Secure Engineering and Modelling of a Metering Devices System

Jose Fran. Ruiz
Fraunhofer SIT
Darmstadt, Germany
jose.ruiz.rodriguez@sit.fraunhofer.sit

Marcos Arjona, Antonio Maña
University of Malaga, Computer Science Department
Malaga, Spain
{marcos,amg}@lcc.uma.es

Niklas Carstens
Mixed Mode GmbH
Gräfelfing, Germany
niklas.carstens@mixed-mode.de

Abstract—This paper presents a security engineering process for the modelling of security-sensitive systems using a real use case of metering devices. The process provides a security framework that can be used with other existing processes (such as the agile ones). It helps to develop and model systems bearing in mind their heterogeneity, real-time and dynamic behaviors. Besides, due to the critical nature of some of these systems (nuclear, emergency systems, military, etc.) it provides tools for identifying, working and solving security threats by using the knowledge of domain experts. This is very important because threats, properties, solutions, etc. that are valid or relevant in a given domain, are not applicable to other domains and subject to constant changes. The security requirements of the systems are fulfilled by means of domain-specific security knowledge. These artefacts contain the specific information of a domain (security properties, elements, assumptions, threats, tests, etc.). The solutions are presented as Security Patterns. Each one describes an implementation solution by using one or several Security Building Blocks (SBBs). The security engineering process presented here describes how to model a security-enhanced system model using a library of domain security knowledge. The process has been developed along with a MagicDraw plugin that covers all the possible functionalities, making the work with the models and the security elements very simple and easy for the user.

Keywords-component: security engineering process; security modelling; security solutions; metering system; domain specific tools

I. INTRODUCTION

The design and modelling of security-sensitive systems is a difficult task due to several characteristics these systems have. The most important and critical ones are their heterogeneous, reactive and distributed nature and their real-time functionality. Because of their heterogeneous nature, the work on these systems is done by different elements, artefacts or subsystems that work in several areas, with different objectives and have a distinct functionality than the other ones (different requirements, threats, regulations, etc.). This attribute difficulties the system design and modelling as the security engineer must know all the functionalities, characteristics and security threats of each element. Their reactive nature implies that they must recollect, process and provide an output in a critical time. Usually these systems are controlling a critical scenario (nuclear plants, military, etc.) so these systems must guarantee that their processing and result will be delivered even in critical situations. Along with this characteristic is the real-time constraint. They must guarantee not only a response in adverse scenarios but do it in real-time. Finally, many security-sensitive systems are composed of different artefacts and subsystems and must work with many input or output data sources.

Modelling is one of the most important activities in the process of engineering of these systems because it establishes the foundations for the rest of the engineering activities. Unfortunately, many of the current modelling methodologies do not naturally integrate security in their approaches. Ideally, a security modelling should (i) be able to coherently deal with the different elements and concepts that are related to security (properties, requirements, threats, assurance, etc.), (ii) be useful as a basis for the selection and integration of appropriate security solutions during the design and (iii) have a noninvasive integration in the engineering processes established in the industry.

This paper presents a novel secure modelling and security engineering process developed in the EU SecFutur Project [19]. The main objective of SecFutur is to support the development of dependable and secure systems composed of embedded components. To this aim, SecFutur has developed a new security modelling framework and an associated engineering process that can flexibly integrate security solutions into the system design and can be easily incorporated into existing engineering processes (such as agile processes). The security solutions are provided in terms of SecFutur Patterns (SFP), founded in classical security patterns [20] and Security Building Blocks [18]. The architecture of the SecFutur Modelling Framework is based on the UML metamodelling capabilities and is composed of three different layers that cover different objectives. It is important to clarify that we use the term domain to refer to specific application domains (e.g. MANET, wireless sensor networks, etc.) as opposed to the popular concept of Domain-Specific Language.

The rest of the paper is organized as follows: Section II presents the State of the Art of the security engineering processes; Section III describes our approach; Section IV presents the SecFutur Process Tool (SPT), a tool developed in the SecFutur project that covers all the modelling necessities presented in Section III; Section V describes the metering devices use case where we applied the security engineering process and finally Section VI presents the conclusions.
II. STATE OF THE ART

One of the main concerns of the system engineers is the integration of security in their architectures. The current existing engineering processes for the design of secure systems address the security integration using different types of approaches, although the major challenge of them all is to use a supported methodology to assist developers to include these security mechanisms in early stages of the software development. This shortage of support is usually consequence of the complex security knowledge demanded to the system modeller, where the engineering processes and tools assume a high degree of security expertise from the user about the functional design of the system and security requirements (solutions, assumptions, assurances, etc.).

The Unified Modelling Language (UML) [1] has become the standard notation for the architectural design in the development process. This large reception from industry has resulted in the availability of many frameworks and utilities using this representation to offer methodologies and engineering processes for the system development. These tools provide engineers a great set of options to create the design and the implementation of their systems. Unfortunately, those frameworks have a lack of specific features to integrate security aspects during this modelling process. MagicDraw [2] is a well-known modelling framework that provides a complete software development life cycle with many engineering facilities but none of them have been specifically designed to support security components.

Finding a proper development methodology with security concerns is a real challenge due to the prerequisites about security knowledge required in order to use those mechanisms. The Tropos framework [3] extended in [5] and [7] is an agent-oriented software engineering (AOSE) well suited for high variability design. It is focused on capturing and analyzing security requirements in order to meet the agent capabilities and goals. Tropos is founded on the Model Driven Architecture (MDA) [4] modelling standard Meta Object Facility (MOF) and it creates a design model of the architecture using a metamodel chain of transformations, providing code generation and implementation functionalities. Adaptive security [6] is an extension of the AOSE where systems change and improve their own behavior based on a symptom monitoring and in a defined decision criterion to take the correct actions. Both engineering approaches demonstrate their benefits providing the modelling of the system architecture but they also show their weak support in defining specific security mechanisms, leaving developers all the decisions for including these aspects in the system architecture.

The CORAS approach was presented in [8] and friendly revisited in [9]. It addresses critical systems to conduct a security risk analysis and assessment. CORAS is a UML model-based methodology that integrates several techniques to create a complete report of the system risks, gathering all the necessary documentation and threats diagrams. But the CORAS method only provides the risk analysis and does not provide any help for the later stages of the development process. The system developers receive a great security summary but no support process for integrating the solutions to those threats in their architecture designs. AURUM [11] follows the same principles as CORAS but it provides a framework that applies a risk management methodology based on an ontology to represent the security knowledge with an interactive decision support. Unfortunately, the usability and interest of the industry on this type of approaches fails since it requires training in a specific environment with a new engineering process.

Domain Specific Modelling Languages (DSMLs) focus in the implementation requirements and they are another common approach in security engineering. The development of DSML is in fact a challenging task itself due to the involvement of different levels of abstraction in several domains and the final result is usually a bounded applicability. An example of this complexity method is presented as SecEML in [10]. It was created to check the validity of the security problem formulation and security design decisions but mostly oriented to business process experts and used only in general security engineering.

Another approach in software development is the Test-Driven Development (TDD) [12] with an example of application to Automation Engineering in [13]. This methodology allows a deep understanding of the system under development with an early description of the test cases bases on defined aspects prior to the implementation and parallel with the design. Although promising, this engineering process based on tests can not handle the complexity and variety of the security issues. Besides, the fulfillment of many security requirements requires a complex architecture design hardly to trace from a parallel testing phase.

Other model-driven software engineering processes are the UML profiles such as SecureMDD [14], which provides formal specification for verification and executable code for security critical systems. Applications of this profile (e.g. [22]) show its use with smart cards and cryptographic protocols. Another well know profile is the UMLSec [15], proposed for modelling security properties of computer systems, and several applications over the last decade like [16] and [17]. Those profiles demonstrate their skills in order to represent and model the security requirements and adjust the design process. For instance, in order to validate or exchange requirement analysis models between different sources in the context of a distributed scenario and changing the interoperability of the architecture design accordingly. Unfortunately they show their limitation in that they have to describe any possible element they want to represent in the model and any further use requires an extension of the profile in order to expand the language capabilities.

III. SECURITY ENGINEERING PROCESS

This Section presents the SecFutur Security Engineering Process along with its artefacts and functionality. Although the process covers a lot of different areas and it offers different functionalities (depending on the objective and role of the one using it), we focus in the creation of the security-
enhanced system model using a pre-existing library of security solutions. A more detailed info about the other processes can be found in [21].

The security engineering process main objective is to help developers and engineers in creating security-enhanced systems by using security aspects and elements created by security domain experts. This is done by deploying security properties to the system model of the use case, in order to fulfill all its security requirements. The proposed security engineering process provides the security solutions into a modelling framework that can be easily integrated with existing processes. This modelling framework uses several artefacts that are used for a) defining the structure that represents the security knowledge of any domain and b) gathering the security knowledge of a specific domain. These artefacts are the Core Security Metamodel (CSM) and the Domain Security Metamodel (DSM). The CSM defines the structure of the knowledge. It includes the definition of security properties, threats, elements, roles, assumptions, etc. The CSM has no information and is general (it can be seen as a language that defines the rules and grammar for creating DSMs). The CSM is used by domain security experts to describe and represent the security knowledge of a domain. This action produces a DSM, which is domain specific and contains its inherent knowledge. The DSMs are stored in a repository so they can be used by the system engineer when modelling their security-enhanced system model. Due to the page limit restrictions and that it is not the objective of this paper we do not explain here the structure and description of the CSM and the DSM. They can be consulted in [21].

As commented before, the objective of this paper is the creation of the security-enhanced system model. Following we describe the system model creation process and how we make it secure using our approach.

From the point of view of the system engineer, the creation of the security-enhanced system model is composed of:

- The description of the use case covers the different artefacts that are contained in the scenario (e.g. communications, networks, sensors, elements, etc.) along with their relations (e.g. integrations, communications, links, etc.).
- The security requirements are in our approach the main needs to fulfill in the system. It is responsibility of the system model engineer to identify and describe all the possible security threats of the use case or system she is working on. As an additional feature and benefit of the security engineering process, the DSM (or the DSMs, depending how many the modeller needs) has a list of threats for each security property defined on it. That way, each security property describes all its possible threats, attacks, characteristic of the attackers, objectives, etc. Besides, they have detailed information about tests for each attack, allowing the system model engineer to check the resilience of the system against those threats.
- The DSM contains the security knowledge of a specific domain. Depending on the domains of the use case the system model engineer will use one or more DSMs to cover all the security necessities. The DSMs are contained in a web repository that can be accessed and managed via web or using the MagicDraw plugin developed in the project. They are imported and used by the system engineers as security libraries.

Next we present the security engineering process for creating security-enhanced system models using the artefacts described above.

A. Security-Enhanced System Model Process

The creation of the security-enhanced system model is composed of a serie of phases. It covers from the analysis of the use case to the enhancing of the system model using the security properties defined in a DSM (or several DSMs). Figure 1 shows a diagram of the phases.

As we can see in Figure 1 there are several steps in the security-enhanced system model process. Each one uses as basis the previous one plus other artifacts.

![Security-enhanced system model process](image)

**Figure 1. Security-enhanced system model process**

The security requirements analysis of the target scenario is the first task of the system engineer. The output of this analysis is the elements/artefacts of the use case, relations, actors, methods, communications, external components plus other elements that, although usually are obtained in a normal analysis, are very important and necessary for our process. These elements are: the security requirements of the system, the threats and attacks known by the engineer, the domain (or domains) of the scenario, the security assumptions of the system, the necessary security certifications and all the extra information related to these elements. This analysis is the same any security engineer would perform in the scenario before modelling but focusing in security elements of the system. Next comes the identification and search of the necessary DSM (or DSMs) for the use case phase. Using the information of the previous step and focusing in the domain (or domains) of the use case, the security engineer searches for the DSM that better fits the scenario. If, for example, the scenario is a MANET (Mobile Ad-hoc Network) one, the security engineer will use a
MANET DSM. In case there are more than one she will analyze its characteristics and descriptions in order to select the most appropriate. The search and selection of the DSM is done using the SecFutur Process Tool (SPT) or the DSM web repository. The SPT is explained in Section IV so we only briefly describe here this functionality. The SPT provides a feature for searching DSMs. The security engineer can use it to find the DSM using the domain, keywords (e.g. “network”, “metering”, “wireless”, etc.), the name of a security property (e.g. secure communication or privacy), etc. Once the system engineer locates the DSM she imports it to her system model using the SPT. When it is imported, the SPT shows all the DSM information so it is easy for the user to apply its security properties to her system. The information shown is, among other, the security properties of the DSM (along with a description), the UML element type (e.g. “class”, “attribute”, etc.) and real model elements where they are applied, the users or roles that interact with the security properties, the assumptions done for each security property, the level of the security property (mandatory, optional, etc.), the security pattern that implements each security property, etc. A very important feature also is the displaying of the threats and attacks that targets each security property. That way, if any threat or attack was missed in the security analysis, the user can check in the security property info view the list of threats and update her own list of threats.

Next, after the system engineer has the basic system model of the scenario and has imported a (or many) DSM she sets the security requirements of the system in its corresponding UML element type. For example, she can set the “Confidentiality and Integrity Requirement” in the communication association (UML element) of the class Server. The information of these requirements was obtained in the security analysis, so this phase is very straightforward.

Following, the system engineer applies the security properties of the DSM (or DSMs) in order to fulfill all the security requirements of the system. Although we said in the previous step that we currently set the requirements as info elements the SPT is able to apply automatically security properties to the system model. This is possible because, as we mentioned before, some security properties can be set to mandatory when the domain expert is defining them. These mandatory security properties are related to specific UML elements and real world elements (e.g. storage confidentiality property is mandatory in the personal data attribute of the router class). So, when the security engineer starts to use the security properties the first task she does is let the SPT to apply automatically these mandatory security properties to her system model. This is also very useful because it can help in case the security engineer forgets about those requirements in the security analysis. The idea behind the mandatory security properties is to define and apply the most important security properties that should exist in a scenario belonging to a specific domain. That explains why the security domain expert is the one that defines them in the DSM, because is the one with the higher expertise and knowledge of that domain and knows which security elements are the most important for a minimum level of security-functionality. After applying automatically these mandatory security properties the security engineer uses the rest of the security properties till she fulfills all the security requirements. The security properties provide information and descriptions about them, so the system engineer can know which one she needs for each security requirement.

Figure 2 shows the SPT description of a security property. Besides, the security properties can be applied also in order to fulfill the all the security certification necessities obtained in the initial security analysis.

Finally, after applying all the security properties to the system, the system engineer checks if any of them needs some additional actions such as defining the actor or role related to a security property, verify the assumptions, check the information of the attributes of the security properties (e.g. time limit for a response, maximum number of connections, etc.). Each security property of the system has a SFP that provides its implementation. The security pattern contains a description of the solution and a number of SBBs for its implementation. More information about the SBBs functionality can be found in [18]. Although the SBBs provide the solution, the SFP describes the relations between the SBBs, their communications procedures, objects they use, constraints to check, etc. This part is transparent to the system engineer, as she only has to select a SFP for the solution of each security property. Each SFP contains the functionality and the references (links) to the SBBs they need. This part is done by the security solutions expert and integrated in the security property by means of the domain expert. This way the system engineer can use the knowledge of the security solutions and domain expert without having expertise neither in the security solutions nor the domain. We present in Section V an excerpt of this using the metering use case as example. Once she finishes, she obtains a security-enhanced system model.

![Security DSM](image)

Figure 2. SPT showing the information of an example security property
IV. SecFutur Process Tool & Security Solutions Repository

The SecFutur Process Tool (SPT) is a plugin for MagicDraw created to provide support and cover all the functionalities and necessities of the SecFutur Security Engineering Process. This means that it covers all the aspects of the SecFutur project (such as the creation of the security solutions) but we focus in this paper in its functionality and support for the system engineers to deploy security aspects in the system architecture after a security requirement analysis. That way the security engineers can create a secure system even with low security knowledge. The description of that process was explained in the previous section. This detachment of work is possible thanks to the separation of responsibilities and roles created in the SPT.

MagicDraw is a powerful modelling tool that provides a rich set of options to work with UML models and support developers in the design of complex architectures, offering many features like code generation, business analysis or collaboration support. The awarded path and long stay in the market proves the reliability of this intuitive framework, which is embraced by industry and many companies use it at some stages in their development processes. Using this parent framework the SPT inherits all the functionalities of the MagicDraw multi-platform environment, adding a new set of tools related to the Security Engineering Process.

A. Security Solution Repository

After the analysis, the security engineer next step is to fulfill the security requirements of her system. For instance, the system engineer focuses in one requirement: “Establish a secure connection between Server and Client”. In order to solve this, she needs a security solution and a DSM of the domain. As her company uses the Security Engineering Process, the domain experts of the company have created several security artefacts and solutions and have made them available in a repository. The system engineer can access this repository either using a web application or the SPT. Although they both have similar use, we are going to focus only in the SPT functionalities in this paper. The SPT can be configured to access any repository (the SecFutur repository, a local one, etc.) and search for solutions there. After the system engineer finds the appropriate solution (DSM) she can analyze the data (security properties, assumptions, threats, solutions, etc.) and download it to her framework. The SPT downloads automatically the solutions of each security property of the DSM along with all its required information. The implementation solutions (the SBBs) are referenced as links, so when the modelling ends and the development starts the SPT will translate these “links” to the real elements (if they are only software) or will inform of the necessary hardware for implementing that solution (for example if the solution works in a TPM artefact).

B. Repository Search Engine

The repository interface supports users to locate the best solution expressed in the form of security properties in the DSMs using the search capabilities of the SPT. Every artefact stored in the repository is processed, analyzed and indexed using all the security keywords and descriptions that uniquely identify it. Therefore, the modellers use that information to seek for the most appropriate security property to achieve the security requirement. The search process can be done using keywords, security properties, the name of the DSM creator (being either a user or a company), real name or UML element type (the one that has the security requirement), threats, attacks or the description of the security property (for example the security engineer can search a solution that “helps establish a secure connection”).

Once the system engineer searches the repository using, for example, the keywords “secure communication exchange”, the SPT shows all the coincidences like the “Secure Communication” security property along with all its information and the DSM (or DSMs if there are more than one) where it belongs. The security property description is “Secure Communication ensures that the data exchange is made in a secure way”. This information helps the security engineer to verify if the security property is the one she was looking for. If correct, the user checks the detailed elements of the security property and the DSMs, the type of element where is applicable, the assumption, the threats or the domain (the domain element contains many detailed information about its use and scope). Figure 2 shows the repository interface and how the information is displayed to the user during the searching process.

C. Import Security Knowledge

The system modeller uses the SPT to import the necessary DSM or DSMs. It allows the search and import of several DSMs at the same time. When the system engineer imports the selected property it is loaded in the MagicDraw environment. The SPT offers also functionality for importing only selected security properties of a DSM instead of a complete DSM. This is useful if the system engineer only needs one or two security properties of a specific domain (such as wireless communication) so she does not need to import the complete DSM (which can have several security properties she is not going to use). Figure 3 shows how the selected property “Secure Communication” is selected from the “Security DSM” to be imported.

The combined use of the search and import features in the same dialog allows the modellers to operate with the repository in a more intuitive way, choosing the most proper security elements and switching components of the final list on the run only in one access.

D. System Model Support

The system engineer uses the MagicDraw framework and all its functionalities (create class diagram, relations, elements) to model her architecture, using the design methodology (as we mentioned before our process fits with any other model-driven design process very easily) established in their company. Whether she uses MagicDraw or not, the goal is to obtain a UML system model of the scenario. After the basic modelling of the architecture, the system engineer uses the SPT to enhance the design with the security improvements as we mentioned in the previous subsection. Therefore, this approach enables the engineer to
focus in the architecture, model all the system components and finally gather the list of security requirements to be fulfilled in the scenario, but avoiding at this moment the creation of complex functional elements to provide any security aspect. The absolute integration between the SPT and MagicDraw is a great advantage for the modeller because she does not need to change to another design framework to use the SPT features and take advantage of the security support.

Figure 3. Solution repository interface - search and import

E. Security-enhanced System Model creation

Once the security solutions importing phase ends, the system engineer can check all the security properties of the DSM (or DSMs) she has imported as it can be seen in Figure 2. In case the user needs to activate (or deactivate) any other security property of the DSM she can do it in this dialog by checking or unchecking them. The deactivation means that it will not be shown in the “applicable properties” context menu of MagicDraw but they will still be available in case the user wants to activate it again.

The system engineer applies security properties to the UML elements of the system model in order to fulfill the security requirements. The concrete type of element where a security property can be applied is defined in the DSM. Therefore the system engineer locates in the architecture design the place to apply the security property. For instance the “Secure Communication” property can be used in any “Operation Element” (UML operation element type) and the engineer wants all the operations of a class to be secured. Then, the user selects in the class diagram the element where she wants to apply the security property (in this example a operation) and in the context menu a SecFutur list of options offers all the possible security properties applicable to that type of element. Figure 4 shows this context menu. The security engineer selects the necessary security property and it is deployed automatically in the architecture with all its required elements designed in the DSM. The added elements of a security property are the attacks, threats, required assumptions, the solutions, etc. This information is necessary because it provides the system engineer with knowledge of the security property and how to improve her system, such as making her system resilient to the specific attacks of that security property, the assurances that a security property offers (e.g. “the security property confers your system with the IAAP certification of secure data”).

F. Solution Deployment

Once the enhanced system model is created, the system engineer specifies the security solutions for each property. This is important because the selection of a specific solution for a security requirement may need an extra feature or element that does not exist in the system already. For example it could need a key storage for securing the communications between two elements. The system engineer has to expand then the system model with this information. All the security properties integrated in the model contain references to their solutions in form of SFP. Every security property contains one or more SFPs and each of them includes the solution implementation in the form of SBBs. The SPT offers a view of the SFPs of each security property so the user can compare their differences and characteristics. That way it is easier for the system engineer to select the one that best fits bearing in mind the target security requirement to fulfill.

Once the security solution (the SFP) is selected, the SecFutur context menu displays all the SBBs included in the SFP. When the system engineer chooses one solution the integration process starts. The SPT deploys the SBB in the system model automatically, generating all its defined elements. As we described before, the SPT checks and informs the system engineer about the elements demanded by the SBBs for the integration. For instance if a security property protects an asset in the system model, the SPT needs to know which element from the architecture is the target to protect. Other examples are the context element. These components are required for the implementation phase so they also should be provided by the system engineer or created by the SPT. When the process finishes the implementation is deployed in the architecture and the developers can use this model to continue with the development stage.

Figure 4. SPT Context menu - Applicable properties.

V. USE CASE EXAMPLE

A. General Use Case Description

The scenario presented here is a metering system with legal requirements. It is illustrated in Figure 5. As we can
see, several metering devices, called TSMs (Trusted Sensor Modules) and TSMCs (Trusted Sensor Module Collectors), are deployed. The TSMs measure the current energy consumption of different households, each one having one or more physical sensors for metering, while the TSMCs collect their data. The acquired measurement data is transferred via a local bus from each TSM to its TSMC and in periodic time intervals the consumption data is sent to the operator server by the TSMC using a General Purpose Network.

In this particular example there are two TSMCs deployed. The TSMCs are connected to a General Purpose Network, to which one or more Operator Servers can be connected for final collection of measurement data, as well as an Administrator Server for management of the TSN. In order to access TSMs and TSMCs remotely, a General Purpose Terminal can also be connected to the General Purpose Network. The TSMs and TSMCs shall be seen as functionally but not necessarily physically separate modules. Some of them may be implemented in the same physical device.

The overall overview described here is called the Trusted Sensor Network (TSN). The TSN is an advanced metering infrastructure consisting of several trusted meters, database servers, client applications and communication infrastructure. Its purpose is to measure energy consumption of households and to facilitate the assignment of consumption data to customers for billing purposes. It supports different user roles in a way that reflects the typical organisation of the parties involved in the business, thereby providing intuitive functions for its users, who are able to interact with the components of the system via local or remote interfaces.

Figure 6 shows the high level functional architecture of the TSN. It can be divided into separated parts: Trusted Meter functions, Operator server infrastructure, Administrator server infrastructure and General Purpose Terminal. Each part has its different deployed components and user interactions. A General Purpose Terminal is a device that is able to access the Operator Collector Storage Server (OCSS) that is included in the Operator Server, and the TSM or TSMC connected to the General Purpose Network. The Administrator server infrastructure inherits an Administrator and a Local Client.

This infrastructure offers functions to administrate and run the TSN. The Operator server infrastructure provides the following functions:

- collection and storage of consumption data from the operators customers (measurement database)
- management of customers and users within the operator organisation (user database)
- management of trusted records

On the other side, the Trusted Meter functions part is the most complex and important one of the architecture. It contains the presented TSMs and TSMCs. These devices must use strong security functions in order to assure the authentication, integrity, secure storage, etc. of consumption data and gauging information stored on the devices. As the devices will be deployed in potential hostile environments (e.g. unsecured user households), the connection through the General Purpose Network and the devices must be secured against tempering and unauthorized access. It must be prevented that any legally-relevant or privacy related information can be accessed by malicious users. Successful attacks could lead into reduced electricity bills or even leaked privacy information.
As described in Figure 6, the TSN contains the following actors:

- **Local User**: checks the power consumption of her household using a local interface.
- **Remote User**: interacts with the TSN in order to access her power consumption. She is similar to the local user but using a remote client.
- **The TSN Administrator**: is the responsible of the administration of the TSN. This actor setups, manages and maintains the TSN using an Administrator Server.
- **The Operator**: represents a user that refers to an organisation that utilizes a TSN to collect measurement data (e.g. an energy provider).

### B. Security and Legal Requirements of the Scenario

After describing the scenario we present here its security objectives and requirements. Bear in mind that, due to the size of the use case, the high number of artefacts, users and communications, the real output of the security analysis would produce more pages than the ones we have for the paper. For that reason we describe here only some of the most important ones. This restriction is applied also to the system model of the scenario and the security properties we apply to it. The real system model and solutions diagram cannot fit in one paper.

Therefore, after performing the security analysis these are some of the security requirements of the system:

- **High-level security objectives**:
  - Trustworthiness of the system: all involved parties shall be able to trust that a measurement was performed under certain conditions and according to certain rules.
  - Integrity and Confidentiality of measurement data: all involved parties shall be able to trust that measurement data is authentic and has not been manipulated.

- **Security objectives for the environment**:
  - Availability: the TSN and the measurement data must be highly available.
  - Robust design and optical inspections: the devices shall be mechanically robust and built in a way to make detection of physical tampering possible.

- **Security objectives for the system**:
  - Integrity of transmitted data: the system shall ensure that all data transmitted between components in the TSN are complete, authentic and unique.
  - Confidentiality of transmitted data: the system shall ensure confidentiality of the content of all data being transmitted within the TSN so that only authorised users can see it.
  - Confidentiality of stored data: the system shall restrict user read access to data stored on a component according to the policy “User data access policies”.
  - Data flow control: the system shall restrict data flow within the TSN according to the policy “Data flow control policies”.
  - Presentation: the system shall restrict presentation of data to the user according to the policy “Presentation”.

### C. Engineering Process Applied to the Scenario

After the analysis of the security requirements of the scenario and its modelling (without security) the system engineer uses the SPT to search for the DSM that best suits the domain of the use case. An excerpt of the system model without security can be seen in Figure 7.

---

Figure 8. Excerpt of the Security-enhanced System Model with Secure Communication
As with the security analysis, due to the high number of elements, actors, functions, communications, etc. of the system we present only an excerpt of it. As the use case is about metering systems in a communication network she searches the DSM repository using those keywords and finds the “Metering Devices DSM”. She imports the DSM to the framework and starts checking for the security requirements.

Following, we present how the security engineer applies one security property to the system. The security requirements are “Confidentiality of transmitted data” and “Integrity of transmitted data”. After analysing the Metering Devices DSM the security engineer checks the Secure Communication property. This property fulfills both the confidentiality and the integrity of transmitted data. After analysing the property and checking it covers all the required functionalities and requirements of the system (the confidentiality and integrity of the transmitted data between two elements) she applies it to the “authenticate()” operation of the TSN Administrator Server class. The property is applied to the operation because it is the one that will communicate with other elements of the system. We think this structure helps understand better where the properties and solutions must be applied after the modelling phase. Figure 8 shows an excerpt of the system model with the applied security property. All the elements of the diagram are imported automatically to the system by the SPT. As we can see, it contains information about the threats and attacks that can target the system that uses the property, thus the system engineer can perform tests to check its resilience. The Secure Communication property is composed, as is shown in Figure 8, of the two security properties the system engineer needed (Integrity and Confidentiality). Being the Figure an excerpt it doesn’t show the assumptions, certifications, etc. but it can give an idea of the potential of both the security engineering process and the SPT providing the important security knowledge gathered and represented in the DSM.

Finally, after the system engineer applies the security properties to the system in order to fulfill all the security requirements she starts to deploy the SFPs attached to each security property to the system. Figure 9 shows the SBB that implements the security solution (the SFP) of the security property Secure Communication. Due to the size restriction we cannot show the complete version of the SBB so we present an excerpt of it.

The SPT automatically integrates the solution in the system model using the SBB functional description included in the SFP. The SBB is composed of several types of components: the main package with the required elements to provide the implementation, the external components and several connectors with the system architecture. The type of these special elements has been described in the SBB modelling process using a markup language which has been previously analyzed by the tool. This merging process guided by the SPT asks system engineers for all the required elements to use or create elements in the system model.

In this case, the integration process requires a context element “OpenSSLinterface”, which is created to indicate the mandatory access of this component. The SPT also requires two exchange elements for using certificates in the implementation. The use of these certificates is explained in the internal description of the security property but they have to be part of the architecture. Finally the SBB is connected to the system through the security requirement it fulfills.

In our previous step this security element was attached to the operation “authenticate()” in the “TSN Administrator Server” class, therefore the SPT uses this element as the

Figure 9. Excerpt of the SBB for Secure Communication
main component to be exchanged with the SBB, creating its corresponding links.

Once the deployment has been completed the system engineer proceeds with the development phase. The SPT does not support this stage of the process but facilitates it because it provides the architecture with the implementation, all the documentation about the security properties and the solutions (as SBBs).

VI. CONCLUSIONS

This paper presents the creation of a security-enhanced system model using the SecFutur Engineering Process and the SPT. The experience suggests (from real experiences in the SecFutur project) that the approach is combinable with the already existing engineering formalisms and iterative and agile development methods. The SecFutur approach can help in solving the demands of secure applications in the nowadays complex world, where security is heavily underestimated and not a focus during the development.

The main goal of the security engineering process is a clear separation between the expertise domains. Security engineering is not only the task of the highly experienced security expert. Using this approach, also system engineers are in a position to make sound security engineering decisions. In general terms, the integration of security engineering into the regular UML-based system engineering has at least two benefits. First and foremost, as security engineering is naturally integrated from the beginning into the modelling process, it helps to avoid design decisions that are contrary to the security requirements (a frequent problem when security is only considered in the final stages of development). Second, the SecFutur artefacts provide a common language for the roles involved in the engineering of the system, facilitating their communication.

The process is supported by a MagicDraw plugin which is being developed along with the NoMagic company and the company partners of the SecFutur project. It provides all the necessary tools for working with the process presented here in a easy and simple way.

We presented the process using a real world scenario of metering systems with legal requirements. Due to the page limit restriction we could not present all the requirements, models, security properties and solutions (in the form of SBBs) we have created in the project but the excerpt can give an idea of the potential of the process and the tool.

ACKNOWLEDGMENT

This work is partially supported by the E.U. through FP7 project SECFUTUR (IST-25668) and FP7 project CUMULUS (IST-318580).

REFERENCES