SecureChange

Security Engineering for Lifelong Evolvable Systems

A Future and Emerging Technologies Research Project funded by the European Union

White Paper – Year2

Table of Contents

Secure Change White Paper – Year 2

SecureChange is a Future and Emerging Technology Research Project co-funded by the European Union targeting the difficult problem of supporting software evolution while maintaining security. Figure 1 shows our targets.

The aim of the project is to support evolution while maintaining security at all levels of the software development process from requirements engineering down to deployment and configuration.

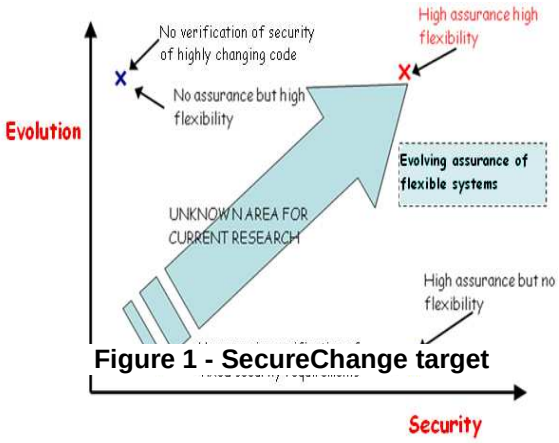

Figure 1 - SecureChange target

Figure 2 shows how the different workpackages of the project address each issues.

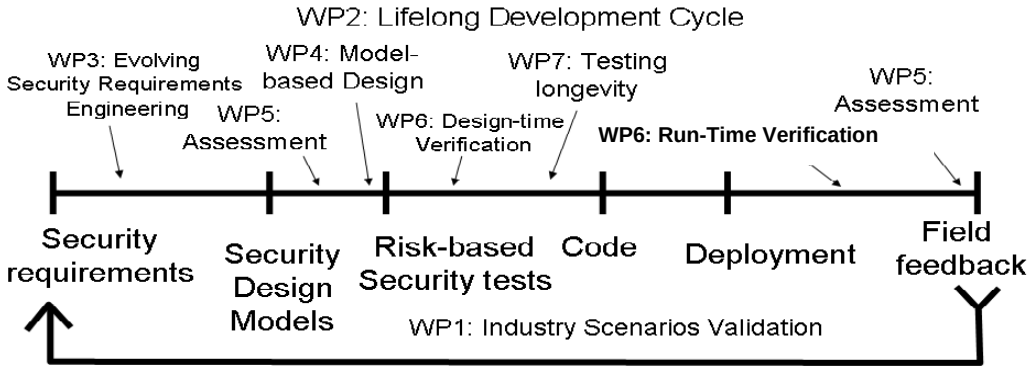

Figure 2 - SecureChange Project Structure

The Challenges of SecureChange

To understand the scope, scale and challenges of maintaining security during software evolution, we exemplify here the results of our empirical study on Mozilla Firefox, one of the most popular browser with millions of users over the world. Our analysis spanned more 5 years of development and 6 mayor versions. In Figure 3 - Version Evolution in Firefox we show a superficial view of the problem.

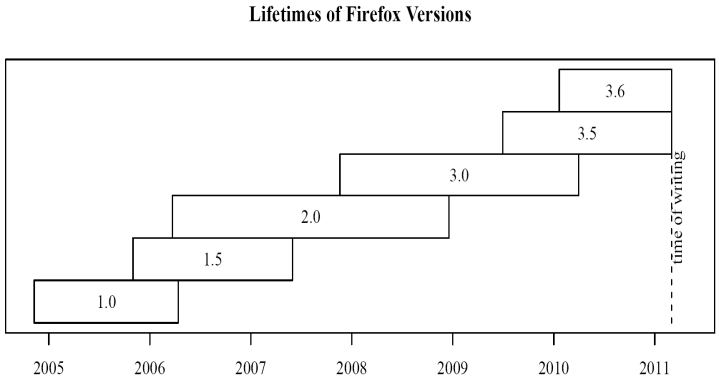

Figure 3 - Version Evolution in Firefox

Here we see the evolution of versions: each version is represented by a box which is born and dies in a very short time span. In this model of the world (i.e. without investigating the actual relationships between the different versions) there would be hardly any need of SecureChange techniques: a quick update to the new version would seem all that is needed to get rid of old bugs and security vulnerabilities.

In Figure 4 we see the real truth if one digs down in the code: each versions is not really new software. Rather it is an old software that evolves to a new versions. Evolution is neither major nor minor but still significant (only 30% is really new). In this setting it is clear that it is important to support requirements engineers in understating how their requirements have changes, to help test engineer to identify which tests are obsolete, which test are new and which tests are untouched, so that you don't need to re-test millions of code lines that have not changed and can effort on what is really new (or even better more risky).

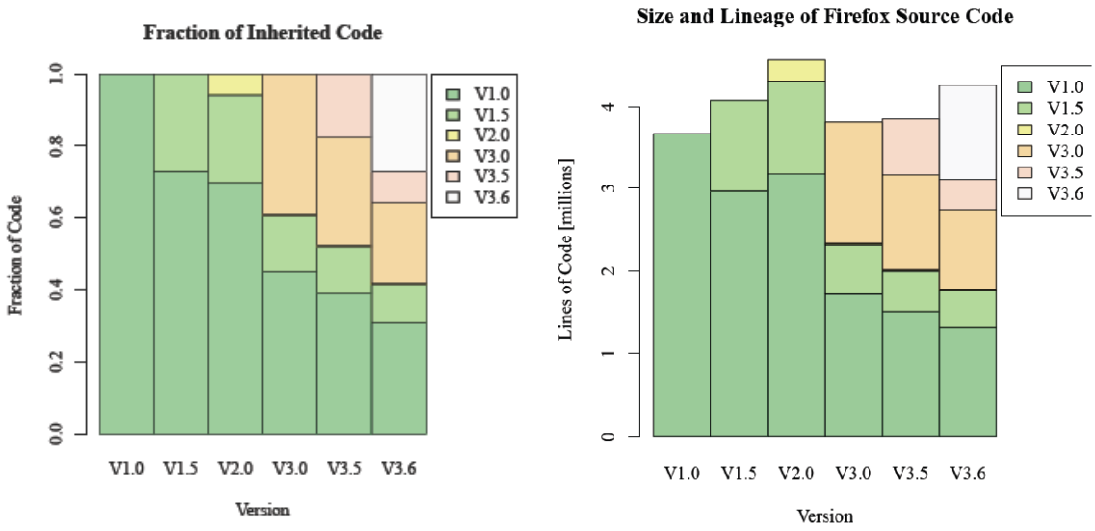

Figure 4 - Code Evolution in Firefox

The SecureChange Light Engineering Process will guide you in the ongoing process of managing software evolution while the requirements engineering methodology and the risk assessment method will provide you with a path to identify the key risks and requirements. The SetGam testing tools will help you in managing the test engineering process in a tight loop with requirements, risks and design models.

Concentrating your effort in the really new thing may not be enough: it is also important to show that the design is right and that its security is verified *and* preserved during evolution: Tabella 1 shows the evolution of vulnerabilities across different versions. The red colour captures foundational vulnerabilities that have been present since the beginning but have only been discovered in later version. Green vulnerabilities are local: they have been discovered and fixed in the version that was current at the time.

As you can immediately see many vulnerabilities are foundational and from this we understand the importance of secure design methods, security testing and verification.

Tabella 1 - Firefox Vulnerabilities (Data July 2010)

first known to apply to	entered for v1.0	v1.5	v2.0	v3.0	v3.5	v3.6
v1.0	79	71	42	97	32	13
v1.5	—	108	104	15	0	0
v2.0	—	—	126	22	0	1
v3.0	—	—	—	67	30	5
v3.5	—	—	—	—	32	41
v3.6	—	—	—	—	—	14

The project in a nutshell

In the course of the first year the project has developed new models, methodologies and processes to guarantee security during software evolution. A large number of academic publications in prestigious journal and magazines (e.g. IEEE Computer) has resulted from this effort. During the second year the SecureChange partners have consolidated these results into a conceptually integrated process and sharpened the project focus to address specific challenges from the industrial case studies of the project.

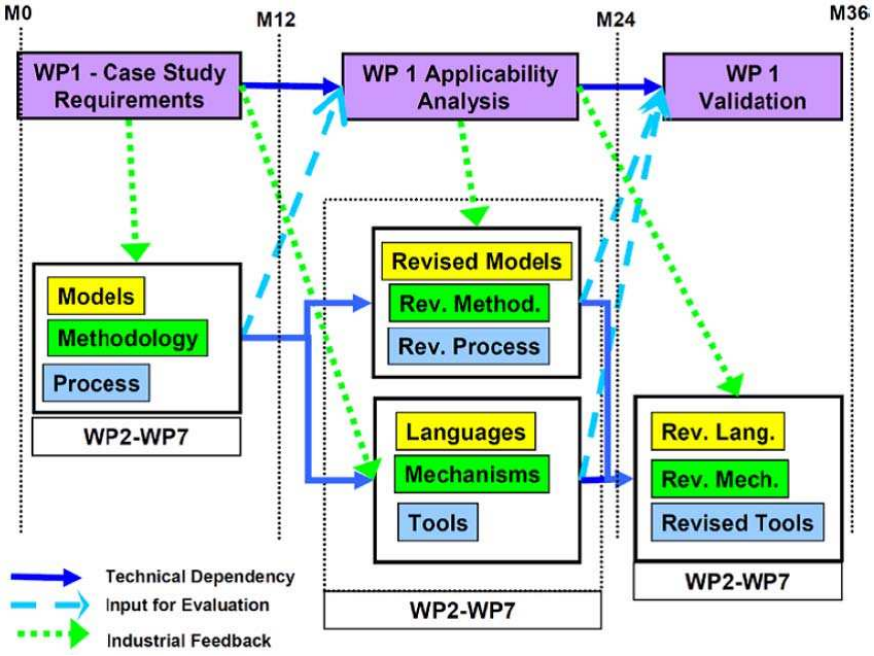

Figure 5- SecureChange Timeline

The case studies are drawn from largely different domains and highlight different change requirements and security properties with the aim of showing that the SecureChange research results do actually work for the widest range possible. They are representative of relevant but not exclusive application domains of SecureChange output. The case studies, named according to their domains, are these ones:

- Air Traffic Management case study (ATM)
- Home Network case study (HOMES)
- Smart Card case study (POPS)

Each case study comes with his specificities of change and impact. Therefore the analysis of the applicability of developed technologies will follow different feasibility and evaluation criteria. In particular, the requirements changes characterising the three case studies fall into three different types: ***Changes in Process***, ***Changes in Configuration***, and ***Changes in Software***.

The ATM case study involves various change requirements due to the introduction of new tools such as the Arrival Manager in the major restructuring of the ATM that will take place with the SESAR initiative. The ATM case study is concerned with how such new tools affect organisational as well as operational aspects. We call this change the ***Changes in Process***.

The HOMES case study is focused on change requirements on policies and critical on software modules providing critical security features of the Home Gateway that is the unique network access point for a wide range of the devices in the home of end customers. HOMES deals mainly with ***Changes in Configuration***.

The POPS case study focuses on an UICC card made of integrated circuit (hardware) and an operating system base on JavaCard and GlobalPlatform specification. This object must be security certified before its issuing, but its life-cycle includes change that could be done in the field. These changes result from adding a new application while preserving the implemented security. Therefore here we are mostly interested in ***Changes in Software***.

Therefore, the solutions provided to solve their change-related security problems are quite different. For example, the *consistency* of the risk analysis modelling artefact for the ATM case study is critical while the performance criteria is more suitable for the embedded running code for POPS or the *availability* criteria for the services for HOMES. In order to achieve few, streamlined, and orchestrated research strands, the technical integration among the WPs have been focussed and driven by the case studies.

The next figures project this overall plan into the ATM case study, the POPS case study and the HOMES case study. They report the main integration links in the project by means of a relevant case study addressed by technical WPs. The meaning of a link is that a clear formal relation between the artefacts developed on the WPs on either side will be identified and the label define the case study on which such relation will be exemplified.

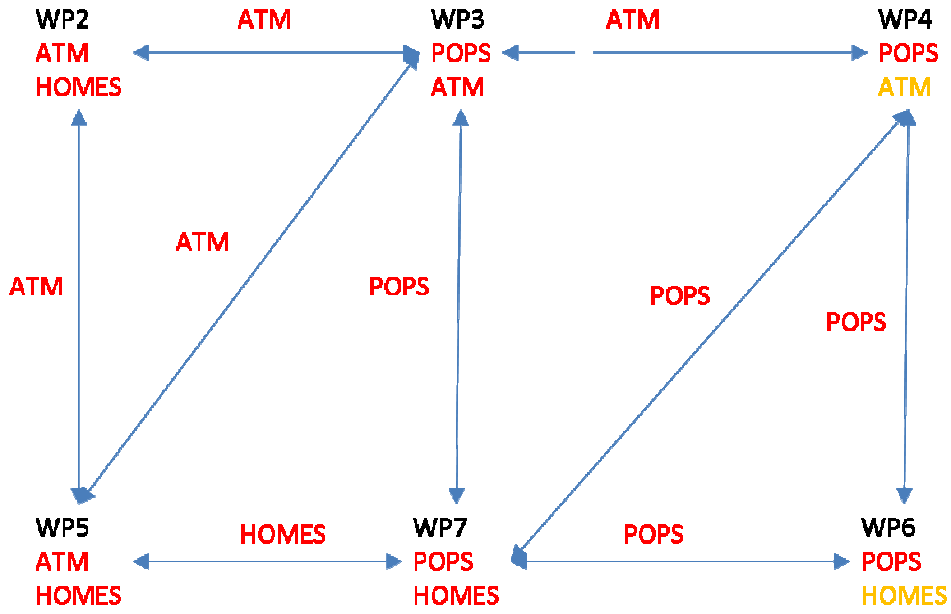

Figure 6 - Integration

The ATM case study is an example of full integration among the early phases of the software design Process. From the overall SecureChange Integrated process we elaborate requirements, risk assessment and design with a proof of concept integration of the activities within the industrial approach of THALES. The process should allow system engineers to design secure systems in which organizational and process level changes are correctly accounted for.

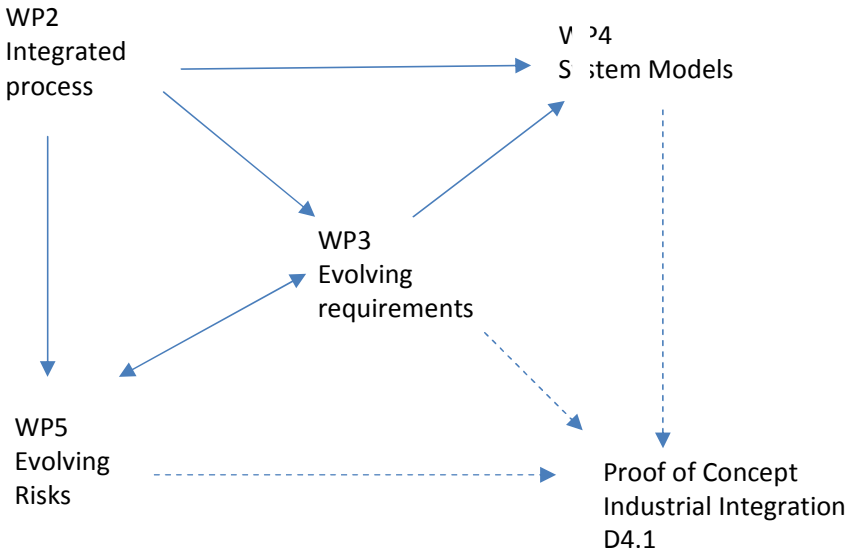

Figure 7 - ATM Case Study

Since the POPS case study focuses on the code level aspects, it mostly involves the workpackages that deal with code and design aspects, while requirements provide the glue between testing and models. Here formally verified design influences and complements both verification and testing aspects, so that one can obtain full--fledged guarantees and where results from test are the feedback provided to requirements models.

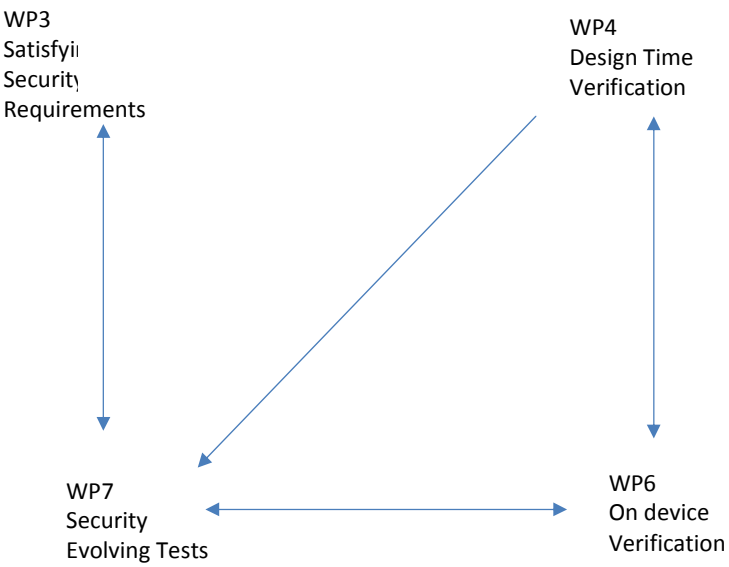

Figure 8 - POPS Case Study

The HOMES case study is an example of vertical integration: we start from requirements evolution and by means of the evolving (security) patterns we derive new architectural constraints that are translated into a concrete Security-As-A-Service deployment solution. Risk consideration is then used to evaluate the key components and testing is finally used to combine the properties.

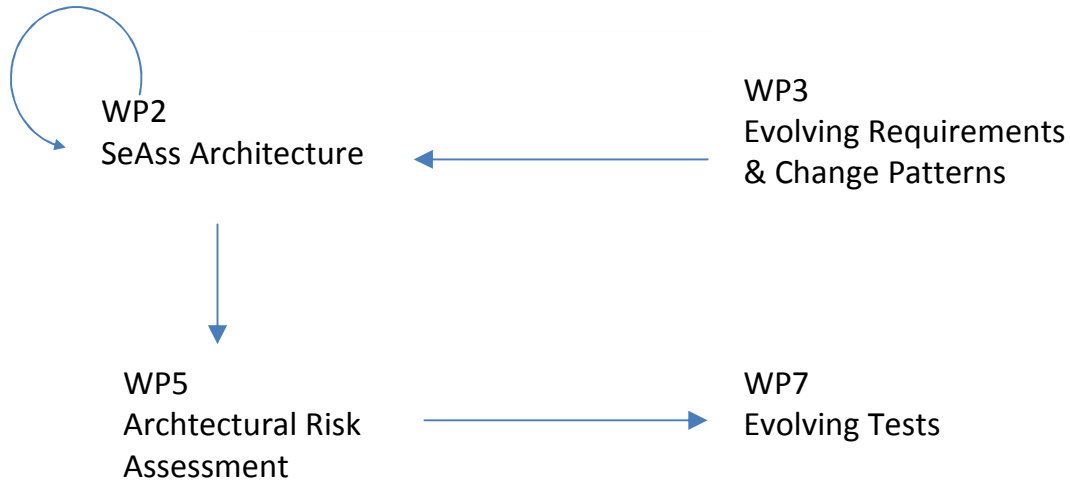

Figure 9 - HOMES Case Study

Applicability and Feasibility Studies

During the second year we have run a number of feasibility studies that have been carried out on each case study of the SecureChange project. The feasibility studies assess the applicability of the artefacts developed in the SecureChange technical WPs to solve the change-related security problems arising within each industry domains.

Our deliverable D1.2 identifies for each case study:

1. a set of focussed security problems;
2. the first feasibility studies that have been carried out between the case study partners, the users of the artefacts and the providers of the research technologies;
3. the technical artefacts developed during the project, that should solve the concrete security problems;
4. the criteria that will be used for the full industrial evaluation of the artefacts during the third year.

The ATM case study focuses on *Organizational Changes*, the main change requirements concern *process level change* and *organizational level change*, for which WP2, WP3, WP4 and WP5 collaborate to preserve *information access, protection and provision* security properties.

For that; WP2 is providing a meta-model to describe the ATM system and related processes with respect to security issues arising after AMAN tool introduction, WP3 is providing model, languages and tools to foster a simpler and more effective collection of requirements that takes into account evolutionary aspects, WP4 is providing models for analyzing and reasoning about security properties in an improved way and WP5 is supporting a complete risk assessment for the Organizational Changes occurring in the ATM case-study.

The HOMES case study focuses on the *Configuration Changes* , the main change requirements are the *update of a security module* and *bundle lifecycle operations*, for which WP2, WP5, WP6 and WP collaborate to preserve the *security expandability, secure extensibility, policy enforcement and resilience to trust changes* security properties.

More precisely, WP2 is delivering an implementation of a Security-as-a-Service engine to make possible application of additional security functionalities along with a tool implementing a methodology to assess the impact of changes into the system; WP6 is providing a tool to verify the actual code of security modules to detect vulnerabilities and a methodology to check the exchange of data between OSGi services; WP7 is designing a test model and test suites to check the impact of changes in advance and finally WP5 is working in collaboration with WP7 in risk modelling of the system with feedback from the testing approach.

The POPS case study focuses on *Code- level Changes* , the main change requirements are the *update of the embedded security software* and the *specification evolution*, for which WP3, WP4, WP6 and WP7 collaborate to preserve *information protection* and *denial 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 critical security properties *from its development to its installation on the card*. For that, an off-card tool checks the application with respect to *denial of service* properties, and then the WP4 verify the *secure communication* between the terminal and the card during the loading. After the loading and before the installation, on-board verification techniques will be provided by the WP6 to check that the loaded application respects the *information flow policy* of the card. The WP7 and WP3 will provide test suites and traceability techniques to check that a new implementation w.r.t an evolution of the specification of the underlying platform respects the information access control properties.

SecureChange Integrated Process

During the first year we have proposed a Living Security Engineering Process as the first security engineering process which is fully driven by change events and change propagation.

In year 2 we have extend our work flexible to provide an integrated process methodology which is abstract enough to connect all new results - of the SecureChange project. We call this light - weight variant of a change driven security engineering process the *Integrated SecureChange Process*. The Integrated SecureChange Process focuses on models as the basic unit of change and provides meta model concepts to instantiate the various methodologies and model types used in the SecureChange project.

By focusing on specific change requirement and security properties we outline in a step-by-step walkthrough how change is handled using principles of change driven security engineering on fragments of the ATM case study our deliverable D2.2.

We also discuss report on the requirements, use cases and technical architecture of the tool support for a state-driven software process and outline in a step-by-step example on fragments of the ATM case study how state based change propagation should be supported.

The final architecture has been implemented (for the HOMES case study) using the last results of this workpackage, the Security—As—A--Serivce approach. Our deliverable D2.2 also illustrates the Integrated SecureChange Process developed by WP2 with the risk assessment methodology developed by WP5 (cf. Figure 1). The integration of the methodologies is described in Section 6.4 and exemplified on the ATM case study. It outlines how a specific methodology and its models can be fit in the overall Integrated SecureChange Process. In particular, the integration is made by instantiating the artefacts of the risk assessment method and the risk models in the Integrated SecureChange Process. The change requirement addressed is “Organizational Level Change” and the properties considered are “Information Protection” and “Information Provision”.

The change requirement *organization level change* with the security *properties information protection and information provision* was addressed.

To keep our deliverable of a manageable size we illustrate how requirements methodology can be fit into the process in another deliverable (D.3.2 in Chapter 9). The integration is based on artefacts and processes and outlines how the SeCMER methodology fits in the overall Integrated SecureChange Process.

The Integrated SecureChange Process was also applied to the HOMES case study, albeit to a lesser extent to show its potential for integrating other technical solutions from the SecureChange project. The change requirement addressed as part of the HOMES case study is the change requirement *bundle lifecycle operations*, with the security properties *policy enforcement* and *security expandability*.

Our deliverable D2.1 instantiates the integrated engineering process between requirements and architecture so a model transformation at the architectural level (called change guidance) to an observed change at the requirements level (called change scenario).

The change at requirements level is modelled as (1) an initial (i.e., before-change), generic requirement template model and (2) a generic change description, by means of which the situation after-change can be derived. The generic requirements template model must be bound to a concrete requirements model via a binding function.

At architectural level, the evolution is described via (1) a generic template architectural model that has to be instantiated in the concrete architecture (via a binding function) in order to support the future evolution, and (2) a generic change guidance that transforms the model according to the desired evolution. This leads to two phases in the approach:

- Preparation: the architectural template is instantiated
- Execution: whenever a change scenario is observed in the requirements model, the architectural model is evolved by executing the guidance

A prototype support tool based on Eclipse has been developed and an initial catalogue of change patterns has been codified according to the above description.

Further, an empirical study is presented that **validates** the promises of the change pattern approach. That is, the study tests the hypotheses that (a) the approach reduces the overall effort for co-evolving requirements and architecture, and (b) the distribution of the effort over the two evolution phases (preparation and execution) is shifted towards the preparation phase.

Evolving Security Requirements

The SecureChange integrated process has a specific word methodology to deal with requirements. Our deliverable D3.2 describes the iterative security methodology for evolving requirements (SeCMER).

Every iteration of the SeCMER process starts with an elicitation stage that analyzes every change request or risk assessment into incremental changes of requirements models. These models are represented using consistent, state of the art modeling languages, such as Tropos and Problem Frames. Through a unified extension of existing Security Goals frameworks (e.g., Secure Tropos and Abuse Frames) it is then possible to represent specifications in such a way so as to reveal vulnerabilities through a systematic argumentation analysis, based on the facts and rules about domain properties.

Using the propositions in the requirements model, the argumentation process analyzes whether the design has exploitable vulnerabilities that might expose valuable assets to malicious attacks. Facts and domain rules that help identify a rebuttal to the security goals are mitigated by introducing induced changes of security properties from the SeCMER conceptual model. In addition to the structured approach to handling change, the SeCMER approach incorporates

incremental transformation of requirement models based on evolution rules. Every evolution rule can be specified formally by events, conditions and actions (ECA). Whenever a change to the requirement model matching some evolution rule(s) is detected, the transformation engine applies specified actions on the requirement model and check whether the existing security goals are still satisfied after the change. If not, the change the passed onto the argumentation process, in order to consider whether the security goal can be restored. When even that is not possible, the security goal will be passed back to the elicitation process where the goal will have to be renegotiated and reformulated.

We illustrate the SeCMER methodology and its iterative process through a concrete example of evolution taken from the ATM domain: the SeCMER models before and after changes of introducing the Arrival Manager tool and the SWIM communication system; the argumentation analysis for the security goal of protecting SWIM information from unauthorized access; and the example of evolution rules to generalize automatable monitoring and adaptation to the triggering and reactive changes to the SeCMER models.

Our Deliverable 3.2 discusses how the SeCMER methodology integrates at conceptual, process and tool level with other SecureChange approaches dealing with the process, architecture and risk. The integration with the design and testing are steps of the engineering process described in the respective WPs. For example, in D4.2 we show how UMLseCh can be used to help with verifying that requirements are actually met by a system and that they are complete with respect to high-level security objectives. The integration is demonstrated with the ATM case study, addressing the organization level change and the security properties of information protection and information provision.

The validation activities will be carried out during the third year of the project by organizing a dedicated workshop with ATM experts. For the purpose of the validation, we will use the *process level change* and the *organizational level change* and the security properties *information protection* and *information access*.

For example we show how requirement model artefacts should be mapped to risk model artefacts and vice versa. The level integration leverages on the conceptual Integrated Process of the requirements elicitation and risk assessment methodologies. The integration is demonstrated in the ATM case study, addressing the organization level change and the security properties of information protection and information provision.

Our deliverable D3.3 presents two different techniques to reason on evolving requirements models, and a semi-automatic approach to requirements change management that is based on incremental graph patterns transformations. We illustrate the reasoning techniques and approach for requirements change management based on the process level and the organizational level change requirements of the ATM case study.

To illustrate the process we have built a CASE toll prototype (D3.4) is an Eclipse-based heterogeneous modelling environment for evolving requirements models that are formulated in different languages appropriate for different work phases and domain expertise.

The main results presented in the demonstrator are the following:

- **Multi-aspect modelling** approach where the security requirements model is composed of several views in different modelling languages, and each work phase can use the facet of the model that is most appropriate to represent their tasks and domain expertise. To cope with the evolving nature of the model, changes made to any of the view models can be **incrementally synchronized** to all other views where appropriate using change-driven transformations.
- **Automated pattern-based analysis** for certain security properties. This analysis is resilient to change and the results are continuously and incrementally kept up-to-date.

- **Interactive and formal argumentation analysis** to support security experts in conducting arguments to verify security properties. Taking up the challenge of evolving models, these argumentation features are complemented by partially automated strategies to cope with changes to the requirements model.

In Year 3, the prototype will demonstrate the feasibility of integration with (1) industrial requirements engineering frameworks and (2) prototype tools of other SecureChange WPs. See Figure 2 - SecureChange Project Structure for a quick overview of the components involved (provided by multiple project partners) and the roadmap for establishing links.

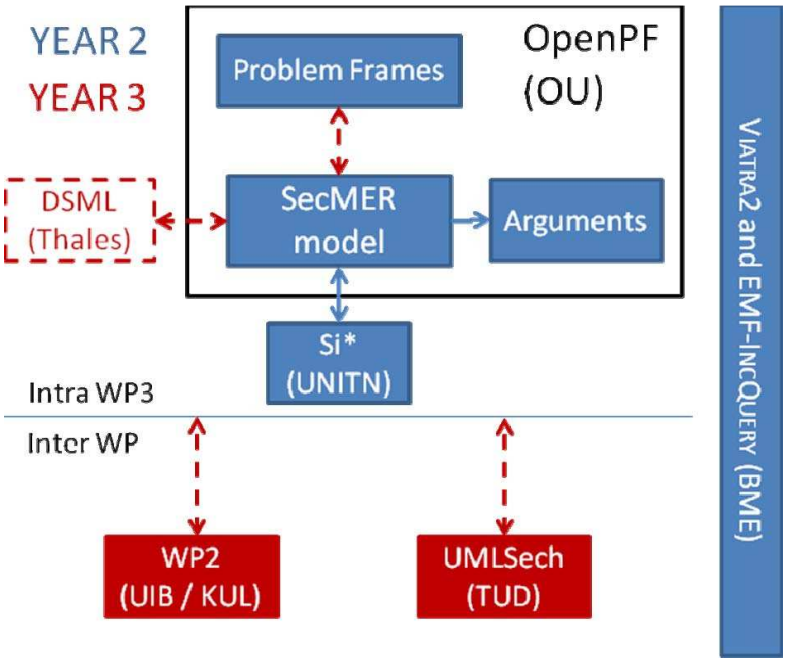

Figure 10 - A roadmap for component integration

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

Our work on requirements also uses the POPS case study, but to a lesser extent to illustrate the integration with testing. The change requirement that is addressed is *specification evolution*, and the security property is *life-cycle consistency*. At the conceptual level, an integration of concepts is presented and it is explained how requirements artefacts should be mapped to test artefacts and vice versa. At the process level, we present the integration of requirements methodology and testing methodology within the SecureChange .The integration is demonstrated based on the specification of evolution change requirements of the POPS case study.

Managing Evolving Risks

Beside Security Requirements the SecureChange - Integrated process also deals specifically with a risk assessment method that meets the methodological needs of assessing changing systems. The guiding principle of the method is that by the occurrence of risk relevant changes, only the parts of the risk picture that may be affected by the changes should be assessed anew. Moreover, in order to properly understand the risks of changing systems as changing risks, the method facilitates the understanding and documentation of the changes to the identified risks. The main technical deliverable artefacts that are presented in our work are the following:

- A risk assessment method for long-life evolving systems.
- A language for the modelling and documentation of changing risks.

• Techniques for tracing changes from target system to risk models.

The method is formally founded by the formalization of the risk modeling language. The syntax of the language for the modelling of changing risks is formally defined, and is underpinned by a formal semantics. The precise reasoning about and analysis of risks are moreover supported by analysis rules that apply to the risk models.

The validation activities for the risk assessment method and the risk modeling language have been initiated by conducting risk assessment case studies. A full ATM risk assessment was conducted during year two, as documented in this deliverable. In particular to show how risk assessment can be used to prioritize the testing activities in WP7. The change requirement addressed is the *organization level change*, and the security properties are *information protection* and *information provision*. WP5 uses also the HOMES case study, but to a lesser extent. It nevertheless serves as part of the validation. The change requirement that is addressed is *bundle lifecycle operations*, and the security properties are *policy enforcement* and *security expandability*. The HOMES risk assessment is documented in this deliverable.

This method is supported by prototype risk assessment tool (D5.4) in order to assist the risk analyst in the tasks and activities that are conducted during the risk assessment process of changing systems. The prototype tool consists of three main editors:

- The diagram editor, which can be used to create and edit diagrams with notation for expressing changing risks.
- The indexing editor, which can be used to index system models that describe the target of analysis.
- The mapping editor, which can be used to create mapping rules between the risk diagrams and the (indexed) target of analysis.

The tool hence supports the task of risk identification and risk documentation, the task of tracing changes from the target system to the risk models, as well as the task of identifying, assessing and documenting changes to the risks.

Evolving and Formally Secure Design

During the first year we proposed a notation that allows one to specify multiple possible evolution paths for UML diagrams. The notation is called UMLseCh and is a further extension of the UMLsec profile. During the second year we have specified a formal foundation for this notation that aims at automatic (re)-verification of security annotated diagrams after evolution (see our deliverable D4.2). To achieve this, we give a more precise definition of the UMLseCh semantics itself, which allows us to pin down what we mean by ‘evolution’ from a model M to an evolved M0. As a result of this, given an UMLseCh diagram we can extract one or more deltas Δi containing the model elements to be added, substituted or deleted from/to the original diagram.

These modifications to an original diagram M have two main consequences: they may alter the consistency of the diagram from the purely UML syntactical point of view, but more importantly they may alter the security properties of M. We discuss the first problem to some degree, but we focus on the latter. For this, we present sound decision procedures for different security properties that allow us to establish whether a given Δ preserves them or not.

Moreover, we have provided a proof-of-concept implementation of these algorithms as plugins for the existing UMLsec Tool Suite. This allows us an automatic verification of UMLseCh annotated models drawn with the ArgoUML tool. Metrics of the efficiency gain of this implementation as opposed to trivial re-verification are presented.

As an application exercise, we model some fragments of the Global Platform (POPS case study) and verify the preservation of selected security properties under evolution. Some of

these fragments are used to illustrate how our formally-based design method can be used to leverage and integrate the approach with other WPs.

Chapter 6 of our deliverable D4.2 illustrates the connection between the modeling and verification techniques developed by WP4 with WP3 (Requirements) based on the ATM Case Study. A risk analysis done with the Thales Security DSML gives high-level security requirements, which are reflected in the System Design and analyzed by means of the UMLseCh approach. The general requirement considered is 'Organizational Level Change' and the properties considered are 'Information Access' and 'Information Protection'. Chapter 4 describes how the result of the verification process at the model level can be used to push constraints to the verification at the code level, based on the POPS case study for a GP specific property and se- cure information flow. The general requirement considered is 'Software update' and the common property is 'Information protection'.

Also using the Global Platform life-cycle (POPS), we illustrates how model-based testing for evolving systems can benefit from formal design techniques. The general requirement considered is 'Specification Evolution' and the common property is 'Life-cycle consistency'.

An Industrial Reality Check for the Early Design Steps

An English proverb runs that "the proof of the pudding is in the eating"- So in order to test the feasibility of integrating the SecureChange solutions for process, risks, requirements and design into an existing industrial engineering process Thales has implemented a proof of concept solution. The focus has been put on the integration of a modelling solution with specialized engineering tooling for security which enables one to define and assess the security requirements which must be implemented on the system. Through this prototype, the full Secure Change chain can be demonstrated with a complete tooling which covers system design, risk analysis and security requirements management.

Since this prototype has been implemented by Thales for industry purpose, the choice of the tooling selected concerns only the technologies needed for the integration of security engineering with the system/software engineering mainstream. For intellectual properties reasons, the prototype does not present Thales engineering workbench. Since the concepts, methods and principles applicable for security and developed in the context of Secure Change are universal, the integration with a design modelling tool can be very well demonstrated on an open source modelling tool supporting UML 2. The choice has been made to use Papyrus UML.

The prototype presents Security DSML, a Domain Specific Modelling Language which captures the security concepts of a risk analysis and enables to annotate a model design. The purpose of Security DSML is to provide tools to conduct a risk analysis when designing a system. The outputs of the risk analysis are the security requirements with a strong rationale related to then. These requirements shall be then exported to a Requirement management COTS such as DOORS T-REK.

Security Verification of Evolving Code

This document summarizes the work performed in Task 6.2 of WP6 of the SecureChange project funded by the European Commission within the Seventh Framework Programme.

The overall objective of SecureChange is the development of verification techniques for evolving systems, with a strong focus on the development time and run--time phases of the software lifecycle. During the first year, we have developed a theory of how to extend a separation-logic based verifier so that it can verify soundly absence of several classes of bugs, even in the presence of unchecked exceptions and dynamic code loading and unloading.

In the second year we have implemented a prototype for these reasoning techniques.

In order to support validation of the verifier on the HOMES and POPS case studies, we have implemented both Java and C front ends for the verifier, and we provide a full implementation of the techniques for dynamic code loading and unloading developed in Task 6.1.

The tool is sufficiently mature that it can verify real Java Card code taken from the POPS case study. For this case study, one of the concerns is robustness (absence of denial-of-service issues) when software updates happen on the card. The prototype tool is being used to verify absence of runtime exceptions and infinite loops in Java Card applets that are loaded on a multi-application smartcard. Several applets have already been verified. A preliminary evaluation shows that verification is efficient, but that annotation overhead is high. In the third year of the project, effort will be spent to reduce this annotation overhead.

Validation of the tool on the HOMES case study is planned for the third year. For the HOMES case study, the tool will verify the *secure extensibility* property for core security module updates of the home gateway. The subset of C that is supported by the prototype tool should be further extended before this validation can start. But initial experiments with simplified models of the code to be verified are promising : they show that the verifier can verify the safety of C code that loads and unloads modules efficiently.

The second stream of our verification activities focuses on the run-time aspects. Our deliverable D6.3 describes compositional techniques to verify evolving security at load-ing time on small embedded devices (multi-application smart cards). Such small devices have restricted memory and usual run-time monitoring techniques cannot be applied on them.

In order to preserve the security of - information exchange in a dynamic environment, where the applications from different stakeholders can evolve and talk to each other, the device should be able to verify the updates autonomously and in a very lightweight fashion. The proposed techniques cover such parts of the information protection requirement as control flow and special type of non-interference.

Our control flow verification provides assurance, that there is no illegal invocation of application services.

Figure 11 below shows the basic idea behind our approach for direct control flown.

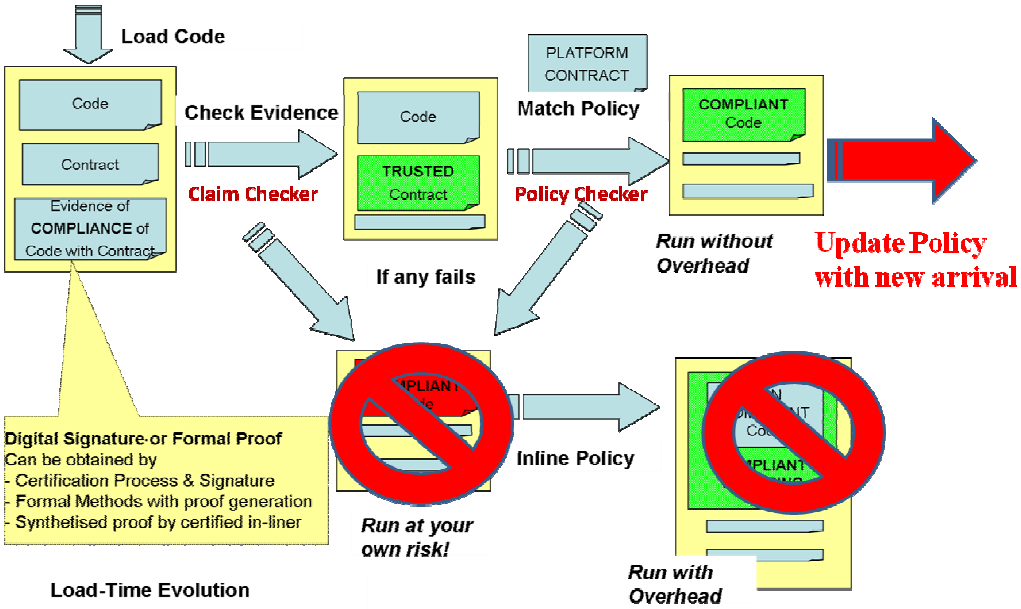

Figure 11- Security by Contract

Our deliverable 6.4 shows the PolicyChecker component for incremental types of updates and discussed how the framework can be integrated with the Java Card system. In Chapter 2 of the current deliverable we finalize the introduction of the Security-by-Contract framework with specifications of the ClaimChecker and the ConfictResolution components, and specification of the PolicyChecker component for the decremental types of changes.

Transitive control flow verification technique extend this framework with the aims of capturing illicit invocations of application methods in case of applications collusion with several solutions for dealing with decremental updates, each solution having a different trade-off between computation overhead and additional system memory required.

Global policy verification technique aims to detect forbidden sequences of methods calls at the system level, i.e. not necessarily only within one or two applications. This approach is inspired from proof-carrying-code (PCC) paradigm: static bytecode analysis conducted off-device generates proofs annotations embedded in the bytecode for easier on-device verification. We show_how to deal efficiently with decremental changes on-device with this model, mainly application removal because updates of the security policy (sets of forbidden sequences of method calls) have an impact on already loaded code stronger than expected and thus requires additional off-device computations but also on-device verifications.

Non-interference verification technique is also a PCC-like approach but whose goal is to detect illicit flows of data between applications clustered in domains. Even if the domain abstraction is strongly inspired from the GlobalPlatform environment, it is generic enough to be applied to any Java-based system.

All the aforementioned techniques support security policy updates. If a system security policy is updated the incremental on-device verification procedures will ensure that all the applications are compliant with new policy. Two approaches are sketched in case some of installed applications are not compliant with new policy. Either the policy update is rejected or the applications conflicting with new policy are made non-selectable.

Depending on the system requirements and stakeholders' needs it is possible to choose the most suitable verification technique. The work remaining for the last year of the project is the implementation of core algorithms for one of the SecureChange project case studies (POPS or HOMES).

We have discussed already the interplay between design and verification in D4.2. The main idea is to verify the same properties at the model level using WP4 techniques and at the code level using WP6 techniques to establish a coherency between (high-level) modeling of applications and their (low-level) implementations. We choose to focus on information protection related properties, and more precisely on the two control flow models and the non-interference model. For each of these models integration has been achieved by the establishment of new specific UMLsec stereotypes. For each WP6 model/WP4 stereotype, we rely on the same input, that is the security policy to be enforced. Furthermore, modifications on the model and the code are both dealt in incremental/decremental way to avoid full re-verification of the model and/or the code.

In addition to verify the same properties at different levels, a coherency report is established between UML models and the code analyzed. Actually, upon successful verification at the model level, some information is extracted from UML to permit additional verifications on the code and thus detect potential incoherencies between application(s) design and implementation.

In this setting testing and verification play a dual role on the information protection property of the POPS case--study. This requirement demands that assets (application data and specific services of security domains) of each stakeholder should be protected from unauthorized access. WP6 (on-device verification) provides techniques to ensure absence of illegal access to information data. WP7 is interested in the access to security domains services. It checks by

testing the absence of possibility to misuse application installation and re-association processes, which grant direct access to security domains services.

In terms of integration we discussed threats for the information protection property and demonstrated that collaboration of two WPs provides protection against these threats. One of the main advantages of the collaboration is possibility for verification to rely on some assumptions about the installation process, because is testing guarantees the confidence in these assumptions. Another benefit is possibility to ensure absence of illegal transitive access to the security domains services, which can be verified by the techniques of WP6.

Testing Evolving Systems

The objective of our work in the second year is to produce a proof-of-concept implementation of our model-based testing techniques for evolving systems. This demonstrator provides a tool-set to ensure the preservation of security properties for long-life evolving systems using software testing. The main results are twofold:

- an approach for testing security properties, based on the use of test schema to which that formalize testing needs. Security properties are covered by a test generation process using a behavioural model of the SUT and associated test schemas.
- an approach for change management by means of model comparison. Our objective is to ensure the important criteria defined in the first year: test repository stability, traceability of changes, impact analysis and ability to automatically structure the test repository into evolution, regression and stagnation test suites.

The demonstrator is based on two elements. On the one hand, it uses the Smartesting test genera-tion technologies (see D7.1 section 4 -WP7 Background), augmented with new algorithms for test generation, an adapted architecture to help handling change analysis and the packaging of a standalone model animator and test generator. On the other hand, the demonstrator is composed of new components:

1. an interpreter of test schemas for testing security properties, a pagina 8,
2. an impact analyzer (that compares two versions of a model in order to identify the changes and produce a change items file),
3. a test classification component (based on the change items file), that uses the model animator and the test generator to produce tests that are organized into test suites,
4. a test publisher that manage the test repository. It keeps tracks of previous tests status and minimizes repository changes.

APPENDIX CASE STUDIES

ATM Case Study: Change Requirements

The ATM case study is concerned with **changes in the operational processes of managing air traffic in Terminal Areas**. Arrival management is a very complex process, involving different actors. Airport actors are private organizations and public authorities with different roles, responsibilities and needs. The subsequent introduction of new tools, i.e., the Queue Managers, and the introduction of a new ATM network for the sharing and management of information, affects the ATM system as a whole at a **process** and **organizational** level.

Process Level Change

ATM procedures need to be updated in order to accommodate the introduction of the AMAN (**A**rrival **MAN**ager). The AMAN is an aircraft arrival sequencing tool helping to manage and better organise the air traffic flow in the approach phase. It is directly linked to the airport organisation and the turnaround process, because arrival sequencing/metering is important for airline operational control and airport operations (e.g., ground handlers, catering services, airlines, security and health authorities, etc.) in order to organise the ground services efficiently.

The introduction of the AMAN requires new operational procedures and functions (as described in the deliverable D1.1). Such new procedures and functions are supported by a new information management system for the whole ATM, an IP based data transport network that will replace the current point to point communication systems with a ground/ground data sharing network which connects all the principal actors involved in the Airports Management and the Area Control Centers.

Goal: The resulting ATM system (with the AMAN and the communication network introduction) needs to comply with suitable security properties, which prevent from corruption, accidental or intentional loss of data and guarantee the integrity and confidentiality of the aircraft sensible data against malicious attacks or intrusions.

Organizational Level Change

The introduction of the AMAN affects Controller Working Positions (CWPs) as well as the Area Control Center (ACC) environment as a whole. The main foreseen changes (as described in the deliverable D1.1) in the ACC from an operational and organizational point of view are the automation of tasks (i.e. the usage of the AMAN for the computation of the Arrival Sequence) that in advance were carried out by Air Traffic Controllers (ATCOs), a major involvement of the ATCOs of the upstream Sectors in the management of the inbound traffic.

These changes will also require the redefinition of the Coordinator of the Arrival Sequence Role, who will be responsible for monitoring and modifying the sequences generated by the AMAN, and providing information and updates to the Sectors.

Goal: The AMAN's interfaces that provide access to different roles and authorizations need to make information available only to authorized personnel or trusted systems.

Security Properties

The following security properties need to be guaranteed at the process and organizational level and will be the focus of the technical WPs.

Information Access. Authorized actors (or systems) must have access to confidential information regarding queue management in the terminal area. Access to information needs to comply with specific role-based access control rules drawn from the operational requirements. **Information Protection.** Unauthorized actors (or systems) are not allowed to access confidential queue management information. **Information Provision.** The provisioning of information regarding queue management sensitive data by specific actors (or systems) must

be guaranteed 24 hours a day, 7 days a week, taking into account the kind of data shared, their confidentiality level and the different actors involved. **Information Need.** Confidential queue management information can be accessed by authorized actors (or systems) only when the information is necessary for operational purposes, which may vary even in real time, due to particular conditions (bad weather, emergency status, etc.)

HOME Case Study: Change Requirements

HOMES is focused on digital home networks where some sensible changes take place from the point of view of the security. We consider some changes, from the large set of changes that anyone may identify in this context, very related to configuration and deployment. Our target is the home gateway as a critical point in the home network.

Core Security Module Update

Home Gateway has some security modules implementing NAC functional components like the PEP. NAC technology and its functional elements are properly described in the deliverable D1.1.. During the lifecycle of the whole system some component updates shall be required for various reasons (better performance, bug fixes, etc.). Updating one of these core security modules in the home gateway is a critical operation and a relevant change. Any attack or failure in this process may be extremely harmful. A possible update on the core security modules could be the extension of information for the security assessment (more information in deliverable D1.1). In this case, the home gateway needs to be updated so that the new security status information is understood and assessed correctly.

Goal: Show that the security properties detailed below are still preserved after an update of a security module.

Bundle Lifecycle operations

A Home Gateway is also a service platform for the home. Customers can install new home services, upgrade or delete existing ones. This type of change is similar to the previous one but here services do not usually implement security functionality. The bundles installed on the home gateway are used for higher level applications. The services may come from third parties and therefore some similar control over this software must exist. Trust relationships among the customer, the service provider, and the third parties may evolve over time. However in some cases security bundles could be deployed (provided by the operator)

Goal: Bundles have to be managed (update, addition, removal) in compliance with the trust relationships and assuring system consistency, i.e. the security properties need to be preserved despite these changes.

Security Properties

Secure extensibility. The home gateway can be extended at run time with additional general software (e.g. bundles) coming from third parties in many cases. Such extensions should be verified to be secure in the sense that they do not introduce unauthorized information leaks or the possibility of denial of service **Policy enforcement.** The Policy Decision Point (PDP) is located in the security domain of the operator. The Policy Enforcement Point (PEP) is a core security module installed on the home gateway. The PEP always enforces policy decisions forwarded by the PDP so that only allowed actions can be carried out.

Resilience to trust changes. The system shall be able to accommodate a change in the trust relationships (among service provider, customers, 3rd parties) with a minimal impact on the software architecture.

Security expandability. System security can be enhanced by taking advantage of the home gateway extension ability (mentioned in the Secure Extensibility property) through the deployment of new security services (e.g., deployment of a non-repudiation service bundle to

ensure that neither service provider nor customer can later deny having sent/received a purchased service). The infrastructure shall be able to efficiently enforce such new requirements with a minimal impact on it.

POPS Case Study: Change Requirements

An USIM card has been certified w.r.t. Common Criteria security certification V3.1. This means that the embedded software on this device ensures a set of properties related to (at least) **confidentiality**, **integrity** and **availability** of its assets (but also non-repudiation, authentication, non-by-passability, etc).

But this “system” during its life cycle will evaluate. The Common Criteria impose that any change that occurs will lead to a re-certification of the card. As the evaluation process is expensive in term of cost and delay, we investigate means, that might be provided by the project, to speed up the re-certification of the card. The means are any kind of artefact that could be used for the evaluation: model, proof, test suites, etc. The objective is then to demonstrate this UICC card ensures those security properties after two realistic scenarios of changes, detailed below.

Specification evolution

An UICC card embeds a component called the card manager, implemented according to GlobalPlatform specifications v2.1. This card component has been extensively verified and tested. The GlobalPlatform specification have been enhanced and extended and v2.2 has been issued. The card manager software component has been updated and extended against this new version. For simplicity reason, we restrict the 2.2 scope to the UICC configuration.

Goal: prove/demonstrate/test that the security properties are still preserved. For that we will concentrate on specific properties detailed below.

Software update

The certified UICC card is deployed in the field. The mobile operator, owner of the card, has a new partner, a bank. He loads a new security *domain (a Java Card application)* on the UICC (card) using an OTA mechanism. This bank will have the delegated management privilege from the Mobile Network Operator to manage its applications in a **confidential** way. In particular, the bank will use its security domain to load an e-purse on the card.

Goal: prove/demonstrate/test that the new application preserves (do not break) the consistency of the existing and implemented security policies. Again the specific properties are detailed below.

Security properties

Denial of service: Any application on the card do not generate a denial—of--service. This means that some robustness properties must be verified by the applets, such as no runtime exception, no infinite loop. Also the memory consumption must be bounded for the durability of the EEPROM and the Flash. For example, bounding the call-stack or detecting loop that updates the persistent memory.

Life-cycle consistency: Any command received by the card must respect the card and applet lifecycle. Its means that any command received in a state s leads to a state s’ and the resulting transition from s to s’ is correct w.r.t. the specifications.

Information protection: The applications on the card must be “isolated” (segregation), that means no illegal access to the data from one application to another. For that several security policies are described and assumed to be implemented on the card, like the JavaCard firewall (access control implemented by the virtual machine) or the security domains of GP. Therefore, some properties must be verified, when an applet is added on the card, like the consistency of

the security domain hierarchy, the non-violation of the information flow policy implemented on the card, etc.

Secure communication: The Secure Channel protocol provides a secure communication between a card and the off-card entity during an application session. It means that the protocol must ensure entity authentication, an entity is an off-card one as the issuer (terminal) or an on-card entity. Each entity proves its authenticity to the other entity. The protocol must ensure also integrity and confidentiality of the transmitted data

Glossary

Acronyms	Definition
ACC	Area Control Center
AID	Application identifier
AMAN	Arrival MANager
APDU	Application Protocol Data Unit
ATC	Air Traffic Control
ATCO	Air Traffic COntroller
ATM	Air Traffic Management
CWP	Controller Working Position
DHCP	Dynamic Host Client Protocol
DMAN	Departure MANager
EMV	Europa MasterCard Visa
FTTP	Fiber To The Premises
ISD	Issuer Security Domain
NAC	Network Access Control
NAT	Network Address Translation
OSGi	Open Service Gateway Initiative.
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PLC	Power Line Communication
PPPOE	Point-to-Point Protocol over Ethernet
QOS	Quality of Service
SCP	Secure Channel Protocol
SIM	Subscriber Identity Module
TMA	TerMinal Area
USIM	Universal Subscriber Identity Module
VPN	Virtual Private Network
WIMAX	Worldwide Interoperability for Microwave Access

Dissemination highlights

The Dissemination activities of the SecureChange project included a presentation of the project at the Project Track of the MODELS 2010 conference in Oslo, Norway to disseminate research questions and first research results to the scientific community. Additional presentation has been done at a number of EU events such as ICT. The project partners delivered roughly 30 additional presentations and published more than 50 papers addressing different parts of the project. In 2010 we had:

- 44 project meetings and 43 events (workshops, conferences, seminars, exhibitions, etc.) organized or attended by SecureChange partners;
- 13 journals publications;
- 35 conferences publications;
- 4 book chapters and 2 reports.

The SecureChange project collaborated with other FET projects by contributing to dedicated workshops and meetings coordinated by the EternalS coordination action. Furthermore, the project organized several internal workshops and meetings to strengthen integration within the project. Industrial partners have identified promising and potentially usable results for exploitation.

The Project Consortium

The consortium is formed by an ideal blend of research institutions, industry and small, research--oriented enterprises:

- Università degli Studi di Trento (UNITN), IT
- Budapest University of Technology and Economics (BME), HU
- Gemalto (GTO) FR
- Institut national de Recherche en Informatique et en Automatique (INR), FR
- Katholieke Universiteit Leuven (KUL), BE
- Smartesting (SMA), FR
- Open University (OU), UK
- Stiftelsen for industiell og teknisk forskning ved Norges Tekniske Hogskole – SINTEF (SIN), NO
- Thales (THA), FR
- Telefonica Investigacion y Desarrollo s.a.u. (TID), ES
- University of Innsbruck (UIB), AT
- Deep Blue s.r.l. (DBL), IT
- Technische Universitat Dortmund (TUD), DE

Further Information

Further information can be obtained by contacting the project coordinator

Prof. Dr-Eng. Fabio Massacci

DISI - Universita' di Trento

via Sommarive 14 - 38123 Trento – Italy

email: Fabio.Massacci@unitn.it

Tel: +39-0461-282086 -- Fax: +39-0461-283987

Or by visiting the project web site**:** **www.securechange.eu**