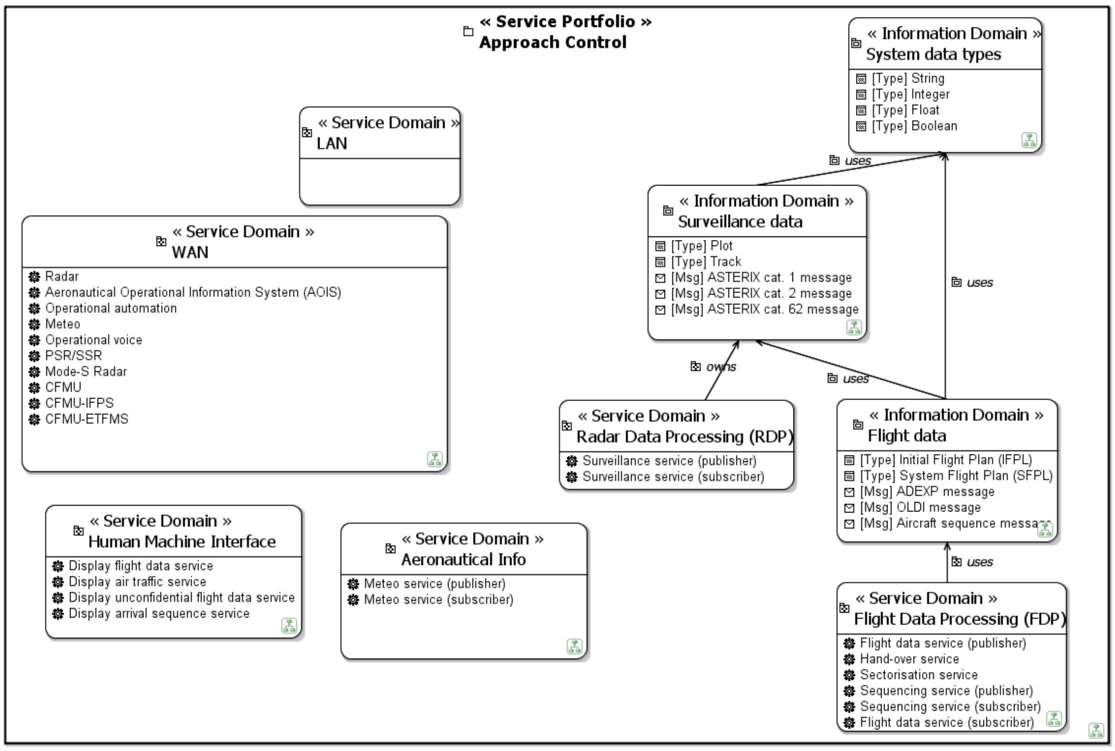
- ACID [Attribute]
- TOC [Attribute]
- Speed [Attribute]
- Position [Attribute]
- State Flight [Attribute]

Parents

Parents :

rce/SecureChange-ATM-caseStudy/model/SecureChange-ATM-caseStudy.aird/Air Traffic Control : Approach Control(Service Portfolio Approach Control) - Eclipse SDK - D:\users\Ste...
Search Project Run Window Help
*SOA View(SOA Project Secure Change ATM case study)
*OverView Diagram for SecureChange(Risk Analysis Secure Change ATM Cas
« Service Portfolio »
Control
« Information Domain »
System data types
[Type] String
[Type] Integer
[Type] Float
[Type] Boolean
uses
« Information Domain »
Surveillance data
[Type] Plot
[Type] Track
[Msg] ASTERIX cat. 1 message
[Msg] ASTERIX cat. 2 message
[Msg] ASTERIX cat. 62 message
uses
uses
« Information Domain »
Flight data
[Type] Initial Flight Plan (IFPL)
[Type] System Flight Plan (SFPL
[Msg] ADEXP message
[Msg] OLDI message
« Service Domain »
CWP
Finish
Cancel
Properties
Information
Model requests interpreter
Time Profiler View
Problems

Figure 96 Adding the “State Flight” information



« Service Portfolio »
Approach Control
« Information Domain »
System data types
[Type] String
[Type] Integer
[Type] Float
[Type] Boolean
« Service Domain »
LAN
uses
« Information Domain »
Surveillance data
[Type] Plot
[Type] Track
[Msg] ASTERIX cat. 1 message
[Msg] ASTERIX cat. 2 message
[Msg] ASTERIX cat. 62 message
« Service Domain »
WAN
Radar
Aeronautical Operational Information System (AOIS)
Operational automation
Meteo
Operational voice
PSR/SSR
Mode-S Radar
CFMU
CFMU-IFPS
CFMU-ETFMS
uses
owns
uses
« Service Domain »
Radar Data Processing (RDP)
Surveillance service (publisher)
Surveillance service (subscriber)
« Information Domain »
Flight data
[Type] Initial Flight Plan (IFPL)
[Type] System Flight Plan (SFPL)
[Msg] ADEXP message
[Msg] OLDI message
[Msg] Aircraft sequence messa
« Service Domain »
Human Machine Interface
Display flight data service
Display air traffic service
Display unconfidential flight data service
Display arrival sequence service
« Service Domain »
Aeronautical Info
Meteo service (publisher)
Meteo service (subscriber)
uses
« Service Domain »
Flight Data Processing (FDP)
Flight data service (publisher)
Hand-over service
Sectorisation service
Sequencing service (publisher)
Sequencing service (subscriber)
Flight data service (subscriber)

Figure 97 Approach control service portfolio, after AMAN introduction

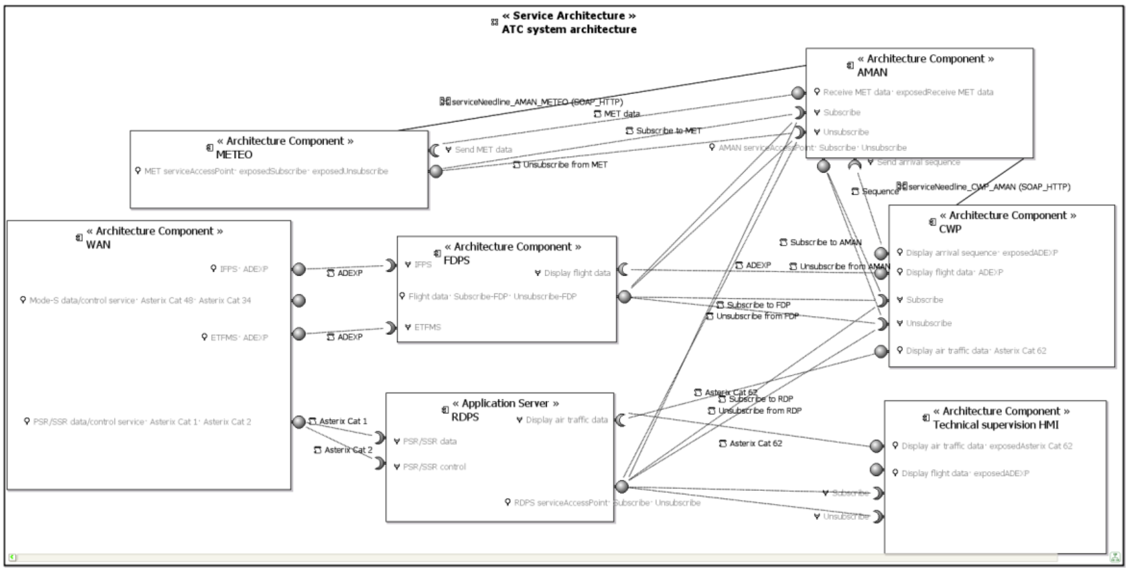
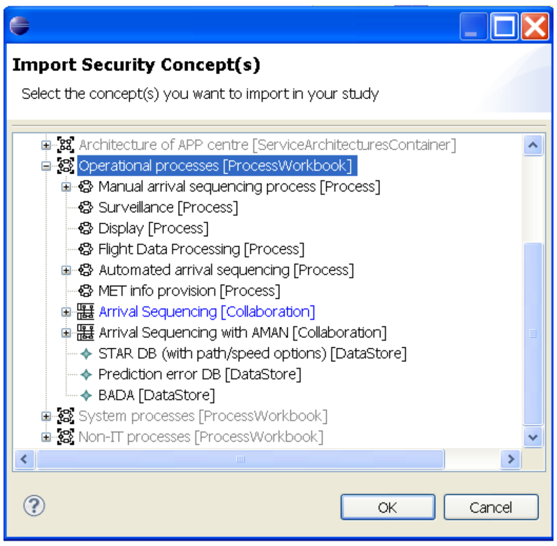

« Service Architecture »
ATC system architecture
« Architecture Component »
AMAN
Receive MET data · exposedReceive MET data
Subscribe
Unsubscribe
AMAN serviceAccessPoint · Subscribe · Unsubscribe
Send arrival sequence
serviceNeedline_AMAN_METEO (SOAP_HTTP)
MET data
Subscribe to MET
« Architecture Component »
METEO
MET serviceAccessPoint · exposedSubscribe · exposedUnsubscribe
Send MET data
Unsubscribe from MET
Sequence
serviceNeedline_CWP_AMAN (SOAP_HTTP)
« Architecture Component »
CWP
Display arrival sequence · exposedADEXP
Display flight data · ADEXP
Subscribe
Unsubscribe
Display air traffic data · Asterix Cat 62
Subscribe to AMAN
Unsubscribe from AMAN
ADEXP
« Architecture Component »
WAN
IFPS · ADEXP
Mode-S data/control service · Asterix Cat 48 · Asterix Cat 34
ETFMS · ADEXP
PSR/SSR data/control service · Asterix Cat 1 · Asterix Cat 2
ADEXP
ADEXP
« Architecture Component »
FDPS
IFPS
Display flight data
Flight data · Subscribe-FDP · Unsubscribe-FDP
ETFMS
Subscribe to FDP
Unsubscribe from FDP
Asterix Cat 1
Asterix Cat 2
« Application Server »
RDPS
Display air traffic data
PSR/SSR data
PSR/SSR control
RDPS serviceAccessPoint · Subscribe · Unsubscribe
Asterix Cat 62
Subscribe to RDP
Unsubscribe from RDP
Asterix Cat 62
« Architecture Component »
Technical supervision HMI
Display air traffic data · exposedAsterix Cat 62
Display flight data · exposedADEXP
Subscribe
Unsubscribe

Figure 98 ATC system architecture after introduction of AMAN

System-level risk assessment using Rinforzando

When starting the second iteration of the risk assessment study, one of the very first steps is to use the "Essential Element(s)" import tool to scan the mainstream system engineering model and see what is new since the last risk assessment study. The import security concept window for primary assets after the introduction of the AMAN in the mainstream system engineering tool, it can be seen that:

- The "Arrival Sequencing" collaboration has already been selected as a primary asset in the previous risk assessment study (because its name is written in blue), and,
- A new "Arrival Sequencing with AMAN" collaboration has been introduced, which has not yet been selected as a primary asset in the current risk assessment study (because its name is written in black).



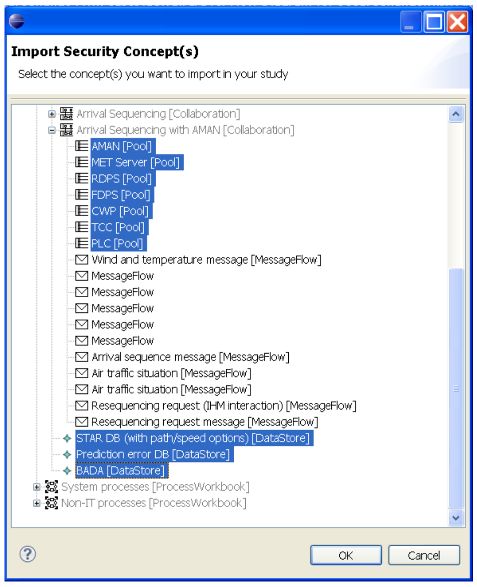
Import Security Concept(s)

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

```
⊞ Architecture of APP centre [ServiceArchitecturesContainer]
⊟ Operational processes [ProcessWorkbook]
    ⊞ Manual arrival sequencing process [Process]
      Surveillance [Process]
      Display [Process]
      Flight Data Processing [Process]
    ⊞ Automated arrival sequencing [Process]
      MET info provision [Process]
    ⊞ Arrival Sequencing [Collaboration]
    ⊞ Arrival Sequencing with AMAN [Collaboration]
      STAR DB (with path/speed options) [DataStore]
      Prediction error DB [DataStore]
      BADA [DataStore]
⊞ System processes [ProcessWorkbook]
⊞ Non-IT processes [ProcessWorkbook]
```

Figure 99 The import security concept window

First, the supporting assets are identified. All the "old" arrival sequencing collaboration supporting assets are also supporting assets of the "new" arrival sequencing with AMAN collaboration, however they are not quite de same. The new AMAN subsumes modified equipments (e.g. surveillance and flight data systems), but also controllers specifically trained for the use of the AMAN. Therefore, even thought a TCC and PLC may be mentioned in both collaborations under the same names, they do not refer to the same "objects". This can clearly be seen in Figure 100, the import security concept window for supporting assets after the introduction of the AMAN in the mainstream system engineering tool, in which the supporting assets can now be selected as sub-elements of the collaboration, and not as independent elements. In addition, there are many more supporting assets, e.g. the AMAN itself, but also a meteorological data server, a database for the aircraft performance model (BADA) – The Base of Aircraft Data (BADA) is a database being maintained and developed by the EUROCONTROL Validation Infrastructure Centre of Expertise located at EUROCONTROL Experimental Centre (EEC) in Brétigny-sur-Orge, France – a prediction error database based on statistical data.


Import Security Concept(s)

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

- Arrival Sequencing [Collaboration]
- Arrival Sequencing with AMAN [Collaboration]
  - AMAN [Pool]
  - MET Server [Pool]
  - RDPS [Pool]
  - FDPS [Pool]
  - CWP [Pool]
  - TCC [Pool]
  - PLC [Pool]
  - Wind and temperature message [MessageFlow]
  - MessageFlow
  - MessageFlow
  - MessageFlow
  - MessageFlow
  - MessageFlow
  - Arrival sequence message [MessageFlow]
  - Air traffic situation [MessageFlow]
  - Air traffic situation [MessageFlow]
  - Resequencing request (IHM interaction) [MessageFlow]
  - Resequencing request message [MessageFlow]
  - STAR DB (with path/speed options) [DataStore]
  - Prediction error DB [DataStore]
  - BADA [DataStore]
- System processes [ProcessWorkbook]
- Non-IT processes [ProcessWorkbook]

Figure 100 The import security concept window

Unlike the supporting assets, the feared events can be shared between the two versions of the collaboration. Here, it is the case for the "Inefficient runway usage". The related risk, which is different from the former, is still considered as negligible. During the previous "acquisition" steps, an issue was raised related to the confidentiality of the "State Flight" information. During the design, a solution was proposed with the "Technical Supervision HMI". However, to limit the costs, the messaging was unchanged, i.e. the confidential data is still available from the network, in particular the wide area network (WAN).

Figure 101 shows a threat scenario in which a technical engineering sniffs the WAN in order to retreive the confidential "State Flight" information. Alone, this threat scenario is not critical. However, if this technical engineering provides this confidential information to a malevolent external actor (e.g. terrorist), then the outcome might be very severe. The internal actors are somehow trusted, due to a necessary accreditation process before being recruited. However, they can be fooled, or bribed, to deliver confidential data to malevolent external actors.

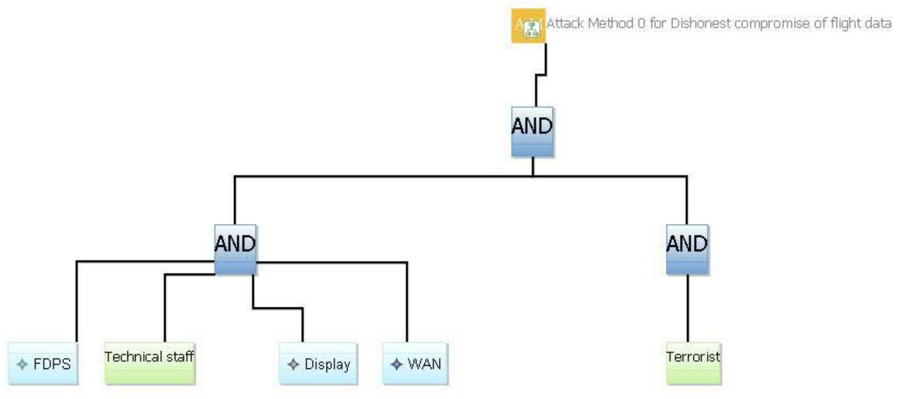
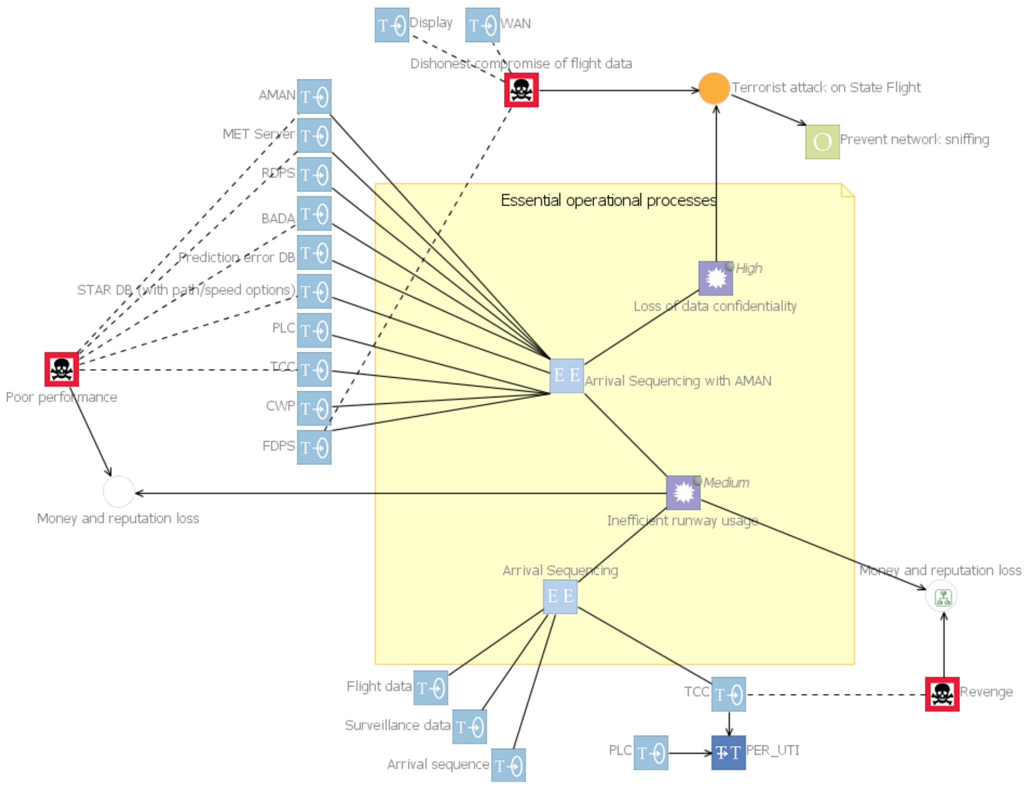


Figure 101 Attack method for dishonest compromise of flight data

The overall risk assessment for the new arrival sequencing with the AMAN is shown in Figure 102. The risk "Terrorist attack on State Flight" is retained (orange colour), and therefore a security objective is defined for that risk. The proposed security objective is to prevent WAN sniffing by technical staff.



Display
WAN
Dishonest compromise of flight data
Terrorist attack on State Flight
Prevent network sniffing
AMAN
MET Server
RDPS
BADA
Prediction error DB
STAR DB (with path/speed options)
PLC
TCC
CWP
FDPS
Essential operational processes
High
Loss of data confidentiality
Arrival Sequencing with AMAN
Poor performance
Money and reputation loss
Medium
Inefficient runway usage
Arrival Sequencing
Money and reputation loss
Flight data
Surveillance data
Arrival sequence
TCC
Revenge
PLC
PER_UTI

Figure 102 Overall risk assessment

This security objective will be provided to the mainstream system engineering team. Multiple design solutions might be envisaged to cover that security objective, e.g. crypt the confidential data, keep confidential data only on the LAN and not sent it to the WAN, etc. This creates a loop in the engineering process. A new risk assessment will need to be performed after the mainstream system engineering team has devised a suitable security control for that risk.

F. HOMES Validation Plan

WP	Artefact	Contact	Version	Date of Availability	Starting Date	Action	Description
WP2	**SeAAS implementation**	UIB	n./a.	Already available	Expected: first half of May	Validation test of the integrated SeAAS solution	Workshop to check the SeAAS in the actual HOMES prototype, once available. The validation will be carried out by TID with some instructions from UIB (basic test plan). Physical meeting is not required.
	Change Patterns methodology and tools	KUL	n./a.	Methodology already available. Tool available by March 1st	2nd half of May	Validation Workshop	Replicate the study already performed by KUL researchers, on a smaller scale, with Telefonica engineers. Evolution scenarios should relate to the HOMES case study.
WP5	**Risk assessment method & Risk modelling language**	SINTEF	n./a.	Already available	n./a.	no planned action	HOMES is a secondary application scenario for these assets. They will be validated in ATM and there is no real need to perform another validation actio in HOMES since there are no relevant differences justifying it. WP5 artifacts are rahter use case-independent.
WP6	**SxC technique for OSGi bundles**	UNITN	Alpha	May-June 2011	May-June	First Validation workshop	Joint workshop between UNITN (technique delvelopers) and TID (technique users) with the goal of teach users to use the technique. TID shall validate the technique from different points of view, as reflected in the validation criteria. Physical meeting is not mandatory but some instructions from UNITN to follow the proper steps by TID staff.
			Beta	October 2011	October - November	Second Validation workshop (if needed)	Same as the first one but with the beta version. I will be carried out just if really needed
	VeriFast: an off-device modular program verifier	KUL	v10.6	Already available	After the 2nd review	Validation test of VeriFast tool	TID to receive the tool and carry out a full analysis of the HOMES security module (PEP). TID shall validate the technique from different points of view, as reflected in the validation criteria. The workshop may not be presential but just prepared by KUL and conducted by TID.

WP7	**TTS language and methodology**	UIB	n./a.	Already available	n./a.	Training and Validation session	UIB shall provide learning material to TID to let them get it touch wiht TTS language and methodology in a practical way. Also, a basic trial script is needed to carry out the validation tests

G. HOMES/WP2 Change Patterns

In deliverable "D2.1 - An architectural blueprint and a software development process for security-critical lifelong systems", an approach is presented to support the architect of a system with preparing the architecture for certain foreseen classes of changes, such that these changes can later be applied with minimal impact. The approach makes use of so-called `change patterns', and a catalog of change patterns is included for dealing with evolving trust relationships.

Each change pattern consists of a *change scenario*, expressed using template requirement models of the situation before and after the change that is captured by the pattern. Furthermore, each pattern has one or more *solutions* attached. Each solution consists of an *architectural template*, which has to be instantiated during the preparation of the architecture, and *guidance* to follow when the change described in the change scenario actually occurs. For more information about the patterns, we refer to deliverable D2.1.

A validation exercise of this approach was performed in the context of the HOMES case study. In the exercise, two participants from the industrial project partner independently applied the proposed approach to the HOMES case study.

The goal of the exercise was to evaluate the industrial applicability of the approach, and to identify possible future improvements to the approach. This document describes the setup, execution and results of this exercise.

Design and setup

The exercise consists of four phases. Each of these phases is explained in the following subsections. In summary, during the **study phase** the participants studied the change patterns approach together with the necessary background (i.e. the UML and Si* modelling languages). In the **setup phase**, the necessary material for the exercise was created based on input from the participants and the researchers. During the **execution phase**, the participants autonomously applied the change patterns approach, which consists of preparing the architecture of the case study for two given evolution scenarios, and afterwards evolving the architecture for one of them. Finally, in the **follow-up phase**, the participants responded to a questionnaire and participated in an interview.

Study phase

In this phase, the participants from the industrial partner studied the necessary background information to perform the exercise. The provided study material consisted of (1) a set of slides about the relevant portions of UML (namely component, composite structure and deployment diagrams); (2) a textual tutorial and slides about Si* (and i* on which Si* is based); (3) the textual description about the change patterns concepts and approach, as given in deliverable D2.1, complemented with slides summarizing the approach; and (4) the catalog of change patterns to be used during the execution phase, as given in deliverable D2.1. The material about UML served as a summary, as UML was already known to the participants beforehand. The other material was new for them.

Setup phase
In the setup phase, the industrial partner created a UML description of the case study's architecture. The research partner then defined a corresponding requirements model expressed in Si*. These models define the initial situation on which the exercise is based. A description of these models is included in appendix **Error! Reference source not found.**. All models were shared with the industry partner before the execution phase started, in order to identify and remove any remaining problems.

Based on the models, the research partner also defined the assignment for the exercise, consisting of two evolution scenarios to be implemented during the execution phase. The description of these scenarios is given in the Assignment section.

Execution phase
The execution phase was executed in the form of a video conference. At the start of this phase, a short presentation about the change patterns approach was given to refresh the participants' memory and to answer any remaining questions.

The researchers could continuously monitor the participants using a webcam, and simultaneously track the performed actions using screen sharing software. During the exercise, the researchers created an event log (e.g. noting down questions that were asked, actions that were undertaken or difficulties that were encountered).

Two possible evolution scenarios were provided to the industrial partner, for which they had to prepare the architecture of the case study by applying the change patterns approach and using the catalog. Each of the scenarios corresponded with a specific pattern from the catalog, but this information was not shared with the participants.

Finally, the participants were asked to evolve one of the two scenarios.

Follow-up phase
After the execution phase, the participants were given a questionnaire to rate their experiences with the approach. Later, a follow-up interview was conducted with each of the participants, to clarify their responses to the questionnaire and to provide additional feedback. The questions from the questionnaire together with the answers from the participants are given in section Questionnaire.

Results

The goal of this validation exercise is to explore whether the change patterns approach is suitable for industrial adoption, and to pinpoint the issues that might hinder this. To be industrially applicable, (1) the effort required from software engineers to learn the approach should be acceptable; (2) the methodology to apply the change patterns should be clear and sound; and (3) the provided catalog of change patterns should be clear and valuable for the user. In the following subsections, the results of the exercise with respect to these three aspects are discussed.

Learning curve

Both participants did not have any prior knowledge about the change pattern approach or the Si* methodology, but were already familiar with UML. This profile is typical for software engineers working in an industrial context.

The participants reported a study time from 5 to 6 hours, which includes the time to study the change patterns approach and the Si* methodology, the time needed to refresh the relevant parts of UML, and the time spent installing and experimenting with the tool that was used. The participants have acquired the necessary knowledge in an off-line and autonomous manner (with the exception of the short summarizing presentation at the start of the execution phase). Hence, it can be concluded that the total time for learning the approach, including its dependencies, is acceptable for software engineers.

In the follow-up interview, one participant indicated that the training was insufficient: while the concepts were clear after studying the provided documentation, some practical experience with the approach was found necessary in order to apply it correctly.

Methodology

The execution phase of the exercise lasted about 2 hours. In general, both participants were able to successfully apply the approach. For both scenarios, the participants selected the correct pattern from the catalog and correctly instantiated a solution.

In the follow-up interview, it was indicated that most problems that were encountered during the exercise are related to the operation of the graphical editors in the tool. The reported problems with the approach itself are the uncertainty about whether a selected pattern is actually the correct one (although both participants eventually selected the correct pattern) and the difficulty of selecting between the alternative solutions for a single pattern.

Furthermore, one of the participants remarked that relying on a catalog with a predefined set of solutions could prevent out-of-the-box thinking, as the range of solutions that are considered for a problem may be limited to the ones in the catalog. On the other hand, this participant acknowledges, the value of the catalog is not to provide every possible solution, but to remind the reader of solutions that have proven to work in the past. Therefore, each solution should be carefully examined in the context of the system before applying it, and it may be necessary to come up with a solution that is not in the catalog.

Overall, the answers to the questionnaire (in particular questions B.2 and B.5) show that both participants found the change patterns approach useful and think it can be successful in an industrial context.

Pattern catalog

In the follow-up interview, one participant remarked that the catalog provides sufficient support for dealing with changing trust relationships, but it should cover additional types of change (besides changing trust relationships) before industrial adoption would be considered.

Furthermore, to alleviate the difficulty of selecting between the alternative solutions of a pattern, the description of the pattern should include more information about the trade-offs (benefits and drawbacks) for each of the solutions that are proposed. Also, both participants indicated that the distinction between the preparation and evolution part of each solution in the catalog should be made more explicit.

In terms of tool support, the availability of an electronic version of the catalog (integrated in the tool) was suggested by both participants.

Initial situation models

Initial situation description

For this experiment, the main actor (and thus the system of interest) is the home gateway. Three other actors are interacting with the gateway: the client devices (operated by users), the service providers and the operator system.

We focus on a single service provider that offers news feeds to the client devices via the home gateway. We assume that the feed service provides personalized feeds to the clients, depending on the user's interest and preferences. Besides the general news feeds, which are free, the feed service provider also offers feeds that are manually selected and redacted by employees of the feed service. To gain access to these feeds, the client has to pay a monthly fee. Hence, the service provider needs to identify its users in order to perform access control to these feeds (this assumption is only made in the context of this experiment, and is not necessarily part of the prototype implementation).

Si* requirements model

The Si* diagram of the initial situation is given in Figure 103. The client wants to obtain (personalized) news feed entries from the service provider. The execution of this task is delegated to the home gateway, which in turn relies on the feed service provider for the execution.

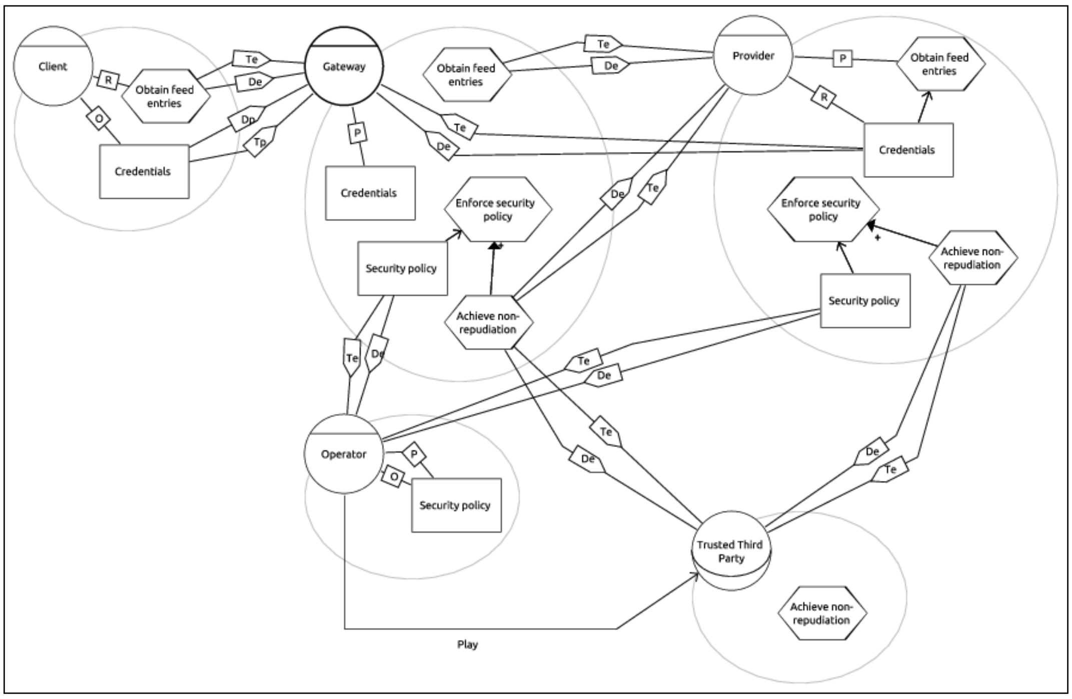


Figure 103 Si* Diagram of the initial situation

To return the requested feed entries, the service provider verifies the client's identity by means of the provided credentials. To avoid repeated entry of the credentials by the user on his client device, the home gateway can cache the credentials. The client, by enabling this functionality, gives permission to the gateway to store and use the credentials for accessing the feed service. Also, the client trusts the gateway in that it will not abuse these credentials.

Furthermore, both the gateway and the service provider are responsible for enforcing the security policies that are stored at the operator. Therefore, they need to obtain the policies from the operator. One of the policies that can be enforced is the execution of a (fair) non-repudiation protocol, using a trusted third party (TTP). The operator plays the role of this TTP. This non-repudiation protocol establishes the trust relationship between the home gateway and service provider regarding the delivery of news feeds.

UML Architecture

The initial architecture of the case study is shown as a UML component and deployment diagram in Figure 104 and Figure 105, respectively.

The gateway contains a feed gateway component (i.e., OSGi bundle) for obtaining the feeds from the service provider and offering them to the client. Both the gateway and the provider contain security-as-a-service (SeAAS) components to enforce the security policies, including component for executing a fair non-repudiation protocol, and a component (CFX) to intercept requests and evaluate them against the security policies. The policies are obtained from a component located at the operator. Additionally, a component to act as a trusted third party for the non-repudiation protocol is deployed at the operator side.

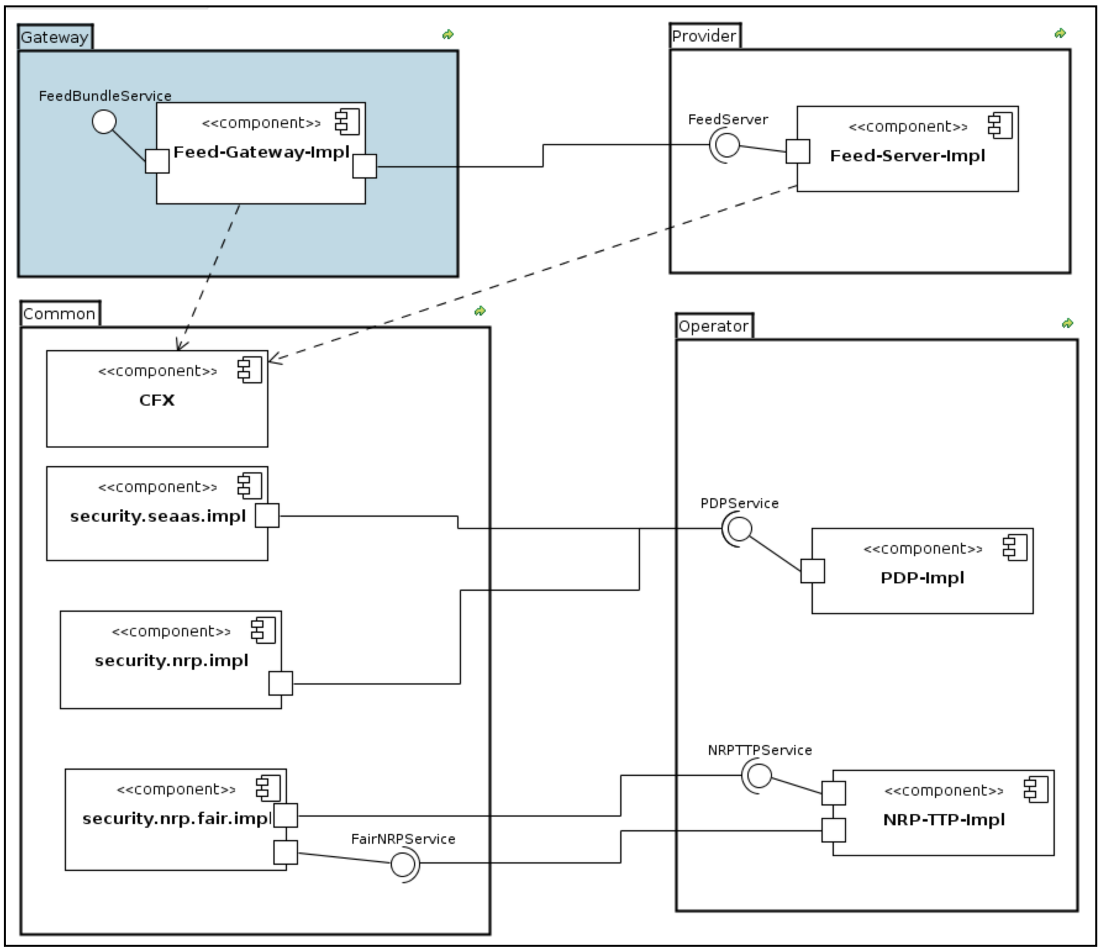


Figure 104 UML Component diagram of the initial situation

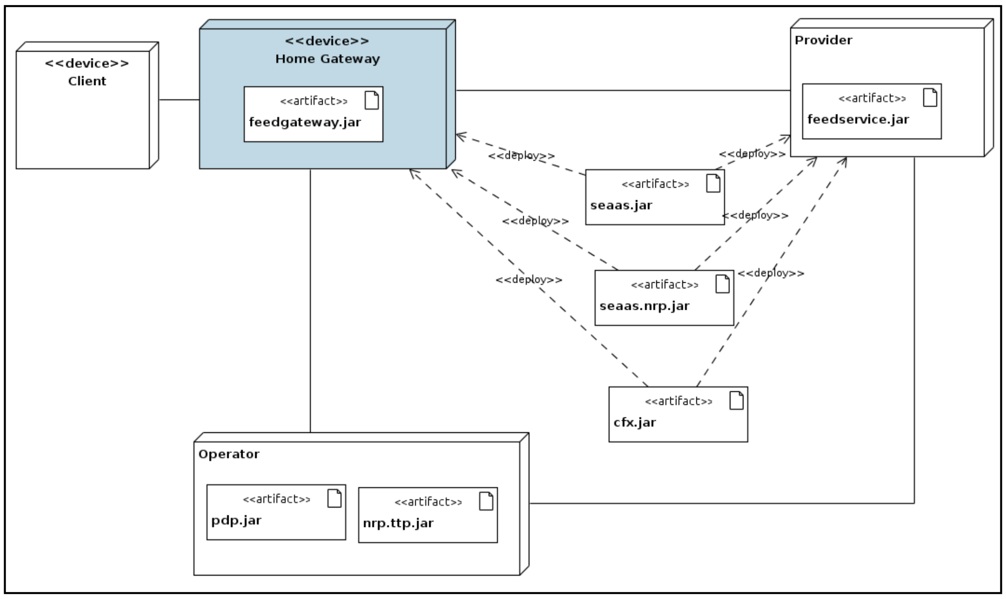


Figure 105 UML Deployment diagram of the initial situation

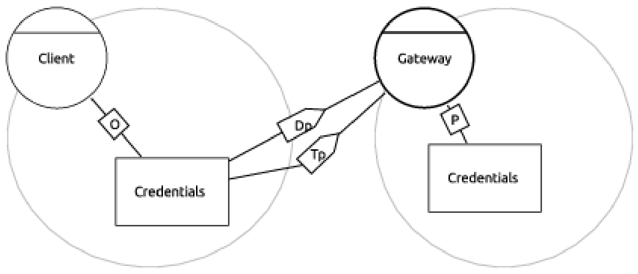
Assignment

Task

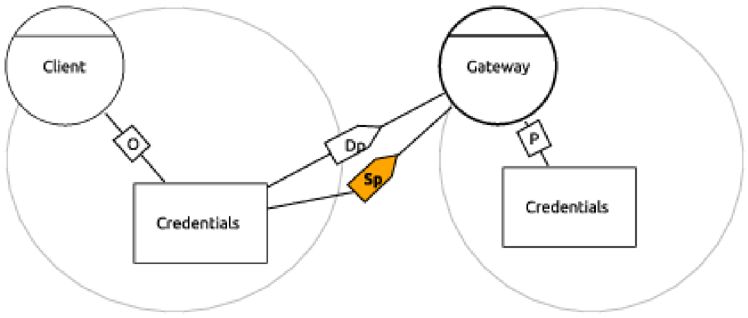
1. Start the exercise from a clean version of the initial UML architecture and Si* requirement models.
2. Select and instantiate the preparation for scenario 1 described below using the change pattern catalog, starting from the initial situation and corresponding models. Update both the UML and Si* models.
3. Select and instantiate the preparation for scenario 2 described below using the change pattern catalog, starting from the result of the previous step. Update both the UML and Si* models.
4. Evolve (implement) the scenario that you are told to implement (i.e., corresponding to that scenario becoming real) on the result of the previous step. Update both the UML and Si* models.

Scenario 1

The users gave permission to the home gateway to cache the credentials for the feed service, as shown in the figure below. They also trust the gateway device to not abuse these credentials.

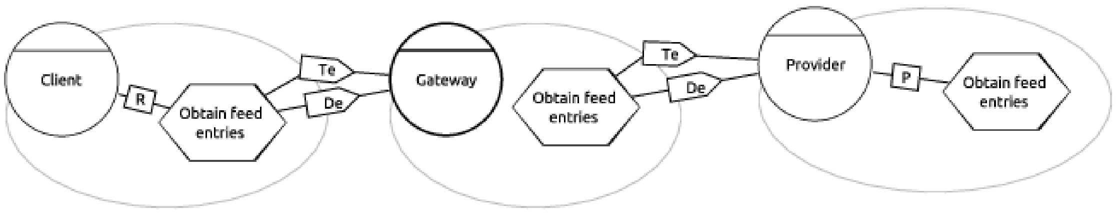


However, in the future, it is conceivable that some users may lose their trust in the gateway device with regard to this permission. For example, a malicious piece of software could use the gateway to obtain access to the paid feeds without the user's consent. This new situation is depicted below.

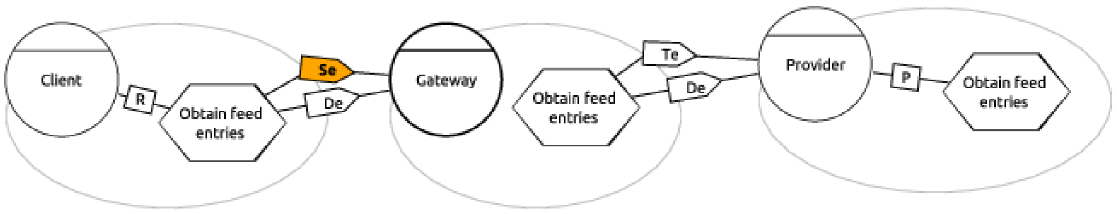


Although the scenario is unlikely in the current setup, it may happen in the future (e.g., when more third parties can install their bundles on the gateway). Therefore, the architecture of the gateway should be prepared such that a low-impact solution to this trust problem can be instantiated in the future.

Scenario 2
Users currently can only use their own gateway for accessing the feed service. Most likely, a user will trust his/her own gateway, as shown below.



However, with mobile devices as clients, it is expected that users will want to access their feeds at all times, also when they are at another place, for example at a friend's house (roaming). In this case, there is no longer a trust relationship with the home gateway for providing the news feed service. For instance, a user may fear that the untrusted gateway will request more (paid) feeds than the user wants. This situation is shown below.



While the roaming scenario is currently not foreseen to be deployed, it is possibly going to be deployed in the future. Therefore, the architecture of the gateway should be prepared such that a low-impact solution to this trust problem can be instantiated in the future.

Questionnaire

Question	Participant 1	Participant 2
A. Understanding and difficulty		
A.1.1. Rate your understanding of the scenario descriptions. – Scenario 1	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear
A.1.2. Rate your understanding of the scenario descriptions. – Scenario 2	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear
A.2.1. Rate the difficulty for each of the scenarios you had to perform. – Scenario 1	○ Very hard ○ Hard ● Average ○ Easy ○ Very easy	○ Very hard ○ Hard ● Average ○ Easy ○ Very easy
A.2.2. Rate the difficulty for each of the scenarios you had to perform. – Scenario 2	○ Very hard ○ Hard ● Average ○ Easy ○ Very easy	○ Very hard ● Hard ○ Average ○ Easy ○ Very easy
A.3. Rate your understanding of the change patterns approach (after studying the provided material and given presentation)	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear	○ Very unclear ○ Unclear ○ Average ● Clear ○ Very clear
B. Approach **For each of the following statements, indicate to what extent you agree or disagree with them.**		
B.1. The change patterns approach puts too many constraints on the designer.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree
B.2. The change patterns approach can be successful in an industrial context.	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree

B.3. The change patterns approach leads to blindly applying patterns without careful consideration.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree
B.4. It is difficult to map the templates provided in the pattern descriptions to the actual system.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree
B.5. Overall, the change patterns approach is useful.	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ○ Agree ● Strongly agree
C. Pattern catalog **For each of the following statements, indicate to what extent you agree or disagree with them.**		
C.1. The change pattern catalog is large enough.	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree
C.2. The change pattern catalog provides alternative solutions that you would not consider otherwise.	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree
C.3. The change pattern catalog is hard to understand.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree
C.4. The description os the change patterns in the catalog is clear and understandable.	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree

C.5. The solutions proposed in the change pattern catalog are trivial.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree
C.6. The distinction between preparation and evolution is clear in the change pattern descriptions.	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree	● Strongly disagree ○ Disagree ○ Neutral ○ Agree ○ Strongly agree
C.7. The solutions in the change pattern catalog adequately solve the evolution need.	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree	○ Strongly disagree ○ Disagree ○ Neutral ● Agree ○ Strongly agree
C.8.1. A better solution exists that is not in the catalog. – for scenario 1	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree
C.8.2. A better solution exists that is not in the catalog. – for scenario 2	○ Strongly disagree ○ Disagree ● Neutral ○ Agree ○ Strongly agree	○ Strongly disagree ● Disagree ○ Neutral ○ Agree ○ Strongly agree

H. POPS Valitation Plan

WP	Artifact	Contact	Version	delivery date by WP	Starting Date for evaluation	Action	Description
4	**UMLseCh** tool + subset of GP lifecycle model	TU-DOR	0.8		**Sep. 2011**	Validation workshop	Technical workshop on the tool and evaluation methodology
				Sep-11	**Oct. 2011**	Tool evaluation	Evaluating the tool on the sample model provided by TU-DOR (of SCP01/02 and card life-cycle). Produce final report.
6 (development-time)	**VeriFast**: an off-device modular program verifier	KUL	**10.6** or higher	august 2011	**Sep. 2011**	Tool evaluation	Evaluating Verifast on GTO applets (JC API support is needed)
6 (on-board)	Information protection techniques (paper work) for JC applets	INR-LIL	N/A	august 2011	**Sep. 2011**	Methodology evaluation	Final report on the methodology
	Proof-of-concept (prototype)		N/A	N/A	**Feb. 2011**	Feasibility workshop	Discussion on the feasibility of on-card implementation: technical details on how to embed the prototype in a GTO platform
			Alpha	may 2011	**Jun. 2011**	1st validation workshop	Feedbacks on the Alpha version
			Beta	Sep-11	**Oct. 2011**	2nd validation workshop	Discussion on the final evaluation of the beta version
			Release candidate	dec 2011	**Jan. 2012**	Tool evaluation	Final evaluation report to be produced on the december delivery
	SxC for smart cards prototype	UNITN	N/A	N/A	**Feb. 2011**	Feasibility workshop	Discussion on the feasibility of on-card implementation: technical details on how to embed the prototype in a GTO platform
			Alpha	May	**Jun. 2011**	1st validation workshop	Feedbacks on the Alpha version

			Beta	Sept	**Oct. 2011**	2nd validation workshop	Discussion on the final evaluation by GTO
			Release candidate	Dec	**Jan. 2012**	Methodolo gy and tool evaluation	Final evaluation report on the tool and the methodology
7	EvoTest lite	INRIA FC and Smarttet sing	1.2	may 2011	**June**	Test model evaluation	Evaluation of the usability, scalability and relevance of the test model
						Tool installation / training	Installation of the tool and training on its usage
					June	Tool evaluation	Preliminary report for the lite version
				Sep-11	**Sep-11**	Tool installation / training	Installation of the additional components
	EvoTest full		1.3		**Oct. 2011**	Tool evaluation	Final evaluation report on the tool and the methodology
4 and 6	WP4 and WP6 integration	TU-DOR INR-LIL	N/A		**Nov. 2011**	Evaluation of effectivene ss	Inconsistencies detection in security policy between model (WP4) and byte-codes (WP6): evaluation is done on the materials provided by two WPs.
6 and 7	WP6 and WP7 integration	INRIA FC and Smarttet sing NR-LIL UNITN			**N/A**	Evaluation of effectivene ss	Done by the evaluation of WP6 and WP7: no addional work is needed
4 and 7	WP4 and WP7 integration	INRIA FC and Smarttet sing TU-DOR	0.8 and 1.3		**Nov. 2011**	Evaluation of effectivene ss	Export of model changes from WP4 to WP7 on life-cycle management: evaluation is done on the materails provided by to WPs.

3 and 7	WP3 and WP7 integration	INRIA FC and Smarttetsing UNITN	3.19 and 1.3	**Jul. 2011**	Evaluation of effectiveness	Evaluating the integration on a portion of GP model: evaluation is done on the materials provided by two WPs.

I. POPS Details on the evaluation activities

The evaluation has been performed essentially by the team involved in the SecureChange project in collaboration with the R&D team of GTO in charge of the security of the open products. Each “evaluator” took the role of the engineer involved in the process : design engineer, validation engineer, etc. The specific tasks that have been conducted for the evaluation are:

For the **WP7**, an adaptation layer has been developed, required for executing the abstract tests generated by the EvoTest tool on Gemalto's product. Gemalto uses a proprietary test environment, EVA.NET for all its testing purposes. The goal of this adaption layer is:

- Translate the sequence of operations described in the abstract XML tests into an effective test script in the language of EVA.NET.
- Provide concrete values for each abstract value in the XML tests.

The adaptation layer is made of the following parts: A **converter**, which takes as input a set of XML files and produces a functional EVA.NET validation campaign; An **adaptation layer**, which enriches the programmatic library of EVA.NET with procedures and constants that instantiate the artefacts from the SecureChange Global Platform UML model; A set of **personalization scripts and procedures**, that allow to set Gemalto's card in the same initial state as the one described in the SecureChange Global Platform UML model.

For the **WP6** and the on-board **verification**, an API layer has been designed and developed to provide the SxC and IFC checkers with the access to the applet being loaded as well as some platform data. This API is necessary because the checkers should not learn information on the platform. The API layer also keeps the Sxc and IFC checkers independent from the platform. On the other hand, the API allows the chekers to benefit from some platform's pre-defined memory buffers and then reduce the RAM and NVM consumption.

The platform's Installer package has been modified to invoke and communicate between the different components of each checkers. Those components include the Java code that stores the card security policy and the C code that performs the verification on the newly loaded applet. The invocation of the checkers is then defined by Java invocation but also native call between Java and C.

The communication between the components is done using a native temporary buffer that has no impact on the overall RAM footprint of the integrated mask.

And last but not the least, several XML files has been developed to compile and link the SxC and IFC checkers source code (both in C and Java) as part of the platform mask. These XML provides instructions on the compiling and linking order, the place reserved for the checkers and how they are identified in the final mask.

For the **WP6** and the **developement-time verification**, the following process was used to perform the applet (phonebook) annotation for the evaluation of the VeriFast tool:

- For each API used by the applet, and not provided by the VeriFast team, we add minimal annotations. These minimal annotations are required to run the VeriFast tool.
- For each method of the applet, we put the annotations needed to prove the absence of NullPointerException. For this we need to specify preconditions and postconditions. Preconditions are conditions that will need to be enforced every time the method is called, they are specified using the keyword @requires. Postconditions areconditions that will need to be enforced every time the method returns, they are specified with the keyword @ensures. The analysis succeeds if VeriFast is able to prove that those pre/post conditions are always enforced in the applet.
- For each loop in the applet (for and while), put the annotations needed to prove the termination of the loop (no infinite loop). For this we need to specify an invariant for every loop. An invariant is a statement that must be true when entering the loop and must remain true on every iteration of the loop. The invariants are specified with the keyword @invariant. The 3 steps above actually have to be completed before running the analysis or an error will be raised by the tool.