# A Model-based Repository of Security and Dependability Patterns for Trusted RCES

Adel Ziani, Brahim Hamid, Jacob Geisel, Jean-Michel Bruel

# A Model-based Repository of Security and Dependability Patterns for Trusted RCES

Adel Ziani, Brahim Hamid, Jacob Geisel and Jean-Michel Bruel
IRIT - University of Toulouse
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
{ziani,hamid,geisel,bruel}@irit.fr

## Abstract

*The requirement for higher Security and Dependability (S&D) of systems is continuously increasing, even in domains traditionally not deeply involved in such issues. Nowadays, many practitioners express their worries about current S&D software engineering practices. New recommendations should be considered to ground this discipline on two pillars: solid theory and proven principles. We took the second pillar towards software engineering for embedded system applications, focusing on the problem of integrating S&D by design to foster reuse.*

*Model driven approaches combined with patterns can be extremely helpful to deal with these strong requirements. In this work, we present a framework for trusted Resource Constrained Embedded Systems (RCES) development by design, by defining both a model to represent S&D pattern language and an architecture for development tools. The implementation of a repository of S&D patterns and their complementary property models is discussed in detail.*

**Keywords.** *Resource Constrained Embedded Systems, Security, Dependability, Repository, Pattern, Metamodel, Model-Driven Engineering.*

## 1 Introduction

Non-functional requirements such as Security and Dependability (S&D) [14] become more and more important as well as more and more difficult to achieve, particularly in Resource Constrained Embedded Systems (RCES) [18, 7]. Such systems come with a large number of common characteristics, including real-time, temperature, memory, computational processing power and/or energy consumption constraints, as well as security, dependability and efficiency requirements. The integration of S&D features requires the availability of both application domain specific knowledge and S&D expertise at the same time. Hence capturing and providing this expertise by means of S&D patterns can enhance embedded systems development.

In our previous work [8], we studied pattern modeling frameworks and we proposed methods to model security and dependability aspects in patterns and to validate whether these still hold in RCES after pattern application. The question remains at which stage of the development process to integrate S&D patterns. In our work, we promote a new discipline for system engineering using a pattern as its first class citizen: Pattern-based System Engineering (PBSE). PBSE addresses challenges similar to those studied in software engineering focusing on patterns and from this viewpoint addresses two kind of processes: the process of *pattern development* and *system development with patterns*. In order to interconnect these two processes, we promote a structured model-based repository of S&D patterns and property models. Therefore, instead of defining new modeling artifacts, that usually are time and efforts consuming as well as errors prone, the system developer merely needs to select appropriate patterns from the repository and integrate them in the system under development.

According to Bernstein and Dayal [2], a repository is a shared database of information on engineered artifacts. They introduce the fact that a repository has (1) a *Manager* for modeling, retrieving, and managing the objects in a repository, (2) a *Database* to store the data and (3) *Functionalities* to interact with the repository. In our work, we go one step further: a model-based repository to support the specifications, the definitions and the packaging of a set of modeling artifacts to assist developers of trusted applications for resource constrained embedded systems. In this paper, we propose an Model Driven Engineering (MDE) approach to pro-

duce a repository of modeling artifacts (S&D patterns, resource models, S&D models,...). S&D patterns, derived from and associated with domain specific models, will help developers to integrate application building blocks with S&D building blocks. The pattern building process, clearly related to domain models, is the reason why we advocate the use of a model-based repository.

The contribution of this work is threefold:

- Design framework: we propose a set of modeling languages to specify a pattern for security and dependability associated with property models independent from end-development applications and a structured repository.
- Tooling: we propose a Connected Data Objects (CDO)[1] based implementation of the repository and a set of EMF tree-based editors to create patterns and the required libraries,
- Validation: apply in practice to a resource constrained embedded system (RCES) in the context of TERESA project [18, 7].

The rest of this paper is organized as follows. Section 2 describes some modeling frameworks dealing with repository of models and patterns. In Section 3, we discuss the concepts as the basis for the definition of the proposed modeling framework. Then, in Section 4, we detail the specification of modeling languages for patterns and properties. Section 5 describes the implementation of the repository. Section 6 describes the usage of the defined modeling framework in the context of trusted RCES applications through the railway case study. Section 7 provides a preliminary evaluation of the approach along ISO-9126's quality-in-use dimensions. Finally, Section 8 concludes and draws future work directions.

## 2 Related Work

In Model-Driven Development (MDD), model repositories [12, 5, 2] are used to facilitate the exchange of models through tools by managing modeling artifacts. For instance, as presented in the standard ebXML [13] and the ebXML Repository Reference Implementation[2], a service repository can be seen as a metadata repository that contains metadata on location information to find a service. In [12], the authors propose a reusable architecture decision model for setting-up model and metadata repositories. In addition, some helpers are included in the product for selecting a basic repository technology, choosing appropriate repository metadata and selecting suitable modeling levels for the model information stored in the repository.

The ReMoDD (Repository for Model Driven Development) project [5] focuses on MDD for reducing the effort of developing complex software by raising the level of abstraction at which software systems are developed. This approach is based on a repository containing artifacts (e.g. documented MDD case studies, modeling exercises and problems) that support research and education in MDD. Another issue is graphical modeling tool generation as studied in the GraMMi project [16]. GraMMi's Kernel allows to manage persistent objects. The kernel aims at converting the objects (models) in an understandable form for the user via the graphical interface. Recently, the MORSE project [9] proposes a Model-Aware Service Environment repository addressing two common problems in MDD systems: traceability and collaboration.

Pattern repositories are introduced to comprehensibly explain pattern classification. The organization of patterns in a repository allows to discover the relationships among them and to facilitate the selection of the most appropriate one. The repository may have a structure in order to optimize the access (selecting patterns with criteria and publishing new ones). However it is still difficult to find patterns solving particular security and/or dependability problems, caused by the lack of a dedicated classification scheme for S&D patterns and the lack of precise specification languages for patterns. Some classifications are based on security concepts. For example, ISO/IEC 13335 [11] provides a definition of the five key concepts: security, confidentiality, integrity, availability and accountability. A pattern classification scheme based on these domain level concepts, will facilitate pattern mining and pattern navigation. An implicit approach for supporting developers in choosing patterns suitable for a given problem is described in [3]. In this vision, the repository contains patterns that are selected depending on the history of their use regarding decisions made by other developers to deal with related problems. An ontological approach for selecting design patterns is proposed in [6] to facilitate the understanding and reuse during software development.

## 3 The Framework for Trusted RCES Applications

In our work, we promote a new discipline for system engineering using a pattern as its first class citizen: Pattern-based System Engineering (PBSE). PBSE addresses challenges similar to those studied in software

[1] http://www.eclipse.org/cdo/

[2] `http://ebxmlrr.sourceforge.net/`

engineering. Therefore, PBSE focuses on patterns and from this viewpoint addresses two kind of processes: the process of pattern development and system development with patterns. The main concern of the first process is designing patterns for reuse and the second one is finding the adequate patterns and evaluating them with regard the system-under-development's requirements. In order to interconnect these two processes, we promote a structured model-based repository of S&D patterns and property models.

### 3.1 An S&D Pattern Repository

Usually, a pattern refers to a template which describes solutions for commonly occurring problems. Unfortunately, most of them are expressed in a textual form, as informal indication,s on how to solve problems. The same limitation applies to S&D patterns [8]. In this work we deal with a modeling and development framework to support the specifications, the definitions and the packaging of a set of S&D patterns and their related models to assist the developers of trusted applications targeting several domains in RCES. Achieving this goal requires to get (a) a common representation of S&D models and artifacts for several domains, (b) a flexible structure, (c) guidelines for platform specific implementations and (d) guidelines to guarantee the correctness of the models and artifacts integration step.

Security and dependability patterns are not only defined from a platform independent viewpoint (i.e. they are independent from the implementation), they are also expressed in a consistent way with domain specific trust models. Consequently, they will be much easier to understand and validate by application designers in a specific area. That is: (1) S&D experts can make patterns publicly available, (2) S&D patterns can be used by RCES developers in other companies, (3) S&D patterns can be derived from and associated with domain specific models and (4) S&D patterns help application developers to integrate application building blocks with S&D building blocks. This is the reason why we advocate the use of a model-based repository, where patterns are clearly related to domain models.

Concretely, our repository system is a structure that stores S&D patterns, property models and relationships among them, coupled with a set of tools to manage, visualize, export and instantiate these artifacts in order to use them in engineering processes (see Fig. 1). We start with a set of definitions and concepts that might prove useful in understanding our approach.

Adapting the definition of [17], we propose the following:

**Definition 1 (S&D Pattern.)** *A security and dependability pattern describes a particular recurring security and/or dependability problem that arises in specific contexts and presents a well-proven generic scheme for its solution.*

**Definition 2 (S&D Pattern System.)** *We define a security and dependability pattern system as a modeling artifact system where its constituent parts are security and dependability patterns, its referenced property models and their relationships.*

**Definition 3 (Domain.)** *We define a domain as a field or a scope of knowledge or activity that is characterized by the concerns, methods, mechanisms,... employed in the development of a system. The actual clustering into domains depends on the given group/community implementing the target methodology.*

In this context, we use the pattern classification of Riehle and Buschmann [15, 4], which is (1) *System Patterns*, *Architectural Patterns*, *Design Patterns* and *Implementation Patterns* to classify these patterns. In addition, we recommend that a domain should include knowledge about: protocols, processes, methods, techniques, practices, OS, HW systems, measurement and certification related to the specific domain. For example, in a Railway domain, it's recommended to use HMAC as a mechanism for the realization of the secure communication pattern.

### 3.2 The Repository System Framework

The repository presented here is a model-based repository of S&D patterns. It constitutes one of the most important key elements in the engineering process for resource constrained embedded systems. Now, we introduce our repository system framework as a workflow components (see Fig. 1).

- *Repository Generation.* The core of the framework is the definition of specification languages to support the design of S&D pattern, S&D property and resource property models. These languages are obtained by using metamodeling approach as illustrated in the next sections.
- *Repository Implementation.* Once these specification languages have been defined, it is possible to develop a repository. The development of such a repository is based on transformation techniques and the availability of MDE tools. As we shall see in Section 4.4, the MDE transformation techniques are use for the automatic generation of the repository structure.

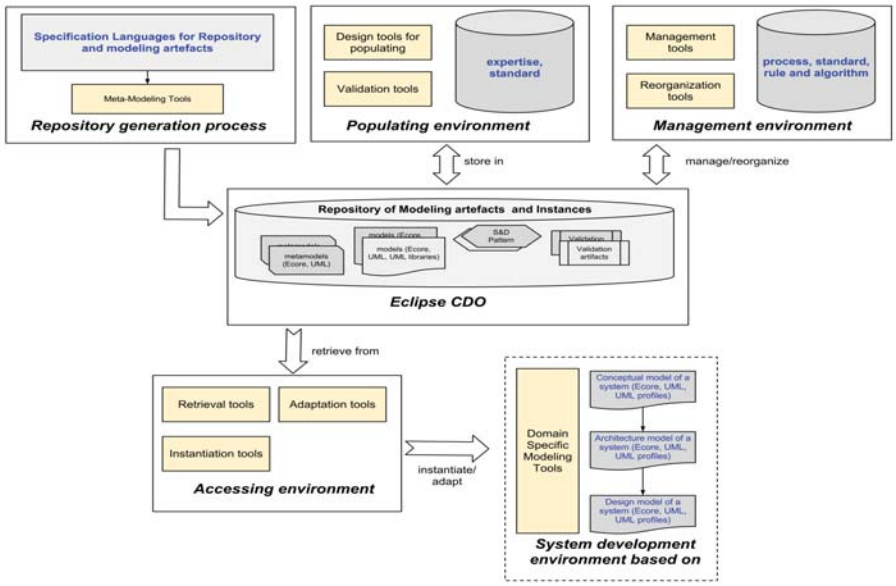


**Figure 1. The proposed framework for trusted RCES applications based on a repository**

- *Repository Populating.* The development environment associated to the repository is based on the specification languages for defining S&D patterns and property models. It is composed of tools for the design, the instantiation and the validation of these modeling artifacts. These tools support basic features including the storage in the repository.

- *Repository Managing.* In this part, we focus on the management of the repository content. We provide software to manage relationships among S&D patterns specifications, and between S&D patterns and their related property models. Moreover, we support basic features such as artifact management and user management.

- *Repository Accessing.* Here we focus on the repository accessing techniques providing simple interfaces one for each artifact kind (S&D pattern and property model). By accessing the repository, we provide features based on model transformation techniques to adapt the model of the pattern to the target development environment. In addition, some facilitators are provided guiding the end-user choices on S&D patterns which can be used to satisfy the system-under-development's requirements.

# 4 Specification Languages

In the proposed vision, a *pattern* is the key element for building trusted applications for resource constrained embedded systems. A *pattern* uses, in its turn, a *property model* for specifying its properties. We begin with an overview of the repository structure model.

## 4.1 A Model-based Repository of S&D Patterns Model

The specification of the structure of the repository, as visualized in Fig. 2 is based on the organization of its content, mainly the S&D patterns, the property models and their relationships. For instance, an S&D pattern is linked with the other patterns and associated with S&D and resource property models using a predefined set of reference kinds.

## 4.2 Generic Property Modeling Language (GPRM)

The metamodel of property [19] captures the common concepts of the two main concerns of trusted RCES applications: *Security*, *Dependability* and *Resource* on the one hand and *Constraints* on these properties on the other hand. The libraries of properties and constraints includes units, types, categories and operators. For instance, security and dependability attributes [1] such as authenticity, confidentiality and

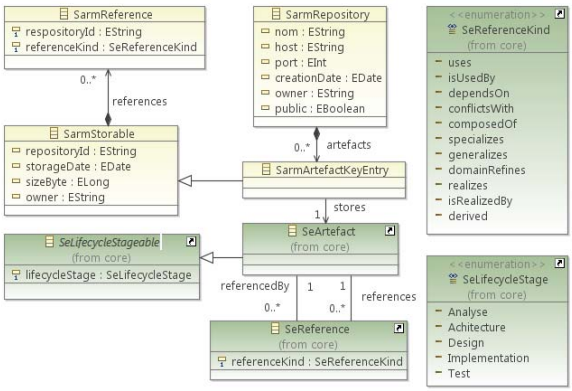


**Figure 2. The (simplified) Model-based repository Model**

availability are defined as categories. These categories require a set of measures types (degree, metrics, ...) and units (boolean, float,...). For that, we instantiate the appropriate type library and its corresponding unit library. These models are used as external model libraries to type the properties of the patterns. Especially during the design of the pattern (see next sections) we define the properties and the constraints using these libraries.

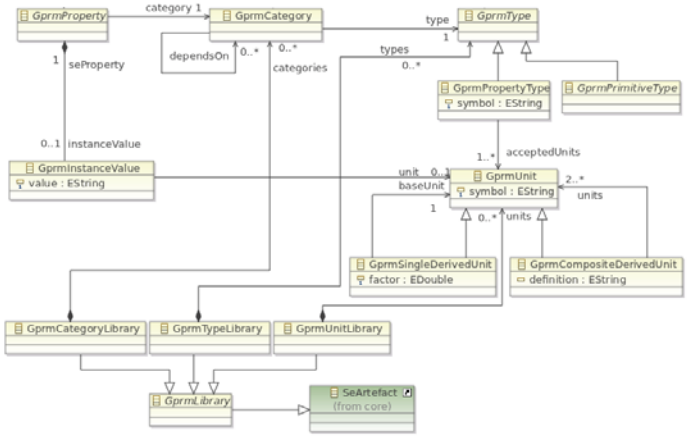


**Figure 3. The (simplified) GPRM Metamodel**

### 4.3 Pattern Specification Metamodel

The System and software Pattern Metamodel (SEPM) [8], as visualized in Fig. 4, is a metamodel defining a new formalism for describing patterns. Such a formalism describes the concepts (and their relations) required to capture all the facets of patterns. In addition to defining pattern concepts, the pattern metamodel provides an instantiation mechanism that enables the separation on the one hand between the domain independent (DIPM) and domain specific (DSPM) and on the other hand between the design life cycle stage. We illustrate the usage of the SEPM for specifying a pattern at domain independent and at domain specific with the example of secure communication pattern. For the sake of simplicity, we only specify the interfaces and the properties.

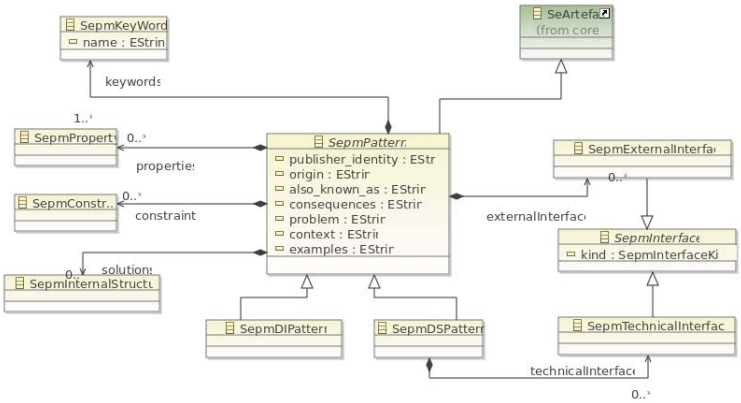


**Figure 4. The (simplified) SEPM Metamodel**

For the DIPM, a subset of the functions provided by the external interfaces are:

- $send(P, ch, m) : P$ sends message $m$ to $Q$ on the channel $ch$,
- $receive(P, ch, m) : P$ receives and accepts message $m$ from $Q$ on the channel $ch$,

with $P, Q \in \{C, S\}$, $ch(C, S) = ch(S, C)$ denoting the communication channel of client $(C)$ and server $(S)$, and $m$ a message. The properties definitions require the availability of the required property libraries. Here, we specify an S&D property: "authenticity of sender and receiver". To type the category of this property we use a category from the once defined in the S&D category library: "Authenticity".

The DSPM modeling level is a refinement of the DIPM. In our example, we use HMAC protocol as a mechanism related to the application domain to refine the secure communication pattern. Therefore, the external interfaces and properties refine the ones defined at DIPM introducing domain specific concepts related to the HMAC mechanism. For example, a

function $send(P, ch, m, mac)$ refines the DIPM function $send(P, ch, m)$ adding the $mac$ as the generated message authentication code.

In addition to the refinement of the concepts used at DIPM, the DSPM involves developing technical internal interfaces as a set of functions related to the use of HMAC to refine the secure communication pattern. A subset of the functions provided by the internal interfaces are:

- $generateAH(P, keymac, m, mac)$ to prepare an appropriate authentication header (MAC) for the message $m$.
- $checkAH(P, keymac, m, mac)$ verifies that the message authentication code for $m$ is correct and originates from $Q$.

with $P, Q \in \{C, S\}$, $ch(C, S) = ch(S, C)$ denoting the communication channel of client ($C$) and server ($S$), $m$ a message and $mac$ the generated message authentication code. The interfaces (external and internal) with the required libraries of properties are then used for developing the pattern properties. Here, in addition to the refinement of the S&D properties identified in the DIPM, at this level we identify some related resource properties, e.g. the size of the cryptographic key.

### 4.4 CDO Repository Implementation

Gaya is a repository platform to store the modeling artifact specifications and instances through the APIs. The structure is derived from the repository model and implemented using Java and the Eclipse CDO Server technology. Then, we provide the environment for the use of Gaya repository.

The server part is responsible for managing and storing the data, and provides a set of features to interact with the repository content. As shown in Fig. 8, thanks to UML component diagram the server part is composed of two components: (1) *GayaServer* providing the implementation of the common API and (2) *Gaya-MARS* providing the storage mechanisms. The server part of the repository is provided as an Eclipse plugin that will handle the launch of a CDO server defined by a configuration file. The client part is responsible for populating the repository and for using its content. For this, we identify a set of CDO-based clients as depicted in Fig. 8. These clients provide APIs interfaces (APIs) for applications in order to create the modeling artifact, in order to use them and in order to manage the repository. These APIs are *Gaya4Pattern* (implements the API4PatternDesigner), *Gaya4Property* (implements the API4PropDesigner), *Gaya4Admin* (implements API4Admin) and *Gaya4SystemDeveloper* (implements the API4PatternUser).

### 4.5 Repository Populating - Design Tools

Tiqueo is an EMF generated tree-based editor for specifying models of properties and constraints. In addition, Tiqueo provides some features to create a library for reusable objects, like the types and units which allows us to use the libraries in a domain independent manner. Furthermore, Tiqueo includes mechanisms to validate the conformity of the property and constraint library to the GPRM metamodels and to publish the results into the Gaya repository using the *Gaya4Property* API.

For a DSPM pattern, the Arabion design environment is presented in Fig. 5. There is a design palette on the right, a tree view of the project on the left and the main design view in the middle. The design palette is updated regarding the one used for a DIPM pattern to display suitable design entities for building patterns at DSPM. These entities are internal interfaces, domain and refinement. DS patterns are built by refining DI patterns. In our example, the SecurityCommunicationLayer@Design pattern refines the SecureCommunication@Design pattern for the railway domain using the HMAC mechanism. Like DI patterns, the DS pattern has external interfaces. Furthermore, Arabion includes mechanisms to validate the conformity of the pattern to the SEPM metamodel and to publish the results to the repository using the repository interfaces (*Gaya4Pattern* API).

### 4.6 Repository Managing

For the repository management, we provide a set of facilities for the repository organization allowing the enhancement of its usage using the *Gaya4Admin* API. We provide also basic features such as user, domain and artifact management. Moreover, we provide features to support the management of the relationships among artifacts specifications and between artifacts specifications and their complementary models. For instance, as visualized in Fig. 6, a pattern is linked with other patterns and associated with S&D and resource property models using a predefined set of reference kinds such as those proposed in the SARM metamodel.

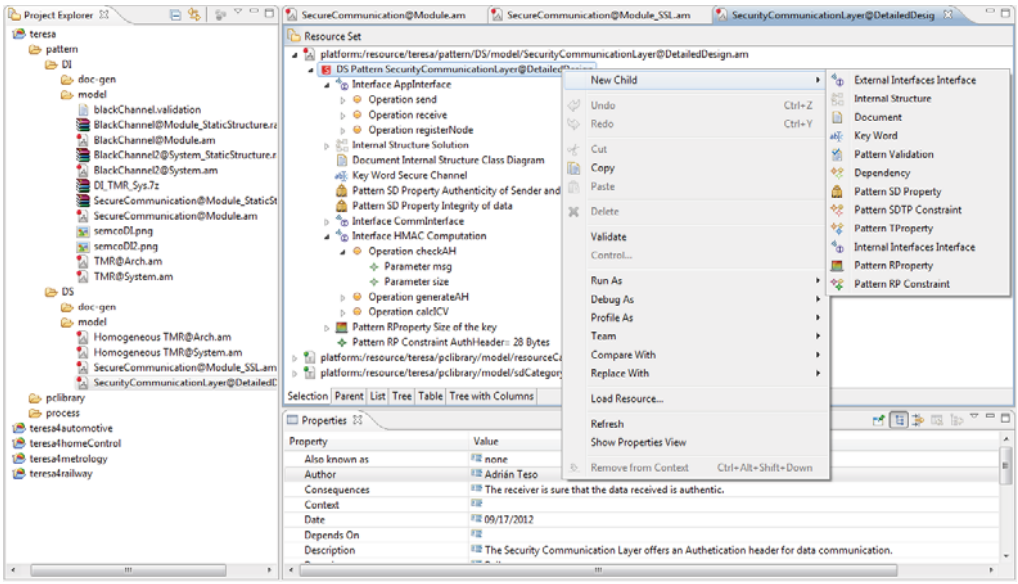


**Figure 5. Designing a Patten**

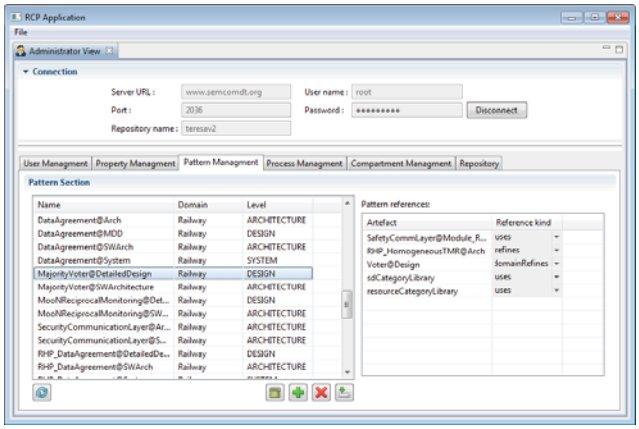


**Figure 6. Repository Management and re-organization**

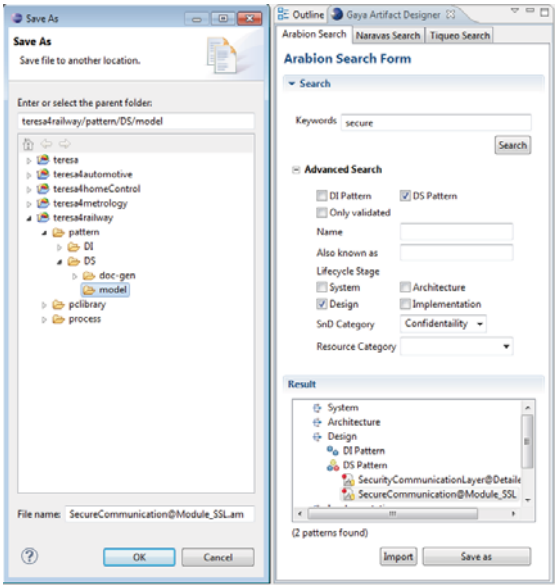


**Figure 7. Access Tool/ Pattern Instantiation**

### 4.7 Repository Accessing (Retrieval)

## 5 Architecture and Implementation Tools

Using the proposed metamodels, ongoing experimental work with the *Semcomdt*[3] tool (IRIT editors and platform as Eclipse plugins), as visualized in Fig. 8.

The tool uses the *Gaya4SystemDeveloper* API for the search/selection/sorting of the patterns which is used during a pattern and a system development process. The Tool includes features for exportation and instantiation as dialogs, mainly those based on model transformation techniques to adapt the model of the pattern to the target development environment. For the use case presented in the next section, the translation is implemented to target Rhapsody UML [10].

[3]http://www.semcomdt.org

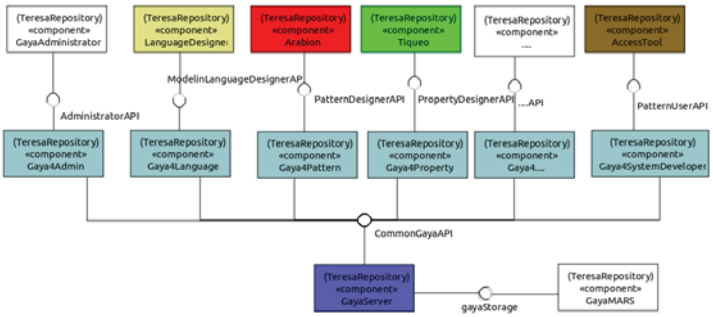


**Figure 8. An Overview of the Tools Components**

Moreover, the tool provides a set of facilities to help selecting appropriate patterns including *key word* search and *life cycle stage* search, as shown in the right part of Fig. 7. The results are displayed in search result tree as System, Architecture, Design and Implementation patterns. For example, the right part of Fig. 7 shows that there is a DI pattern at design level targeting the *Confidentiality* S&D property[4], named communication and has a keyword *secure*. In our case, we select the *Secure Communication* pattern for instantiation providing the necessary information, including the project path and instance name (see the left part of Fig. 7). In addition, the tool includes dependency checking mechanisms. For example, a pattern can't be instantiated, when a property library is missing, an error message will be thrown.

## 6 Application of a Model-based Repository of S&D Patterns to Industrial Case Studies

In the context of the TERESA project[5], we evaluated the PBSE approach to build two demonstrators combining MDE and a model-based repository of S&D patterns and their related property models: (1) Railway *Safe4Rail* application in charge of the emergency brake of a railway system and (2) Metrology *SmartMeterGateway* application in charge of connecting Smart Metering devices. The TERESA repository contains so far (as of March 2013):

- *Users.* 5 organizations and 10 users.
- *Property Libraries.* 55 property model libraries, including 12 Unit Libraries, 23 Type Libraries, 20 Category Libraries.
- *Patterns.* 59 patterns, which are 20 System Level patterns (12 DIPM, 8 DSPM), 25 Architecture Level patterns (9 DIPM, 16 DSPM) and 14 Design Level patterns (3 DI, 11 DSPM)

We used the design tools (Tiqueo and Arabion editors) to populate the Gaya repository and the Gaya manager tool to set the relationships between the patterns. Fig. 9 depicts an overview of the railway S&D pattern language.

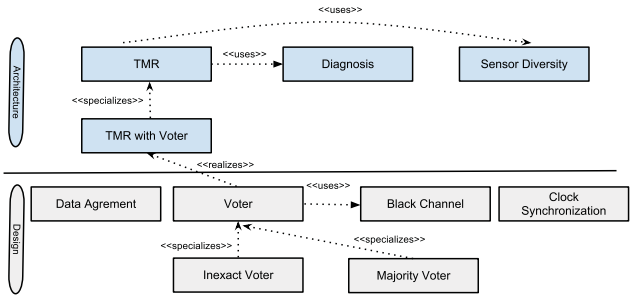


**Figure 9. Railway Pattern Language - Overview**

Once the repository[6] is available, it serves our underlying PBSE for trust engineering process. In this process model, the developer starts by requirements engineering/specification, followed by system specification. For each phase, the system developer executes the search/select from the repository to instantiate appropriate patterns in its modeling environment using the Access tool and then integrates them in its models following an incremental process. Moreover, the system developer can use the pattern designer tool (Arabion) to develop their own solutions when the repository fails to deliver appropriate patterns at this phase.

[4]In our modeling, this means that the pattern has a property with a confidentiality category type.

[5]http://www.teresa-project.org/

[6]The repository system populated with S&D Patterns.

The process flow for the example of the Safe4Rail application development can be summarized with the following steps:

- Once the requirements are properly captured and imported into the development environment, for instance Rhapsody[7], the repository may suggest possible patterns to meet general or specific S&D needs (according to requirements and application domain): e.g. if the requirements contain the keywords *Redundancy* or *SIL4*, a suggestion could be to use a *TMR* pattern at architecture level. In addition, some diagnosis techniques imposed by the railway standard may be suggested: TMR (suggested by the tool), Diagnosis techniques (suggested by the tool) and Sensor Diversity (searched by the System Architect).

- Based on the selected patterns, the repository may suggest related or complementary patterns. For instance, if the TMR has been integrated , the following patterns may be proposed, thanks to the repository structure, in a second iteration, for instance at design phase: Data Agreement, Voter, Black Channel Clock Synchronization.

## 7 Assessment

This section provides a preliminary evaluation of the approach along ISO-9126 's quality-in-use dimensions, i.e. effectiveness, productivity, safety and satisfaction. Eleven TERESA members participated. They were handed out a sheet with instructions for each task (e.g., what properties to specify and what patterns de develop, when to take note of the time, etc.).

The study was divided into three tasks. Before they started, a general description of the aim of the study was given (30'). Some running examples was introduced to them. After these two tasks, achieved during the TERESA MDE workshop in Toulouse (April 2012), a 6-months evaluation was conducted. All the subjects were already familiarized with MDE, S&D patterns and Eclipse, though some did not know some of the companion plugins (e.g. Acceleo). Hence, the generation of documentation was not part of the evaluation. The procedure includes five tasks: SEMCO plug-in installation, property models development, pattern development, patterns instantiation and patterns integration. Here, we present only the users satisfaction.

Satisfaction is the capability of the software product to satisfy its users. In this case, the product is the repository of S&D patterns engine, and its ability to develop a trusted RCES application. We asked participants to give scores from 1 to 5 (5 is the best) and comments. We first evaluated the perceived usefulness of the solution itself (items 1-5). Next, we focus on the tool-suite as a mean to build the modeling artifacts. We separately collected the satisfaction along the four tasks (items 6-14). Finally, we want also to measure the willingness to use repository of modeling of S&D patterns in the future in the related activities (items 15-20). The following table depicts an overview of the results of our experiment.

| Item | Mean | St. Dev. |
|---|---|---|
| 1. I think 'model based repository' is a good idea | 4.90 | 0.18 |
| 2. I think 'model based repository' helps to keep focus without being distracted by other aspect of software engineering | 4.40 | 0.48 |
| 3. I think 'model based repository' are useful for defining meaningful 'units of solution' | 4.50 | 0.48 |
| 4. I think 'model based repository' save me time to develop S&D Embedded Systems | 4.40 | 0.50 |
| 5. I think 'model based repository' avoids re-inventing existing solutions | 4.60 | 0.48 |
| 6. I think the installation of the SEMCO plug-in is easy | 4.10 | 0.48 |
| 7. I think repository populating tools are easy to use | 3.80 | 0.54 |
| 8. I think repository access tools are easy to use | 4.10 | 0.68 |
| 9. I think it is easy for me to develop new S&D patterns | 3.50 | 0.36 |
| 10. I think it is easy for me to develop new properties models | 3.80 | 0.70 |
| 11. I think S&D patterns instantiation is easy to use | 3.80 | 0.64 |
| 12. I think properties models instantiation is easy to use | 3.50 | 0.64 |
| 13. I think S&D patterns integration is easy to use | 3.40 | 0.60 |
| 14. I think properties models integration is easy to use | 4.60 | 0.60 |
| 15. I would like to develop S&D patterns in the future | 3.80 | 0.56 |
| 16. I would like to develop properties models in the future | 4.10 | 0.68 |
| 17. I would like to install other SEMCO plugins in the future | 3.50 | 0.54 |
| 18. I would like to exchange SEMCO in the future | 3.60 | 0.56 |
| 19. I would like to customize some SEMCO plugins in the future | 3.60 | 0.76 |
| 20. I would like to extend some SEMCO features in the future | 3.70 | 0.83 |

**Figure 10. Satisfaction Results from 1 (total disagreement) to 5 (total agreement).**

These results seem to suggest that subjects like the notion of model-based repository of patterns as a way to speed the development of S&D applications by design (e.g. reuse existing solution through pattern), and in so doing, improving focus on tough tasks (e.g. implementation). However, pattern integration stands up as the main stumbling block for pattern-based system development adoption. More to the point, if we consider that the subjects were programmer natives (i.e. accustom to use programming language for security engineering). Specifically, users tend to overlook the four rules that govern pattern-based system development (i.e. (1) each pattern must be specified domain-application independently, (2) every pattern should instantiated in the context of the system-under-development, (3) every pattern should instantiated in the target domain-development environment, and (4) more than one pattern is required to fulfill one S&D property). They also point out the importance of the automatic search for the user to derive those 'S&D patterns' from the requirements analysis.

[7]Rhapsody is the modeling environment used by Ikerlan Centre engineers.

## 8 Conclusion

In this paper, we target the development of a model-based repository of S&D patterns that follows the MDE paradigm. Our framework is based on metamodeling techniques that allow to specify the S&D patterns at different levels of abstraction and an operational architecture of the repository. Furthermore, we walk through a prototype with EMF editors and a CDO-based repository supporting the approach. Currently the tool suite named *semcomdt* is provided as Eclipse plugins.

The approach presented here has been evaluated in in the context of the TERESA project for a repository of S&D patterns and property models targeting RCES applications. First evidences indicate that users are satisfied with the notion of 'model-based repository of S&D patterns'. The approach paves the way to let users define their own road-maps upon the PBSE methodology. First evaluations are encouraging with 85% of the subjects being able to complete the tasks. However, they also point out two main challenges: pattern integration and automatic search for appropriate patterns.

As future work, we plan to perform additional case studies to evaluate both the expressiveness and usability of the methodology, the DSLs and the tools. We will seek new techniques for the automation of the search and instantiation of models and patterns. Our vision is for 'S&D patterns' to be inferred from the browsing history of users built from a set of already developed applications.

## References


[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1:11–33, 2004.

[2] P. A. Bernstein and U. Dayal. An Overview of Repository Technology. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 705–713. Morgan Kaufmann Publishers Inc., 1994.

[3] R. Birukou, E. Blanzieri, P. Giorgini, and M. Weiss. Facilitating pattern repository access with the implicit culture framework. In *Proceedings of "EuroPLoP 2007"*, 2007.

[4] G. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: a system of patterns*, volume 1. John Wiley and Sons, 1996.

[5] R. B. France, J. M. Bieman, and B. H. C. Cheng. Repository for Model Driven Development (ReMoDD). In *MoDELS Workshops'06*, pages 311–317, 2006.

[6] R. Girardi and A. N. Lindoso. An ontology-based knowledge base for the representation and reuse of software patterns. *ACM SIGSOFT Software Engineering Notes*, 31(1):1–6, 2006.

[7] B. Hamid, N. Desnos, C. Grepet, and C. Jouvray. Model-based security and dependability patterns in RCES: the TERESA approach. In *1st International Workshop on Security and Dependability for Resource Constrained Embedded Systems (SD4RCES)*, 2010.

[8] B. Hamid, S.Gurgens, C. Jouvray, and N. Desnos. Enforcing S&D Pattern Design in RCES with Modeling and Formal Approaches. In J. Whittle, editor, *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)*, volume 6981, pages 319–333. Springer, octobre 2011.

[9] T. Holmes, U. Zdun, and S. Dustdar. MORSE: A Model-Aware Service Environment, 2009.

[10] IBM. Rhapsody UML. http://www-142.ibm.com/software/products/us/en/ratirhapfami/.

[11] ISO/IEC. Standard: ISO/IEC 13335, 2004.

[12] C. Mayr, U. Zdun, and S. Dustdar. Reusable Architectural Decision Model for Model and Metadata Repositories. In *FMCO*, pages 1–20, 2008.

[13] Oasis. ebXML: Oasis Registry Services Specification v2.5, 2003.

[14] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst.*, 3(3):461–491, 2004.

[15] D. Riehle and H. Züllighoven. Understanding and using patterns in software development. *TAPOS*, 2(1):3–13, 1996.

[16] C. Sapia, M. Blaschka, and G. Höfling. GraMMi: Using a Standard Repository Management System to Build a Generic Graphical Modeling Tool. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8*, HICSS '00, pages 8058–. IEEE Computer Society, 2000.

[17] M. Schumacher. *Security Engineering with Patterns - Origins, Theoretical Models, and New Applications*, volume 2754 of *Lecture Notes in Computer Science*. Springer, 2003.

[18] TERESA Consortium. TERESA Project. http://www.teresa-project.org/.

[19] A. Ziani, B. Hamid, and S. Trujillo. Towards a Unified Meta-model for Resources-Constrained Embedded Systems. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 485–492. IEEE, 2011.