Guiding the selection of security patterns based on security requirements and pattern classification

ANAS MOTII, CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, F 91191, Gif-sur-Yvette
BRAHIM HAMID, IRIT (Institut de Recherche en Informatique de Toulouse), University of Toulouse
AGNÈS LANUSSE, CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, F 91191, Gif-sur-Yvette
JEAN-MICHEL BRUEL, IRIT (Institut de Recherche en Informatique de Toulouse), University of Toulouse

Security pattern-based system and software engineering (PBSE) approaches aim at building secure software and systems by capturing and reusing artifacts that encapsulate security expert's knowledge called security patterns. In this context, security patterns are selected by developers based on security requirements. On the other hand, security risk management is an iterative approach that consists of: (1) a risk assessment activity for identifying, analyzing and evaluating security risks and (2) a risk treatment activity to mitigate these risks which result in issuing security requirements. Hence, risk management and security PBSE can be used together. In this context, this paper aims at guiding the selection of security patterns in security PBSE based on security risk management results and pattern classification. For illustration purposes, we consider an example of a SCADA (Supervisory Control And Data Acquisition) system.

Categories and Subject Descriptors: **D2.11 [Software Engineering]**: Software Architecture—Patterns

General Terms: Pattern

Additional Key Words and Phrases: UML, SysML, Pattern-based system engineering, Model-based system engineering, concrete and abstract security patterns

ACM Reference Format:

Motii, A. and Hamid, B. and Lanusse, A and Bruel, J-M. 2015. Guiding the selection of security patterns based on security requirements and pattern classification. jn 0, 0, Article 0 (July 2015), 17 pages.

1. INTRODUCTION

Securing software and systems is a challenging issue for industry. Academia all agree that security should be treated in early stages of the software and systems development lifecycle [Weiss and Mouratidis 2008]. Otherwise, security vulnerabilities are more likely to be introduced in various stages [McGraw 2006] and the cost of protecting them becomes increasingly more important. In this context, the use and application of security mechanisms through the lifecycle process would be easier if designers and developers had security guidelines during development. This can be done by capturing and providing security expertise with the help of security patterns. Such approaches are called Pattern Based System and Software Engineering (PBSE) approaches. Indeed, PBSE [Ackerman and Gonzalez 2010] is a system and software engineering discipline that considers patterns as first class citizens. It addresses two kinds of processes: pattern development and system development using patterns. The first concerns designing patterns from expertise for reuse. The second concerns selecting appropriate patterns in each development lifecycle stage. In this work, we are focusing on the second process of PBSE for security that uses security patterns to build secure software and

EuroPLoP '15, July 08 - 12, 2015, Kaufbeuren, Germany
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3847-9/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2855321.2855332

systems. In this context, risk management identifies estimates and evaluates risks and delivers security requirements. However, selecting appropriate security patterns from security requirements is not an easy task [Weiss and Mouratidis 2008].

The solution that we envisage here is to use risk management defined in the ISO/IEC 27005 standard [ISO/IEC 27005 2011] with a PBSE approach called (System and software Engineering with Multi-Concerns) SEMCO [Hamid et al. 2013][Hamid and Percebois 2014] during the design process. SEMCO provides a model-based repository where S&D (security and dependability) patterns can be stored in a first step and reused in the development process. In this paper, we only focus on security patterns.

Our goal is to exploit risk management results (i.e., security requirements) and security pattern classification in order to guide the selection of security patterns from the SEMCO pattern repository. The use of Model-based System Engineering (MBSE) enhances communication rather than using traditional documents to exchange information. We use SysML [Friedenthal et al. 2008] as a modeling language for illustration. SysML is a general purpose visual modeling language for systems engineering applications.

Software and systems follow classical development lifecycle process models in Figure 1. In this paper, we focus on one lifecycle stage: high level design. At a first step, the system functions are described independently of any underlying platform and software details producing a system architecture. In a second step the functions are realized by hardware/software resources producing a hardware/software architecture.

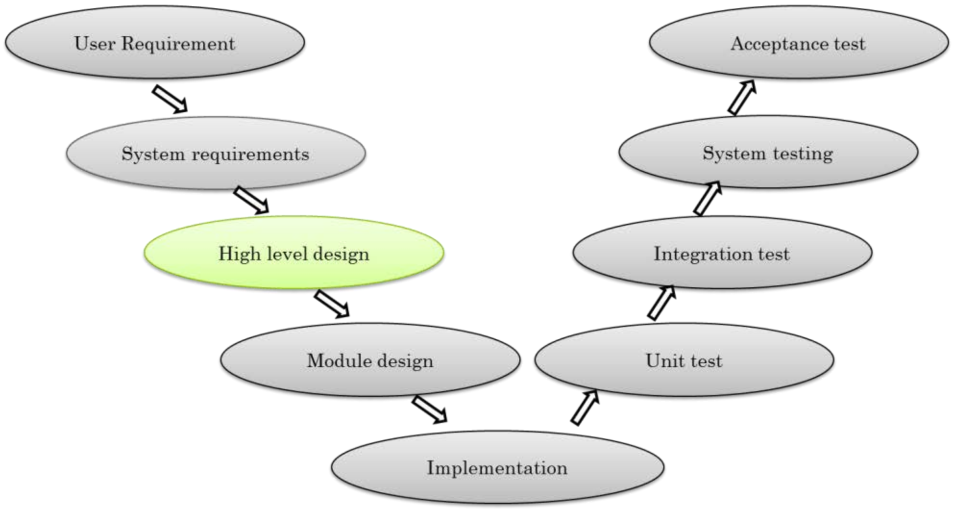


Figure 1. Lifecycle process model [Friedrich et al. 2009]

The selection deals with two types of patterns: abstract patterns at system architecture and concrete patterns at the hardware/software architecture. In this context, we exploit the concepts of abstract and concrete patterns were introduced in [Fernandez et al. 2008]. Abstract patterns describe the features they offer to the system without mentioning details about implementation. In contrast, concrete patterns implement features provided by an abstract pattern. In addition one abstract pattern can be implemented by many concrete patterns. This work has been inspired from [Fernandez 2013] where the author presents a pattern-based method to build secure systems and uses abstract patterns at early stages and concrete ones at lower stages.

The remaining sections are organized as follows. Section 2 gives an overview of the approach describing the context and security risk management issues, and then it introduces the SEMCO approach. Notably, section 2.2 presents the conceptual models of security risk management and related patterns in SEMCO and identifies some interesting concepts that allow the selection of security patterns. Section 3 presents the security pattern selection method. Section 4 illustrates the proposed method on a SCADA (Supervisory Control And Data Acquisition) case study; it shows how risk analysis can issue security requirements that are used to select security patterns from SEMCO repository. Section 5 discusses related works and finally section 6 concludes the paper and presents directions for future work.

2. BACKGROUND

2.1 General overview of the approaches

In this section, we first present the context of the paper by giving an overview of the general approach for building secure software and systems. Then, we present security risk management based on the ISO/IEC 27005 standard and the SEMCO approach which is parts of the general approach.

2.1.1 Overview of the general approach

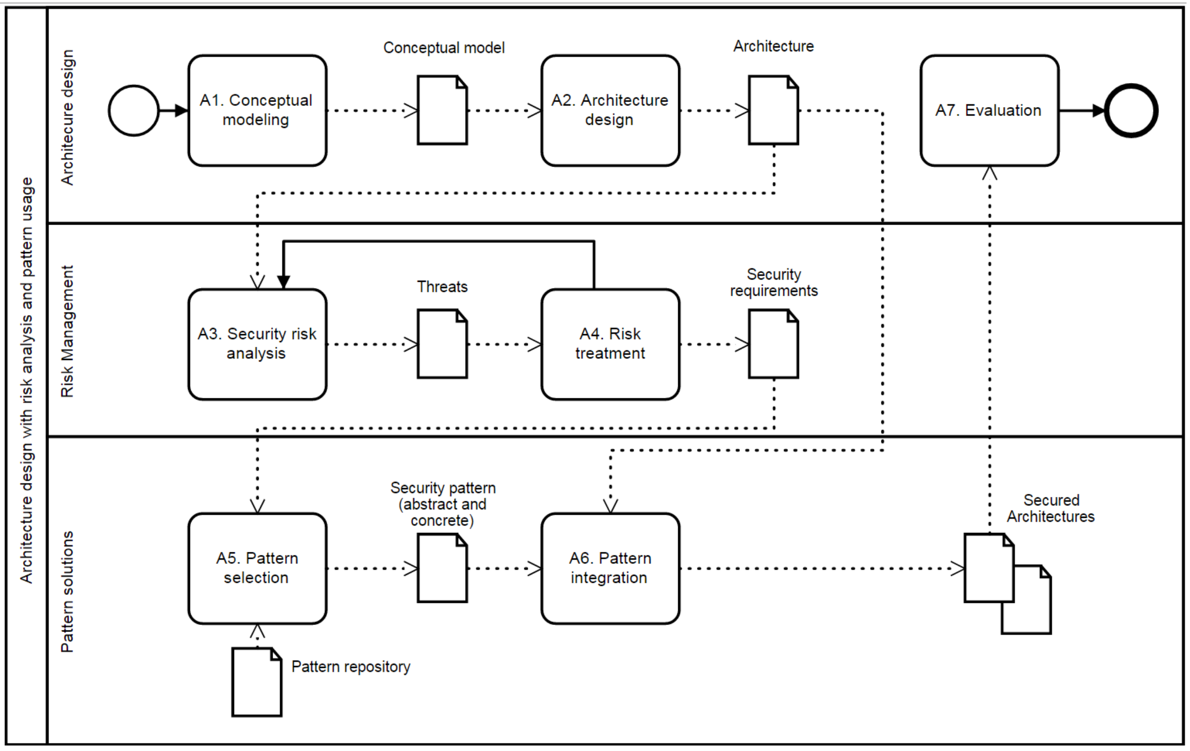


Figure 2. Architecture design with risk analysis and pattern use

The general approach consists of building secure software and systems at high level design stage. Figure 2 shows an excerpt of the approach which consists of three collaborative processes: architecture design, risk management and solution using patterns. In the architecture design process, first a conceptual model of the system is designed (A1) then system architecture is designed describing the functions of the system (A2). These functions are realized by hardware and software resources producing hardware / software architecture.
In the risk management process, the system architecture is submitted to risk analysis in order to enumerate threats (A3) and to derive security requirements (A4). This process is done by a security risk analyst. Security requirements describe what should be provided by security mechanisms in order to stop some threats. They do not deal with how they are implemented.

In the pattern solution process, a selection of patterns providing the right properties to satisfy security requirements is performed (A5), then the selected patterns are applied to the architecture (A6) and submitted to evaluation by designers (A7) in order to check if they do not introduce side effects on other criteria than security issues (timing constraints, load, costs).
In this paper we focus on activities (A3), (A4) and (A5) to guide the selection of security patterns (abstract and concrete). For that matter, we exploit security requirements on one hand and pattern classification on the other hand.

2.1.2 Overview of security risk management

The ISO 27005 [ISO/IEC 27005 2011] standard belongs to the family of ISO 27000 standards [ISO/IEC 27000 2014]. It provides guidelines for security risk management and supports the concepts defined in ISO 27001 [ISO/IEC 27001 2013]. The standard does not provide a particular methodology. However methodologies that claim to perform information security risk management may conform to security standards such as the ISO 27005. In this paper, we do not use a specific methodology for risk management but a general approach that conforms to the standard. The process model of this risk management approach is displayed in Figure 3.

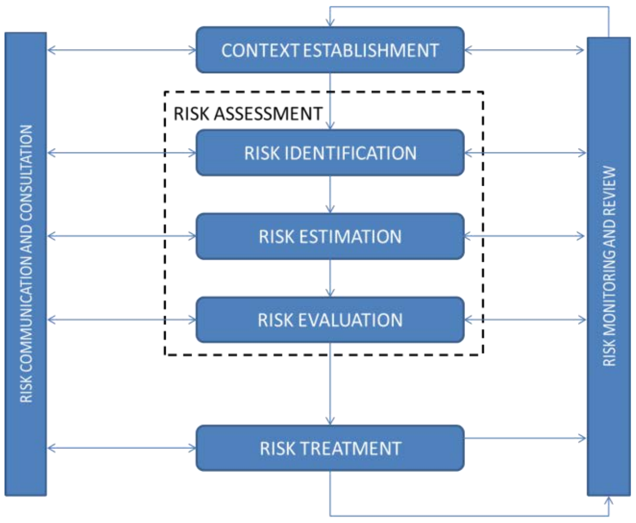


Figure 3. Risk management process model [ISO/IEC 27005 2011]

The information security risk management process consists of three main phases:

- **Context establishment:** the aim of this phase is to describe the system under study and to put boundaries to risk management.
- **Risk assessment:** consists of:
  - Risk identification: aims at defining assets and identifying threats and vulnerabilities of the system. There are several ways to enumerate threats such as: attack trees [Schneier 2000] , misuse cases [Sindre and Opdahl 2000], misuse activities [Braz et al. 2008] and others.
  - Risk estimation: aims at assigning a likelihood level (occurrence) and impact level (consequence) to each risk by security risk analysts. The assignment of likelihood considers factors such as threat source motivation and capability, nature of the vulnerability and existence and effectiveness of current controls [Stoneburner et al. 2002]. In the same way, impact is determined by performing an impact analysis [Stoneburner et al. 2002] that requires information related to the system mission (e.g., the process performed by an IT system), system

and data criticality (e.g., the system's value or importance to an organization), and system and data sensitivity. However this point is out of the scope of the paper and is not detailed.

- Risk evaluation: evaluates the risk level that is computed starting from the likelihood and impact levels according to a risk matrix. An example of a risk matrix is shown in Table 1. There are two levels in this case: *Low (L)* and *High (H)*.

Table 1. Risk matrix for evaluating risk form likelihood and impact levels [Braber et al. 2007]

Likelihood level / Impact level	Rarely	Sometimes	Regularly	Often
Harmless	L	L	L	H
Moderate	L	L	L	H
Serious	L	H	H	H
Catastrophic	H	H	H	H

- **Risk treatment:** aims at reducing, avoiding, accepting or transferring risks. The output of this activity is a set of security requirements.

Risk communication and monitoring are processes that communicate risk information to all decision makers and ensure monitoring of risk levels respectively.

2.1.3 Overview of SEMCO

SEMCO is a model- and pattern-based approach for software and systems development. It provides a model-based repository of reusable security patterns. In fact, security patterns are developed from the security expert's knowledge, validated and stored in the model-based repository. As it was said before, in this paper we do not discuss this aspect.
The approach uses a pattern language inspired from GoF [Gamma et al. 1995] to describe security patterns that is discussed in section 2.2.2. The solution of patterns is language independent. However, in this paper we use UML language [Rumbaugh et al. 2004].
The approach provides facilities to filter pattern search. Patterns are classified in the repository according to expected **security properties** related to categories (confidentiality, integrity, availability, authorization, non-repudiation, authenticity etc.) also called security attributes in [Avizienis et al. 2004]. In addition, in the case of domain specific patterns (e.g., patterns for networks); a second classification can be made according to the **application domains** they can be applied to.
In general, patterns are used to stop a specific threat and are selected according to this kind of threat. This requires some security expertise. Since most developers tend to view security requirements in terms of what should happen. The approach views patterns in terms of what they provide as security properties rather than the kind of threats they act against [Bunke et al. 2012].

2.2 Risk management and SEMCO conceptual models

In this section we present the conceptual models of patterns in SEMCO and risk management.

2.2.1 Security risk management conceptual model

There are several conceptual models for risk management such as EBIOS conceptual model [ANSSI 2010], CORAS conceptual model [Braber et al. 2007], Information System Security Risk Management (ISSRM) [Dubois et al. 2010] and others. For presenting the concepts, we choose the ISSRM which is in compliance with the ISO/IEC 27005 standard and contains three groups of concepts: asset-related concepts, risk-related concepts and risk treatment-related concepts. Figure 4 shows a simplified conceptual model of ISSRM. The definitions of the concepts are extracted from [Dubois et al. 2010].

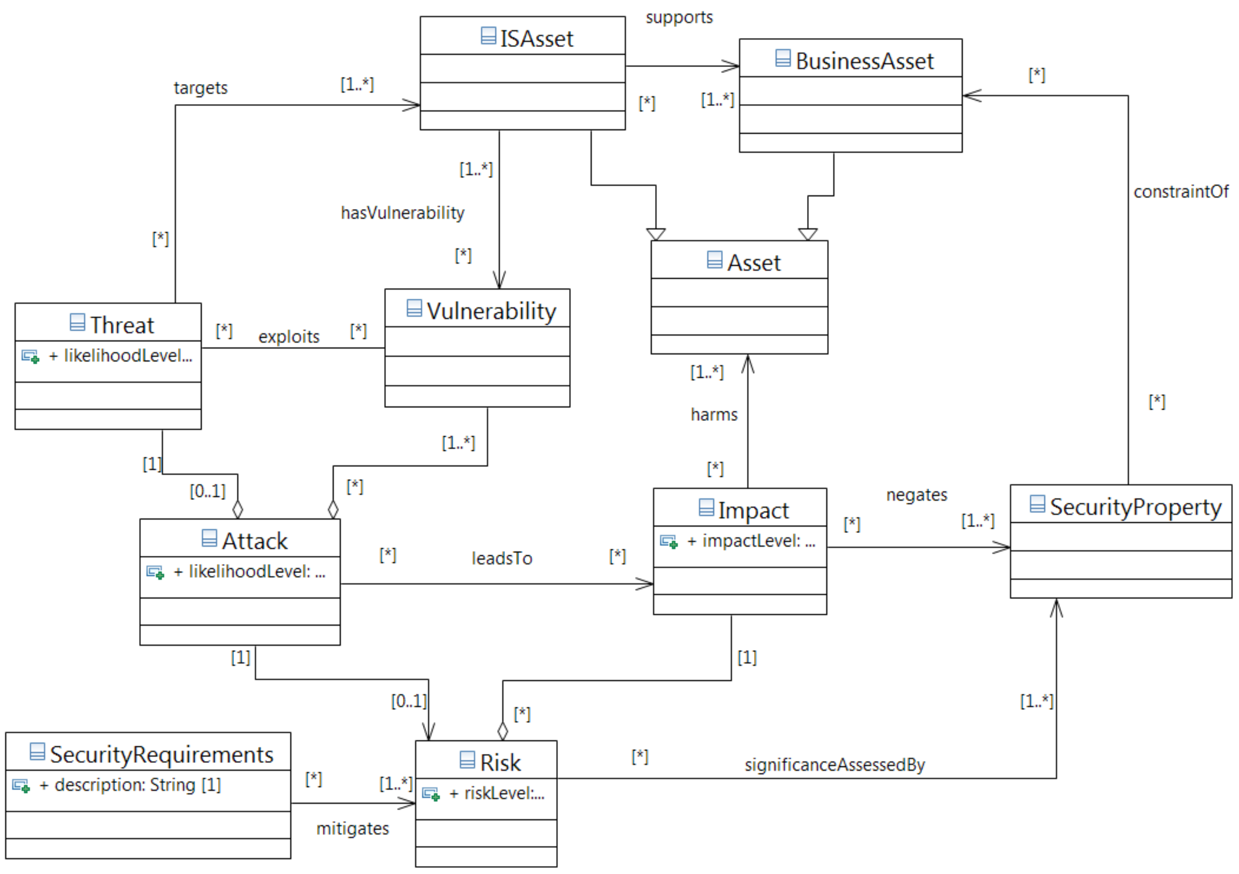


Figure 4. Simplified ISSRM Risk management conceptual model [Dubois et al. 2010]

Asset related concepts:

- **Asset:** something that has value in an organization and is essential to reach its objectives. This concept generalizes the business asset and the IS (Information System) asset.
- **Business Asset:** information, process or skill inherent to the business of the organization. It is essential for achieving its objectives (e.g., technical plan, structure calculation process, architectural competence). Business assets are immaterial.
- **IS Asset:** a component or part of an IS that has value to the organization and necessary for reaching its goals. IS assets are classified according to these **categories**: *hardware*, *software*, *networks*, *people* or *facilities* involved in the IS security. IS assets are material with the exception of software.
- **Security property:** are properties or constraints at organization level over business assets that reflect their security need (e.g., confidentiality, integrity, availability, non-repudiation, accountability etc.). Security properties are used to express security objectives of an IS (e.g., confidentiality of the business of the organization).

Risk-related concepts:

- **Vulnerability:** flaw in an IS asset that constitute a weakness.
- **Threat:** potential attack by a threat agent using an attack method. It has a likelihood level.
- **Attack:** the combination of a threat and one or more vulnerabilities. It has a likelihood level, which is the maximum level among all threats. For example: a hacker uses social engineering on an unexperienced employee exploiting weak awareness.
- **Impact:** negative consequence of a risk that harms an asset, when a threat or an attack is accomplished. It has an impact level.
- **Risk:** the combination of threat with one or more vulnerabilities leading to an impact that harms one or more assets. The risk level is computed as the product of the consequence of impact times the likelihood of the attack.

Risk Treatment-related concept:

- **Security requirement:** are properties that the system must possess over the environment by installing the IS, in order to mitigate risks. There are other risks treatment-related concepts but we do not present them for a matter of simplicity.

2.2.2 SEMCO: pattern conceptual model

Patterns in the SEMCO approach are represented using the conceptual model in Figure 5 named (System and software Engineering Pattern Metamodel) SEPM. SEPM is a refinement of the GoF [Gamma et al. 1995] pattern conceptual model adapted to security. The main concepts in SEPM are:

- **SepmPattern:** it is the main concept; it encapsulates a solution of a recurrent problem. A pattern can be **SepmDIPattern** (domain independent) or **SepmDSPattern** (domain specific, e.g., networks). In addition a pattern can be *abstract* or *concrete*.
- **SepmInterface:** patterns interact with their environment with interfaces. Each interface consists of a set of operations. There are two types of interfaces:
  - **SepmExternalInterface**: allows implementing integration of a pattern into an application.
  - **SepmTechnicalInterface**: allows implementing interaction of a pattern with the platform. At a low abstraction level, it is for example possible to define links with a software or hardware module for the cryptographic key management. These interfaces are realized by the SepmDSPattern. Note that a SepmDIPattern does not have technical interfaces.
- **InternalStructure**: it is the implementation of the pattern solution. An internal structure can be: static or dynamic.
- **SepmProperty:** is a characteristic of an abstract or concrete pattern describing the security aspect it deals with. It is related to a category. It can be a *security property* or a *resource property*. It is important to note that there is a difference between a **security property** and **security property category**.

  For example: “confidentiality” is a general security property category while, “confidentiality of transmitted data at transport level” is a specific security property. These properties are validated during the pattern development process [Hamid and Percebois 2014].

In the case of domain specific applications, the pattern delivers a *security property* assuming that the domain application satisfies a set of *constraints*.

- **SepmConstraint:** related to concrete patterns, assumptions that the application must satisfy in order for the pattern to work properly. Constraints may concern resources, security properties or both (e.g., a pattern provides authenticity of data between two units but the application must have a safe communication channel).

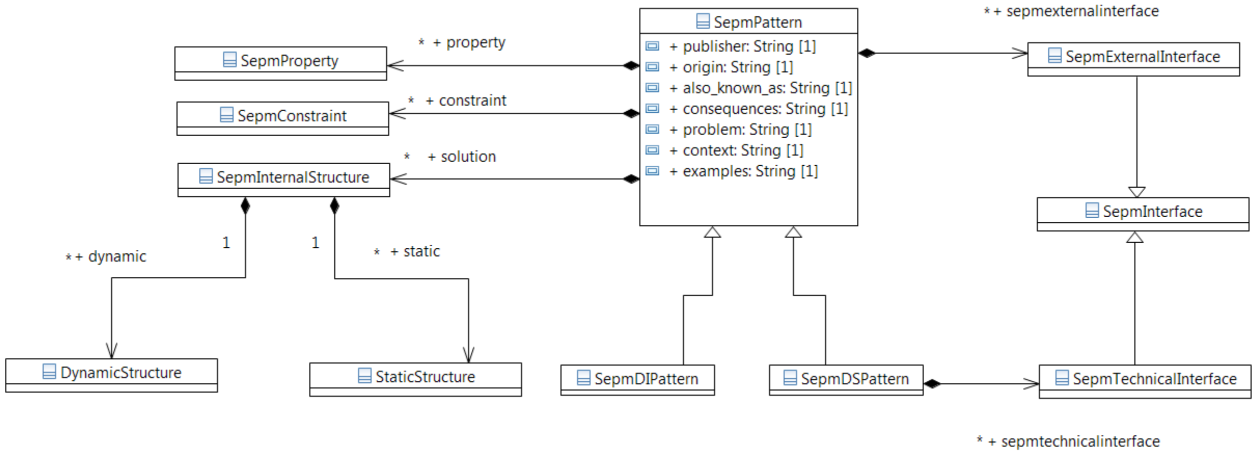


Figure 5. SEPM conceptual model (Excerpt)

2.2.3 Discussion

The conceptual models of patterns and security risk management have allowed us to identify interesting concepts: **security property**, **application domain/ asset category** and **constraints**. We believe that these concepts can guide the selection of security patterns.
On one hand, risk management derives security requirements that can be expressed as a table of: **assets**, **asset category**, **risks** and **expected security properties.**
On the other hand, security patterns are classified in SEMCO according to **security properties** and **application domain**. Indeed, according to the **application domain** and the desired **security property**, an abstract domain specific security pattern is selected describing only the basic functionalities (interfaces) and the provided *security properties* without implementation details. In addition, abstract security patterns may be realized by many concrete patterns. The selection of one implementation against another is driven by its *constraints* (e.g., resource constraints) with regards to the application model.

3. SECURITY PATTERN SELECTION APPROACH

The discussion made in section 2.2.3 has led to a clear identification of the concepts: *security property, application domain* and *constraint* in order to guide the selection of security patterns.

From the security analysis, security requirements are issued. They are classified according to the nature of the concerned assets and to general security property categories.
Asset category can be [Bunke et al. 2012]:

- Network
- Software/Hardware
- Cryptographic

- Users
- Enterprise

Security property categories can be [Fernandez 2013]:

- Confidentiality: means that data must remain secret for unauthorized users
- Integrity: means that only authorized data modification in a system is allowed
- Availability: means that legitimate users must have access to their system.
- Authenticity: expresses the necessity to ensure communications are genuine. Evidence should be given for proving that both parties are who they claim to be.
- Non-repudiation: means that users are responsible for their action and should not have the ability to deny their actions.

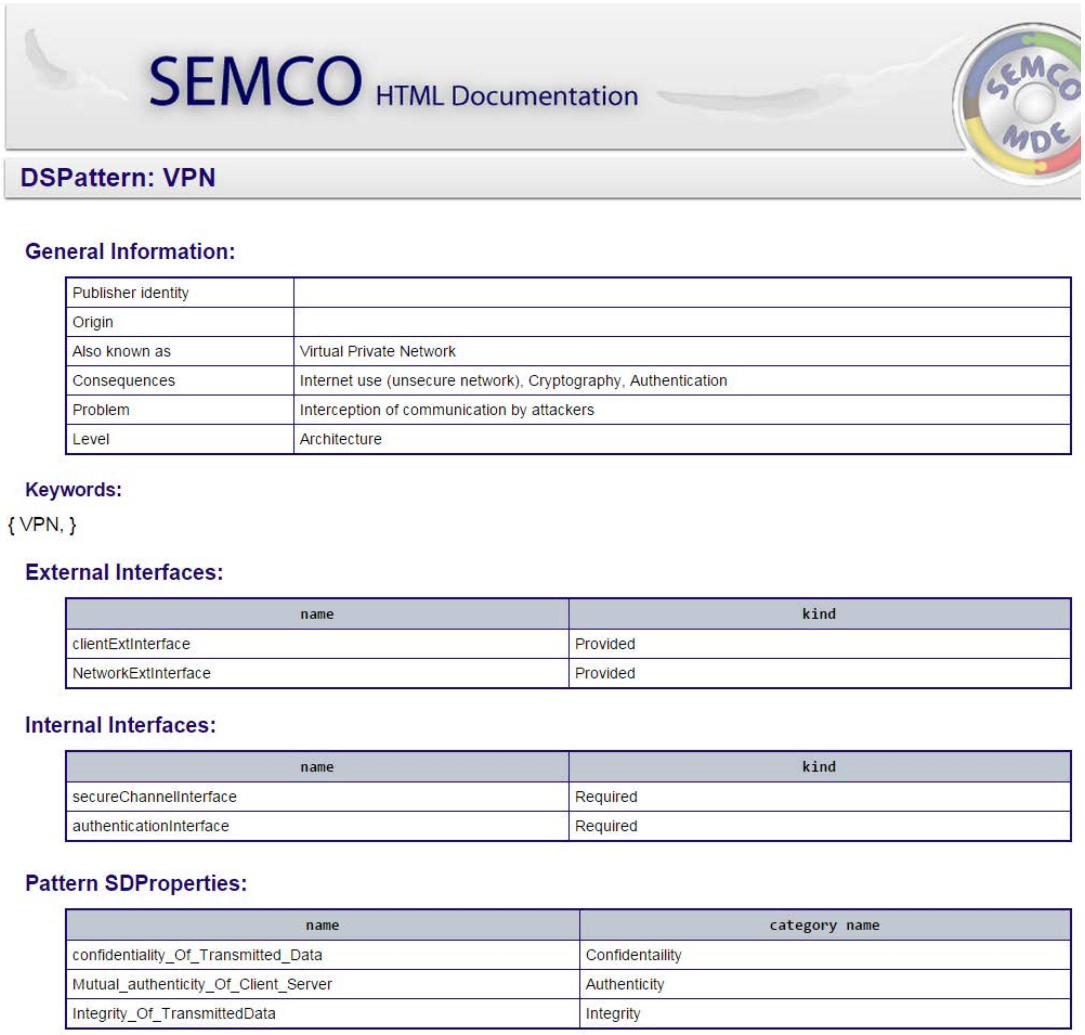
These criteria are then exploited to select pattern candidates. The selection is achieved using the classification of security patterns according to *security properties* and the *application domain* during activity (A5) in Figure 2. The SEMCO pattern repository suggests a set of domain specific abstract patterns. Each implementation (i.e., concrete pattern) of these patterns has a set of *constraints* over the application domain model in order for the pattern to provide the desired security properties. As a result, the choice of the patterns is simplified.

Example: the selection of a Virtual Private Network (VPN) pattern model [Fernandez 2013]

A software architect searches for a security pattern for networks. It should offer the following security properties:

- confidentiality of transmitted data
- integrity of transmitted data
- mutual authentication of client and server

Using the above information about **application domain** and needed **security properties**, the search in the repository leads to the identification of the security pattern VPN. VPN describes how to set up a secure channel between two endpoints using cryptographic tunneling, with authentication at each endpoint. Figure 6 shows the generated HTML documentation of the VPN pattern in SEMCO. It describes general information (problem, consequences and lifecycle stage), the external and internal interfaces and the security properties of this pattern.
The internal structure (static) of the pattern solution is represented in Figure 7 . It has two external interfaces: *ClientExtInterface* and *NetworkExtInterface*. It uses the *secure channel* [Braga et al. 1999] and *authentication* patterns [Fernandez 2013]. In fact, when a client tries to access an endpoint in a network, the *authentication* pattern creates a proof of identity used to establish a *secure channel*. The client can then communicate securely with the network endpoint.
At this level the VPN pattern does not have constraints because it is abstract. VPN can use IPsec protocol (IPsec VPN) at the IP layer or TLS protocol (TLS VPN) at the TCP layer. Each implementation (concrete pattern) of the VPN pattern has its own constraints on the application model. The decision between one of these VPN implementations depends on the satisfaction of their **constraints**. For Example, the IPsec/TLS VPN requires an installation of large software packages (6 to 8 MB) and some processing resources [Fernandez 2013] .



DSPattern: VPN

General Information:

Publisher identity	
Origin	
Also known as	Virtual Private Network
Consequences	Internet use (unsecure network), Cryptography, Authentication
Problem	Interception of communication by attackers
Level	Architecture

Keywords:

{ VPN, }

External Interfaces:

name	kind
clientExtInterface	Provided
NetworkExtInterface	Provided

Internal Interfaces:

name	kind
secureChannelInterface	Required
authenticationInterface	Required

Pattern SDProperties:

name	category name
confidentiality_Of_Transmitted_Data	Confidentaility
Mutual_authenticity_Of_Client_Server	Authenticity
Integrity_Of_TransmittedData	Integrity

Figure 6. VPN pattern html documentation in SEMCO

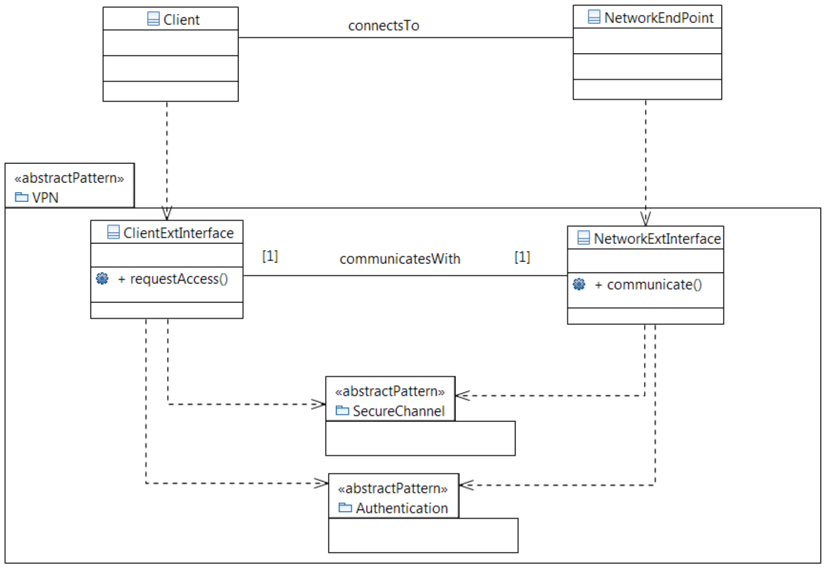


Figure 7. VPN pattern solution model with UML

4. ILLUSTRATION ON A SCADA SYSTEM

In this section, we introduce the SCADA system case study and illustrate the architecture design (A2), risk management (A3, A4) and pattern selection (A5) activities.
SCADA systems are designed to continuously control and monitor plants such as water treatment and power generation. Nowadays, industrial demands have made SCADA systems connected to corporate networks and Internet. Consequently, security can no longer be ignored. Basically, SCADA systems consist of a central controller, a communication network and field units (local controllers, sensors and actuators) in Figure 8. The central controller provide the user with a HMI to send set points such as (temperature, pressure, flow) and commands to local controllers called Programmable logic Controllers (PLC). In addition, the central controller polls data from PLCs and visualizes in real time the current process and status information. This data can also be archived in a data server and reused for further evaluation. The feedback loop is programmed inside PLCs which are connected to sensors and actuators. The communication network connecting the central controller and PLCs can be MODBUS, Fieldbus, CAN, Profibus and Ethernet.

4.1 Architecture design

A simplified generic SCADA system is modeled with SysML diagrams using the Papyrus tool [Gérard 2015]. The components are represented as SysML blocks in the left part of Figure 8. The right part shows the system architecture connecting the control center, the communication network and the field unit with each other through input/output ports.

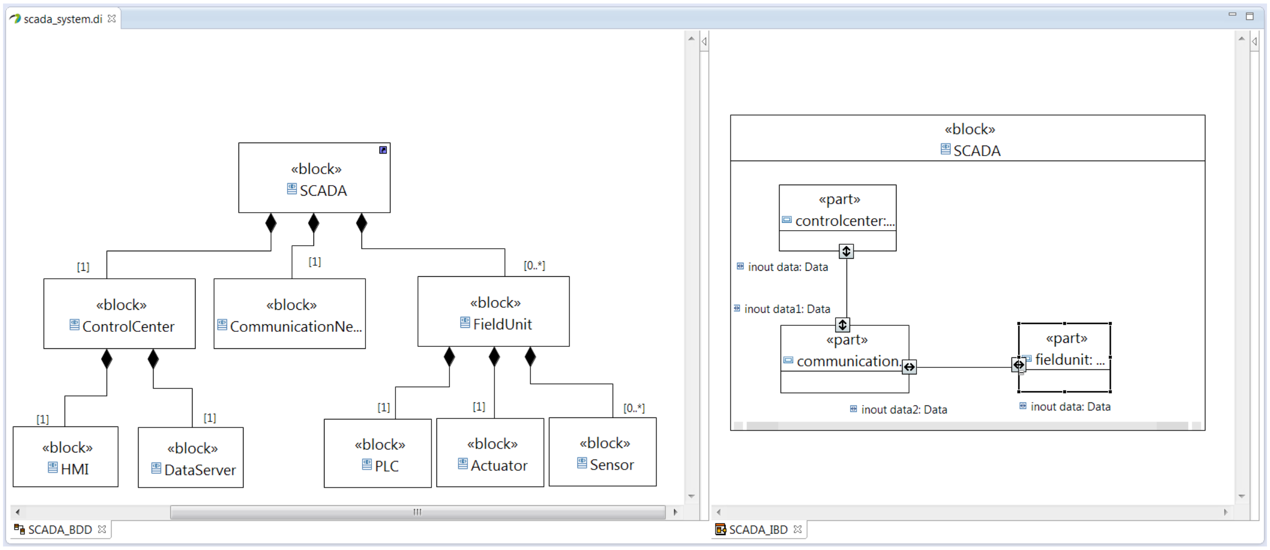


Figure 8. SCADA system architecture SysML model

4.2 Risk management

We follow the risk management approach described in section 2.1.2 illustrated on the SCADA system. The illustration follows three steps: context establishment, risk assessment and risk treatment. Risk management decisions such as risk evaluation and risk treatment are done by security risk analysts. Here, we only illustrate the steps of decision making. The tooling support is based on previous work [Abdallah et al. 2015].

- **Context establishment:**

The context of the study is the SCADA system in Figure 8. The study is limited to the control center, the communication network and the field units described in the model. Hence, these components are considered to be assets. SCADA systems contain other components such as local networks. For simplification, these components are not detailed here.

- **Risk assessment:**

Risk identification

The considered assets are the control center, the communication network and the field units. As it was said, there are several ways to enumerate threats such as: attack trees [Schneier 2000], misuse cases [Sindre and Opdahl 2000], misuse activities [Braz et al. 2008] and others. Table 2 shows a list of some of the risks targeting the assets for such SCADA systems [Fernandez 2013].

Table 2. List of security risks in SCADA

Assets	Risks
Control center	Risk.1 Physical attacks
	Risk.2 Malicious settings of the field units
	Risk.3 Wrong commands sent to the field units
	Risk.4 Malicious alteration of the parameters of the control center
	Risk.5 Denial of service
Network communication	Risk.6 Sniffing commands
	Risk.7 Spoofing
	Risk.8 Denial of service
Field units	Risk.9 Physical attacks
	Risk.10 Malicious alteration of the runtime parameters
	Risk.11 Incorrect commands sent to the central controller
	Risk.12 Malicious alarms sent to the central controller
	Risk.13 Denial of service

Risk estimation and evaluation

This phase concerns assigning likelihood levels and impact levels. An example of risk assignment is given in Table 3 using risk matrix in Table 1 for likelihood and impact levels. For the two risks, the risk levels are high which means that they have to be treated.

Table 3. Risk table with risks and risk levels

Risks	Likelihood level	impact level	Risk level
Risk 2	Sometimes	Serious	**High**
Risk 3	Regularly	Serious	**High**

- **Risk treatments:**

This step concerns the specification of a set of security requirements that mitigate the risks. For example, some of the security requirements for risks 2, 3, 4, 6, 7 and 8 are:

- There should be an access control mechanism for the control center.
- There should be a mechanism that enforces non-repudiation.
- There should be a mechanism for authentication.
- There should be mechanism for secure communication that guarantees data integrity, confidentiality and authenticity.

The above security requirements are reformulated in form of a table in order to be used by the pattern selection activity. The table in Figure 9 specifies the assets, their categories, their risks and the needed security properties. Since the control center and field units have similar risks we only display the ones of the control center. The security properties are the key elements to guide pattern selection. In our case, a set of security properties have been specified for each asset.
For the control center, the security properties are:

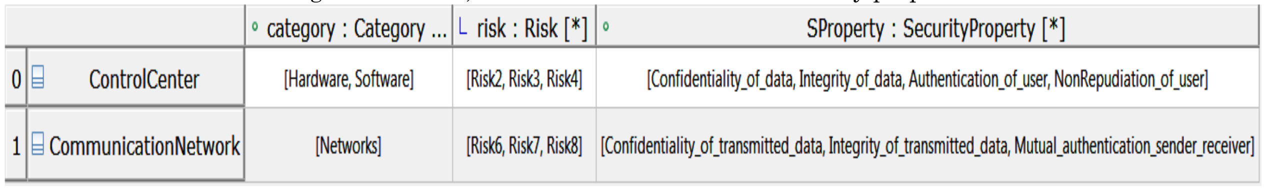
- Confidentiality of data
- Integrity of data
- Authentication of users
- Non repudiation users

For the communication network the security properties are:

- Confidentiality of transmitted data.
- Integrity of transmitted data

- Mutual authentication of sender and receiver.

Figure 9. Assets, their risks and the needed security properties



		category : Category ...	risk : Risk [*]	SProperty : SecurityProperty [*]
0	ControlCenter	[Hardware, Software]	[Risk2, Risk3, Risk4]	[Confidentiality_of_data, Integrity_of_data, Authentication_of_user, NonRepudiation_of_user]
1	CommunicationNetwork	[Networks]	[Risk6, Risk7, Risk8]	[Confidentiality_of_transmitted_data, Integrity_of_transmitted_data, Mutual_authentication_sender_receiver]

4.3 Pattern selection

Once the assets, their categories and the security properties have been established in Figure 9, they serve as the key information to identify abstract patterns and concrete patterns in the model-based repository.

- Abstract patterns:

For the control center asset (hardware/software category), the security properties "confidentiality of data", and "integrity of data", "authentication of users" and "non-repudiation of users", " lead to the identification of the abstract patterns: **Authentication, Security logger and auditor, and Authorization** [Fernandez 2013]**.**
In the same way, for the communication network, the asset category and the security properties have led to the identification of the abstract pattern **VPN** [Fernandez 2013]. The secured system architecture with abstract patterns usage is shown in Figure 10. Notice that since the control center and the field units have the same risks they use the same abstract patterns.

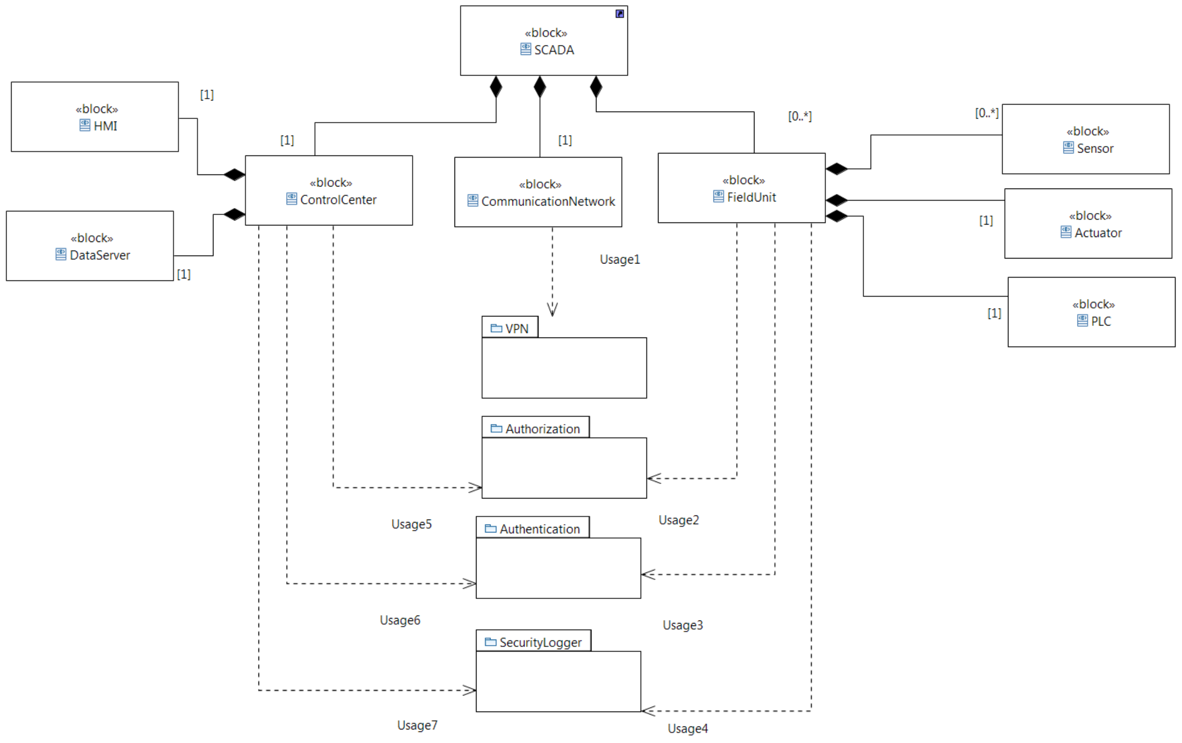


Figure 10. Secured system architecture with abstract patterns usage

- Concrete patterns:

Finally, it is necessary to select a concrete pattern to apply among possibly several candidates. We illustrate this step with the case of securing the communication network asset. It falls in the **network category** and presents risk 6, 7 and 8 in Table 2. As it was shown in the example in section 3 using the **security properties**, the model-based repository suggests the abstract **VPN** pattern that has two implementations (concrete patterns): **IPsec VPN** and **TLS VPN**. The two candidate solutions are applied on the architecture and then evaluated against other criteria (timing constraints, load, cost) using architecture exploration tools (Qompass [Mraidha et al. 2011]) not detailed here. The final decision of selecting one of the concrete patterns is driven by their constraints over hardware/software resources: HMI, Data servers, PLC, sensors, actuators etc.
The integration of the concrete patterns in the architecture model can follow a methodology that has been done in [Hamid et al. 2012] and [Radermacher et al. 2013]; a new version of such automatic pattern integration is currently under development but is not in the scope of this paper.
For instance, Figure 11 shows the application of the TLS pattern on the SCADA system. The elements in light grey represent the additional components added by the pattern (vPNClientSide and proxy). The central controller uses the main URL for the proxy server in the web browser, and connects to it through TLS-protected HTTP.

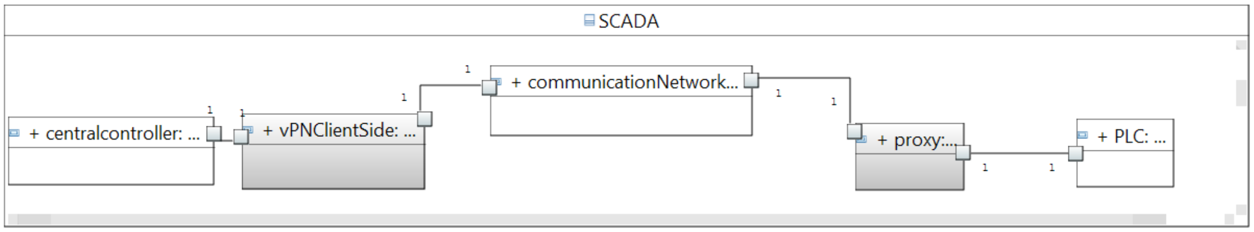


Figure 11. SCADA system with SSL VPN application

As a conclusion, experimentally speaking, the concepts of security property, application domain and constraints allow the selection of security patterns.

5. RELATED WORK

Selecting appropriate patterns is an essential phase during the development of secure software and systems. So, many works are tackling this subject. We present here a selection of them regarding several aspects of this concern. The discussion is thus organized according to: secure development methodologies, general pattern selection techniques, and more specific methods devoted to the domain of security issues.
In [Fernandez 2011], the authors presented a secure development methodology where the selection of security patterns is aided with a multidimensional classification of patterns according to: lifecycle stage, concern, domain, architectural level and response type. In [Uzunov et al. 2012], the authors have surveyed and compared relevant methodologies for distributed systems. The surveyed methodologies use different ways for the selection of security patterns. Some use ad-hoc fashion, i.e. directly from security requirements. Others base the selection on a structured catalog of patterns, following guidelines in the overall approach or according to pre-defined schemas and conceptual frameworks.
Other works have presented methods for selecting patterns. They used different techniques: text classification [Hasheminejad and Jalili 2012], goal oriented formalization [Weiss and Mouratidis

2008], classification based on properties and threats [Bunke et al. 2012][Alvi and Zulkernine 2012], and ontologies [El Khoury et al. 2008].
These works have been applied more specifically for security patterns, in [Weiss and Mouratidis 2008], the authors formalized security pattern in Goal-oriented Requirements Language (GRL) [Amyot et al. 2010] in order to assist a designer with the selection of security patterns. In [Hasheminejad and Jalili 2009], the authors proposed an automatic security pattern selection method based on text classification and learning techniques for the classification of security patterns. The process is automated for selecting security patterns from security requirements. The experiments have showed that Naive Bayes is the most appropriate learning technique for selecting security patterns. In [El Khoury et al. 2008], the authors proposed a security pattern selection method based on ontological mappings at two different levels: (1) at design level: between requirements for design-based developer profiles and (2) at implementation level: between threat models, bugs and errors for implementation-based developer profiles.
In [Alvi and Zulkernine 2012], the authors have provided a classification of security patterns for each development lifecycle stage. For instance, they proposed generic security properties for the design phase. In [Bunke et al. 2012], the authors introduced classification schemes based on application domains, security aspects (e.g., confidentiality, integrity, availability, accountability, authentication and access control) and pattern recognition needs.
The proposed approach is based on the works of pattern classification [Fernandez 2011][Bunke et al. 2012] according to properties and asset categories which is a subset of pattern dimensions. It adds the usage of risk analysis to derive security properties and constraints to refine the selection of concrete security patterns.

6. CONCLUSION

We propose a security pattern selection method based on classification according to properties and application domain. This method is guided by security risk analysis. The selection follows two steps: first the selection of abstract patterns responding to identified security requirements issued by a risk analysis; then we use constraints introduced by relevant concrete patterns to refine the selection.
This work takes part in a global development process, where the candidate patterns are integrated into the architecture in order to be evaluated and compared regarding other criteria in order to provide cost/benefit evaluations of solutions.

Acknowledgements. This work is conducted in the context of a Ph.D. thesis funded by CEA LIST and co-leaded by CEA (LISE) and IRIT (MACAO). In addition, we would like to thank our shepherd Dr. Eduardo B. Fernandez including the members of the writer's workshop who gave us valuable feedback on preliminary versions of this paper.

REFERENCES

Abdallah, R., Yakymets, N., and Lanusse, A. 2015. Towards a Model-driven based Security Framework. *MODELSWARD 2015 - Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, ESEO, Angers, Loire Valley, France, 9-11 February, 2015*, SciTePress, 639–645.

Ackerman, L. and Gonzalez, C. 2010. *Patterns-Based Engineering: Successfully Delivering Solutions via Patterns.* Addison-Wesley Professional, Upper Saddle River, NJ.

Alvi, A.K. and Zulkernine, M. 2012. A Comparative Study of Software Security Pattern Classifications. *2012 Seventh International Conference on Availability, Reliability and Security (ARES)*, 582–589.

Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E. 2010. Evaluating Goal Models Within the Goal-oriented Requirement Language. *Int. J. Intell. Syst. 25*, 8, 841–877.

ANSSI. 2010. *EBIOS 2010: Expression des besoins et Identification des Objectifs de Sécurité (2010).* ANSSI FRANCE.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. 2004. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput. 1*, 1, 11–33.

BRABER, F., HOGGANVIK, I., LUND, M.S., STØLEN, K., AND VRAALSEN, F. 2007. Model-based Security Analysis in Seven Steps — a Guided Tour to the CORAS Method. *BT Technology Journal 25*, 1, 101–117.

BRAGA, A., RUBIRA, C., AND DAHAB, R. 1999. *Tropyc: A Pattern Language for Cryptographic Software.* .

BRAZ, F.A., FERNANDEZ, E.B., AND VANHILST, M. 2008. Eliciting Security Requirements through Misuse Activities. *19th International Workshop on Database and Expert Systems Application, 2008. DEXA '08*, 328–333.

BUNKE, M., KOSCHKE, R., AND SOHR, K. 2012. Organizing security patterns related to security and pattern recognition requirements. *International Journal on Advances in Security 5*.

DUBOIS, É., HEYMANS, P., MAYER, N., AND MATULEVIČIUS, R. 2010. A Systematic Approach to Define the Domain of Information System Security Risk Management. In: S. Nurcan, C. Salinesi, C. Souveyet and J. Ralyté, eds., *Intentional Perspectives on Information Systems Engineering*. Springer Berlin Heidelberg, 289–306.

EL KHOURY, P., MOKHTARI, A., COQUERY, E., AND HACID, M.-S. 2008. An Ontological Interface for Software Developers to Select Security Patterns. *19th International Workshop on Database and Expert Systems Application, 2008. DEXA '08*, 297–301.

FERNANDEZ, E.B. 2011. Using security patterns to develop secure systems. In: *Software engineering for secure systems. Industrial and research perspectives*. 16–31.

FERNANDEZ, E.B. 2013. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns.* Wiley Publishing.

FERNANDEZ, E.B., WASHIZAKI, H., AND YOSHIOKA, N. 2008. Abstract Security Patterns. *Proceedings of the 15th Conference on Pattern Languages of Programs*, ACM, 4:1–4:2.

FRIEDENTHAL, S., MOORE, A., AND STEINER, R. 2008. *A Practical Guide to SysML: Systems Modeling Language.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

FRIEDRICH, J., HAMMERSCHALL, U., KUHRMANN, M., AND SIHLING, M. 2009. Das V-Modell XT. In: *Das V-Modell® XT*. Springer Berlin Heidelberg, 1–32.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. 1995. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

HAMID, B., GEISEL, J., ZIANI, A., BRUEL, J.-M., AND PEREZ, J. 2013. Model-Driven Engineering for Trusted Embedded Systems Based on Security and Dependability Patterns. In: F. Khendek, M. Toeroe, A. Gherbi and R. Reed, eds., *SDL 2013: Model-Driven Dependability Engineering*. Springer Berlin Heidelberg, 72–90.

HAMID, B. AND PERCEBOIS, C. 2014. A Modeling and Formal Approach for the Precise Specification of Security Patterns. In: J. Jürjens, F. Piessens and N. Bielova, eds., *Engineering Secure Software and Systems*. Springer International Publishing, 95–112.

HAMID, B., PERCEBOIS, C., AND GOUTEUX, D. 2012. A Methodology for Integration of Patterns with Validation Purpose. *Proceedings of the 17th European Conference on Pattern Languages of Programs*, ACM, 8:1–8:14.

HASHEMINEJAD, S.M.H. AND JALILI, S. 2009. Selecting Proper Security Patterns Using Text Classification. *International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009*, 1–5.

HASHEMINEJAD, S.M.H. AND JALILI, S. 2012. Design patterns selection: An automatic two-phase method. *Journal of Systems and Software 85*, 2, 408–424.

ISO/IEC 27000. 2014. *Information technology — Security techniques — Information security management systems — Overview and vocabulary.* .

ISO/IEC 27001. 2013. *Information technology — Security techniques — Information security management systems — Requirements.* .

ISO/IEC 27005. 2011. *Information technology — Security techniques — Information security risk management.* .

MCGRAW, G. 2006. *Software Security: Building Security In.* Addison-Wesley Professional.

GÉRARD, S. 2015. MDE with Papyrus, novelties and beyond. *EclipseCon Europe 2015.* https://www.eclipsecon.org/europe2015/session/mde-papyrus-novelties-and-beyond-sponsored-cea-list.

MRAIDHA, C., TUCCI-PIERGIOVANNI, S., AND GERARD, S. 2011. Optimum: A MARTE-based Methodology for Schedulability Analysis at Early Design Stages. *SIGSOFT Softw. Eng. Notes 36*, 1, 1–8.

RADERMACHER, A., HAMID, B., FREDJ, M., AND PROFIZI, J.-L. 2013. Process and Tool Support for Design Patterns with Safety Requirements. *Proceedings of the 18th European Conference on Pattern Languages of Program*, ACM, 8:1–8:16.

RUMBAUGH, J., JACOBSON, I., AND BOOCH, G. 2004. *Unified Modeling Language Reference Manual, The (2Nd Edition).* Pearson Higher Education.

SCHNEIER, B. 2000. *Secrets & Lies: Digital Security in a Networked World.* John Wiley &amp; Sons, Inc., New York, NY, USA.

SINDRE, G. AND OPDAHL, A.L. 2000. Eliciting security requirements by misuse cases. *37th International Conference on Technology of Object-Oriented Languages and Systems, 2000. TOOLS-Pacific 2000. Proceedings*, 120–131.

STONEBURNER, G., GOGUEN, A., AND FERINGA, A. 2002. Risk management guide for information technology systems. *Nist special publication 800*, 30, 800–30.

UZUNOV, A.V., FERNANDEZ, E.B., AND FALKNER, K. 2012. Securing distributed systems using patterns: A survey. *Computers & Security 31*, 5, 681–703.

WEISS, M. AND MOURATIDIS, H. 2008. Selecting Security Patterns that Fulfill Security Requirements. *16th IEEE International Requirements Engineering, 2008. RE '08*, 169–172.