Software Engineering Techniques Applied to AmI: Security Patterns

Francisco Sanchez-Cid, Antonio Muñoz, Daniel Serrano, M.C. Gago
Computer Science Department
University of Málaga. Spain.
{amunoz,cid,mcgago,serrano}@lcc.uma.es

Abstract. The realization of the Ambient Intelligence concept entails many important challenges, but the most important barriers to this realization is the lack of adequate support for security. In this paper we present a conceptual model of our solution for building secure systems for AmI environments, taking as basis the concept of Security and Dependability (S&D) Pattern as a precise representation of validated S&D solutions and mechanisms. The main elements embedded in our solution framework (S&D library, monitoring interface and S&D Manager) are presented both conceptually, and also using a simple example scenario based on an hospital AmI environment.

1. Introduction

The EC Information Society Technologies Advisory Group (ISTAG), defines its vision of Ambient Intelligence as people surrounded by ubiquitous computers with intelligent and intuitive interfaces embedded in everyday objects around them, making the physical environments adapt and respond to users’ needs in an invisible way in order to provide anytime/anywhere access to information and services.

As the current computing scenarios evolve to ubiquitous and dependable systems, new connection and interoperability challenges appear into scene and arises unforeseen security issues. While a good system design is necessary to enforce user requirements, this is not enough when system requirements change daily as users and devices appear and disappear at run-time.

The main aim of this paper is to present SERENITY approach to provide a framework for applying Security Engineering techniques to develop secure systems using the expertise of security engineers.

Several approaches have been introduced in order to capture the specialized expertise of security engineers and make it available for automated processing providing the basis for automated synthesis and analysis of the security and dependability solutions of software-based systems. Our proposal is based on the enhanced concept of security patterns, widely accepted as a suitable approach to represent and reuse the precise knowledge of security experts.

Present security patterns (most of them security design patterns) contain information about the problem, the requirements and the consequences of applying it, and the possible interactions with other well-known patterns. Nevertheless, as they do not present a pre-defined structure, and most of the times are written in a non-technique language, they are not suitable for automatic processing. This makes them appropriate when designing a system (usually with a modelling language like UML) but useless when the condition is to analyse and apply them while the target system is running. SERENITY framework aims to provide a Security & Dependability patterns library, structured not only for its automated processing but also for the monitor of the completeness and correctness of the applied patterns at run-time.

2. Related work

The existing approaches for modelling security and dependability aspects in Ambient Intelligence (AmI) ecosystems go from the Components through the Frameworks and Agents to the enhanced concept of Pattern.

Components capture expertise in the form of reusable software elements that solve a certain problem in a certain context, having a set of well-defined interfaces and an associated description of their behaviour [1, 2, 3]. The main interest of component composition is to build new systems from their requirements by systematically composing reusable components. In general, this concept is not appropriate to represent S&D solutions because security mechanisms can not always be represented as units that can be connected to the rest of the system by means of well defined interfaces.

Middleware-based approaches capture expertise in the form of standard interfaces & components that provide applications with a simpler façade to access a set of specialized, powerful and complex capabilities. An important problem with middleware-based approaches is that the computational cost of the middleware components is far too high for computing devices with limited capabilities. Finally, the security infrastructure of middleware systems is usually restricted to authorization and access control issues [4, 5].

Application Frameworks [6, 7] have emerged as a...
powerful technology for developing and reusing middleware and application software. Because frameworks are application templates, they are not well suited to cope with scenarios with high degrees of heterogeneity, dynamism and unpredictability. Likewise, this approach does not support secure interoperation with external (and not trusted) elements.

The Agent paradigm is especially well-suited for highly distributed environments such as AmI scenarios thanks to properties like: autonomy, interaction, context-awareness and goal-oriented nature. In the case of modelling security aspects, agent paradigms are much more limited [8], since an agent is an independent entity by definition and many security solutions can not be represented as agents.

Regarding with Aspects, the main idea is to simplify the development of a complex system by “isolating” the different aspects that must be considered in its development (functionality, performance, reliability, security, etc.). Unfortunately, aspects are mainly an implementation technique and not suitable to provide and manage S&D solutions as a whole [9, 10].

The concept of security pattern was introduced to support the system engineer in selecting appropriate security or dependability solutions. However, most security patterns are expressed in textual form, as informal indications on how to solve some (usually organizational) security problem [11, 12, 13, 14]. Some of them do use more precise representations based on UML diagrams, but these patterns do not include sufficient semantic descriptions in order to automate their processing and to extend their use [15].

Perhaps the first and the most valuable contribution as pioneer in security patterns as we know them at present, is the work from Joseph Yoder and Jeffrey Barcalow proposing to adapt the object-oriented solutions to recurring problems of information security [14]. In their own words, seven patterns were presented to be used when dealing with application security. A natural evolution of [14] is the work presented by Romanosky in [16]. It takes into consideration new questions that arise when securing a networked application.

Going one step down in the abstraction scale, Eduardo B. Fernandez in his work about authorization patterns [15] combines for the first time the idea of multiple architectural levels with the use of design patterns. In [17] they propose the decomposition of the system into hierarchical levels of abstraction.

The same author et al. offers in [18] a good source to study the historical approaches that have been appearing in the scientific literature as pattern systems for security. Wassermann and Cheng present in [19] a revision of most of the patterns from [14] and [17] and categorise them in terms of their abstraction level. In order to facilitate the reuse of security knowledge, variations of the design pattern template given in [20] are included in order to better suit the presentation of security specific information. In the field of web application protection, [21] is the source of some patterns for Application Firewall and XML Firewall.

The special needs of secure-ware systems and the constantly changing context in which the systems are designed nowadays arise some new problems that have an starting solution in works like the one presented by Cheng et al in [13]. The proposal consist of a template for security patterns that takes into account essential information that has not been necessary in the general design patterns but appears as mandatory in the new security context. Taking Gamma et al. [20] security patterns as source, a new patterns library is created by the addition of new relevant information (e.g. Behaviour or Supported Principles are two new field to describe the security pattern) and altering some existing fields (e.g. Consequences is altered to convey a new set of possible consequences including confidentiality, usability, integrity, etc.).

Some security patterns have also been proposed for multiagent and Web Services systems. From the very beginning, the tendency has been to use the object oriented paradigm: in [22] an object oriented access control (OOAC) was firstly introduced as a result of consequently applying the object oriented paradigm for providing access controls in object and interoperable databases. Fernandez proposes in [23] some specific solutions oriented to web services: a pattern to provide authentication and authorization using Role-based access control (the so-called Security Assertion Coordination pattern) and a pattern for XML Firewalls. The Security Assertion Coordination pattern takes as source the abstract security architecture defined by SAML (the main standard to provide security for Web Services). However, a minor work has been done in the field of agent-based systems. A good start point is [24].

Some other authors have proposed ways to provide formal characterizations of patterns. The idea of precisely specifying a given class using class invariants and pre- and post-conditions for characterizing the behaviours of individual methods of the class is well known and is the basis of the design by contract (DbC) [25]. Evolutions of that approach have appeared and [26, 27] are some examples of that. Eden et al. [26] proposes to use logic formalism with an associated graphical notation to specify rich structural properties. However, it provides only limited support for specifying behavioural properties. In this sense, [27] has as main goal to preserve the design integrity of a system during its maintenance and evolution phases. Also Mikkonen in [28] focus his approach on behavioural properties. Here, data classes model role objects, and guarded actions (in an action system) model the methods of the roles.
3. Serenity Framework

3.1. Overview

Components, frameworks, middleware, and patterns have been proposed as means to simplify the design of complex systems and to capture the specialized expertise of security engineers and to make it available for non-expert developers. However, all these approaches have important drawbacks and limitations that hamper their practical use. Our approach aims at integrating the best of these approaches in order to overcome the problems that have prevented them to succeed individually. The main pillar to build solutions is the enhanced concept of security pattern can capture security expertise in a way that is more appropriate than other related concepts. Furthermore, because secure interoperability is an essential requisite for the widespread adoption of our model, trust mechanisms will be provided for patterns.

Therefore the objective of proposal is to work on the development and implementation of a tool-supported integral framework for security and dependability of software applications in AmI environments. Based on an integral model of security, our approach will consider not only static issues (related to secure systems development) but also dynamic aspects (related to monitoring and controlling the software at runtime). Validation mechanisms to deal with the dependences and the suitability of the applied solutions are used as well.

Due to the inherent dynamic nature of AmI environments, a flexible and runtime adaptable solution is required to confront the scenario. An appropriate solution should be succinctly described by means of an architecture composed of four elements: (i) a library of S&D Security Patterns available to bridge the gap between the new requirements and the security needs; (ii) a monitoring interface to control the execution of the security mechanisms; (iii) a negotiation interface to establish an appropriate framework configuration in case of: a change of the running context or an interaction with a new framework; (iv) and a context retrieval interface and an end user interface to detect new (and ever-changing) system requirements at runtime. Apart from that, in order to complete the architecture, a final element is necessary as well: a security and dependability manager, to coordinate the work of all the previous elements in order to look for one unique objective: apply sound security solutions to changing AmI environments.

3.1.1. S&D Patterns

The S&D patterns contain descriptions of reusable and validated S&D solutions that include a precise specification, along with applicability conditions. Ideally, it should also include trust mechanisms. SERENITY’s S&D patterns are precise specifications of validated security mechanisms materialised as files containing models that could be described using formal or non-formal languages (e.g. XML and logic-based language) to capture the expertise of security engineers with the objective of being used by automated means. S&D patterns include a precise behavioural description, references to the S&D properties provided, constraints on the context required for deployment, information describing how to adapt and monitor the mechanism, and trust mechanisms.

S&D patterns, along with the formal characterisation of their behaviour and semantics, are the basic building blocks of used by SERENITY’s integration schemes. These schemes are used to specify ways for systematically combining S&D patterns into systems composed of dynamically collaborating elements that operate in mobile, heterogeneous, and highly dynamic ICT infrastructures. In addition, integration schemes are an invaluable guide when two different elements/devices try to collaborate in an AmI context: each one of the elements will include an instance of the SERENITY framework, and therefore, a negotiation phase between these frameworks starts when trying to interact reciprocally. The integration schemes serve as a consult instance for the SERENITY framework in order to establish the way in which the elements should be connected, the security & dependability implications of the negotiation, and the possible security risks emerging with the process.

The description of an S&D Pattern with the objective of being used by automated means in dynamic environments requires different aspects to be described. Some of these elements are:

- **Creator:** Identity of the creator/provider of the pattern.
- **Trust mechanisms:** Digital signatures and other mechanisms to guarantee that the pattern description corresponds to the pattern/solution, that it has been produced by the creator, and that it has not been modified.
- **Provided Properties:** Reference to the properties provided by the pattern. Properties have a timestamp and refer to descriptions provided by the entity that defines the property.
- **Components:** Every pattern will have several components that are used together in order to provide the pattern properties. Usually, they are the physical elements used by a pattern (e.g. a CCTV camera or a sensor, for a movement detection pattern).
- **Parameters:** One important aspect about components is that sometimes they can be generic. In this case, the component will appear as a formal parameter in the pattern. We define the terms formal parameter and actual parameter analogously to the typical definitions used in programming languages:
The formal parameters are the names that are declared in the parameter list of the pattern. The actual parameters are the elements used when the pattern is instantiated (we could consider the pattern instantiation as a call to a subprogram).

- **Pre-Conditions:** A pattern is not necessarily a universal solution. This means that for the pattern to be successfully used to provide the declared properties, some pre-conditions must be met.
- **Static Tests Performed:** Security engineers will be responsible for the static testing of the pattern. This section will describe all relevant information regarding the static tests performed. We foresee that it might be necessary to develop mechanisms for the description of the tests, in a similar way as the description of the properties. This section will be useful for the monitoring mechanism, as some monitoring rules can be derived from it.
- **System Configuration:** In addition to the instantiation and integration of the pattern in the system it will be sometimes necessary to perform some actions before the pattern can be used in the system. Likewise, when the pattern is to be removed, some actions may also be necessary. In summary, the system configuration section of the description will describe the initialization and closing up processes, along with any other relevant system-specific information.
- **Monitoring:** This section describes all information necessary for the monitoring of the pattern. In particular, it must include which monitor to use, and the configuration of such monitor (events to monitor, rules, reactions, etc.)

### 3.1.2. Negotiation and Monitoring Interface

Every instance of SERENITY framework as system will provide interfaces in order to allow interaction with other systems or other SERENITY instances; the following model presents this fact (figure 1). In summary, the framework will provide two main interfaces: On the one hand, it will provide a *negotiation interface* that will be used in order to establish the configuration of the interacting frameworks in a way that suits the requirements from all participants.

On the other hand, SERENITY frameworks will offer a *monitoring interface*. External elements interacting with an instance of the framework will be able to monitor that the behaviour of the framework is correct. The key to success is to capture security expertise in such a way that it can be supported by automated means. With this interface, we provide support for the dynamic supervision and adaptation of the security of systems to the transformations in ever-changing ecosystems.

Of course, the SERENITY framework will also feature the counterparts of these offered interfaces (required interfaces). It is foreseen that some external elements other than SERENITY frameworks will be able to interact with them by way of these interfaces.

![Figure 1: Deployment model, more than one framework instance.](image)

### 3.1.3. Context Retrieval Interface & User Interface

Figure 2 presents an overview of the SERENITY framework. In the figure, the main interaction involves the Security Experts providing sound solutions and keeping track of the evolution of these solutions; the *Context Retrieval Interface* which captures critical information considered to be relevant as a change of the context can sometime alter the applicability of the current solutions; and the *end-user* (in some cases also with the system administrator role) who defines the preferences and the S&D requirements that the system have to meet.

![Figure 2. Global SERENITY framework](image)

The SERENITY framework is instantiated every time it has to be applied to a concrete scenario. The *end users* specify the security and dependability requirements for the systems they are designing by means of the User Interface. For that process, they use some specification tool (e.g. a UML modelling tool, the user menu from a PDA graphic interface, etc) fully integrated with the framework. The system then gets the information of the current context (using the context retrieval interface), looks for the most suitable solution fulfilling each one of the requirements, and starts the negotiation with the framework to choose the most appropriate pattern/s or combination of them to apply in the problem. Then, the end user does the final selection of a solution and the...
framework checks the dependencies and verifies the correctness of the selected model. After that, the run-time monitoring mechanism starts monitoring the workflow of the system. This will allow to react on time after any vulnerability on the requirements or to apply an improved solution (e.g. a new security patch), always taking into account that the patterns are in constant evolution and the system may decide that a new pattern has substitute the current one. More than that, the monitoring process appears as indispensable in case of change of the context of the system (e.g. a connection to a trusted network using your PDA, and then changing to an untrusted environment such as internet), when SERENITY has to adapt on-time the current solution in order to face the new situation.

3.1.4. Logical Components of the Architecture

In order to describe this architecture, we need to combine different descriptions representing different aspects or elements of the ecosystem. In particular, we envision the following descriptions:

- **Description of Patterns.** For the moment we assume that this is common for S&D Patterns and Integration Schemes. The description of patterns includes two model elements: the PatternCertificate and the S&DSolutionSemantics.

- **Description of Properties.** Related to the description of the patterns is the description of the properties provided by them. This description must fulfill two basic objectives: (i) univocally identifying the properties, and (ii) enabling interoperation of the different properties. It corresponds to the S&DPropertySemantics in the conceptual model.

- **Description of the User S&D Configuration.** This description contains the end-user S&D requirements (properties that must be enforced) as well as preferences and user amendments to pattern definitions. It corresponds to the S&DConfiguration in the model.

The logical model presented in figure 3 shows the conceptual elements that are used in SERENITY as well as its relations. On the right hand side of the model, S&D Patterns and Integration Schemes (S&DPatterns) refer to solutions (S&DSolutions) and contain the semantics (S&DSolutionSemantics) that describe such solution. The semantics are described in terms of the semantics (S&DPropertySemantics) of the particular properties (S&DProperty) provided by the solution. All the S&D patterns are added to a library, and the framework can use them to enforce a specific set of S&D properties defined by the user. Security Engineers are in charge of the creation and maintenance of such library. Solutions can be monitored by using certain monitoring mechanisms (MonitoringMechanisms).

S&D Patterns and Integrations Schemes are certified by a special type of digital certificate (PatternCertificate). Therefore, the libraries of patterns (S&DLibrary) are composed of S&D Patterns and Integration Schemes that have a certificate, denominated certified patterns (CertifiedS&DPattern).

Finally, users will define the security and dependability requirements (S&DConfiguration) for their systems, which will contain a set of specific requirements (S&DConfigurationElement). Each specific requirement will specify a set of properties (S&DProperty) that must be enforced for a particular element of the system (SystemElement). With all this elements, the user benefits from a SERENITY framework instance that satisfies all the security requirements.

3.2. Instantiation and Adaptation of the Framework

The SERENITY framework is instantiated every time it is applied to a concrete scenario and device. Moreover, in AmI environments S&D requirements as well as external context elements should change so that some of the applied patterns are no longer applicable. It arises the necessity of defining a run-time protocol to respond to the new applicability conditions.

Figure 4 outlines the basic steps taken during a typical change of context situation. Once the framework is applied and configured, the S&D Manager component asks the Monitoring Internal Interface to start the monitoring process. Once an external condition changes (e.g. the user takes his laptop out of the office and tries to check his emails from his favorite restaurant), the framework gets into action. As the external context changes, the current system requirements are also modified. In this concrete example, a device previously connected to a trusted private network, suddenly tries to connect to the office email server from a non-trusted network (e.g. the wireless network of the restaurant).
triggering the need of having a stronger encryption mechanism.
The change of context fires an event that is captured by the External Monitor, which automatically sends this hot information to the Monitoring Internal Interface. The Serenity Framework uses the information given by the motoring mechanisms to check the validity of the current configuration. It access into the Active Patterns component and detects that the pattern called “Secure connection over trusted network” is not applicable under this new conditions, alerting the S&D Manager. The framework creates a new query containing a set of search criteria in order to search into S&D Library for a new S&D Pattern or Integration Scheme applicable under the new conditions.
Once a new pattern is found, the S&D Manager starts the verification of its applicability. It is not only necessary to verify the application restrictions and requirements but also to establish the integration procedure regarding with the rest of the patterns placed in the Active Patterns component.

The Integration Schemes are basic items in this process. They convey the information that allows instantiating the concrete solution specified in the pattern, and to include it among the rest of the solutions of the Active Patterns. The result is the same Serenity Framework instance, but with a new adapted configuration.

4. An Application Example

After the specification of the SERENITY framework as well as the instantiation and adaptation process, we introduce a real world scenario. This will illustrate concrete Security & Dependability problems and how the framework deals with them.

4.1. The Hospital scenario

In our hospital a doctor is in charge of several patients. The doctor visits each patient under his assignment and when necessary, he prescribes treatment based on his observations. In each visit, doctors retrieve patient health information that they take into account in order to prescribe treatments.

In addition, our hospital has two new technological advances that help both doctors and rest of staff members to improve their work.
The first one is the patient tracking system. Basically, this is a network of sensors that submits patient health data to a near computer. Patients’ beds are provided with sensors able to monitor different aspects of their health (blood pressure, temperature, heart beats, breathing, etc.). All data collected by patient’s sensors is managed by a room computer. Hospital Central Server (HCS) provide access to different rooms computers. Thus, each patient can be monitored using a computer connected to the hospital intranet.
The second advance offers the chance to check patient’s health by the use of a mobile device, such as a PDA. By using this device the doctor is in permanent contact with his patients either if he is in the hospital or if he is outside the building. The mobile devices connect directly to the HCS using the secure intranet when doctors are inside the hospital boundaries, and try to use another method (Internet, GRPS, 3G …) when the doctors change their location.

4.2. Analysis of the scenario

After a quick analysis, the previous scenario shows at least two key questions: (i) the patient’s personal data is sent over the network and (ii) only the authorized doctors
should be able to access these data. Apart from that, taking into account the connection possibilities and the mobility of the system, we can distinguish two cases, graphically expressed in figure 5:

Case 1: The doctor requires information from inside the hospital. Using a mobile device such as a PDA, a smartphone or a laptop, the doctor is able to access the data of an assigned patient. Also, if a sensor detects critical levels in the measures, the patient tracking system can establish a direct connection with the doctor to alarm him. The hospital relies on desktop computers too. With a more flexible interface than a PDA, they could be used to perform more complex actions than a PDA or a smartphone. In this case, the doctor can not only access the data but modify or prescribe the treatment. The access control system has to take into account the type of device used to access the data, as well as the level of authorization of the doctor. More than that, it is important to assure that the person using the device is correctly authenticated against it, usually via a biometric system installed in the mobile device.

Case 2: The doctor requires the information from outside the hospital. In this case, the system has to allow the doctor to establish a secure connection with the internal system of the hospital through an insecure network. The system does not know a priori the origin of the user connection. The user can connect from a train station, using a public wireless network; from the train, using 3G via his mobile phone and so on. This situation arises two problems: the first is to achieve transparency for the connections, as the user has to be unaware of the connection changes; the second to provide security properties such as privacy, authenticity or authorization using an untrusted network. It is important to note that is much more difficult to maintain the privacy of the patient data once his information is going out of the scope of the hospital trusted environment. More than that, it is additionally difficult to make it transparent as the user is checking the patient data in his PDA while leaving the building from the reception of the hospital, in order to take a coffee in a place crossing the road. In a perfect AmI system, the device has to be able to switch between two operating modes without user notification, or with the slightest interaction from the user side.

4.3. Applying SERENITY to the scenario

The most suitable way to make all the previous system work is to provide every device with a mechanism to react on time to the constant and unpredicted changes of context. The SERENITY framework will be running in every device that composes the hospital system. Thus, there is one framework instance in each mobile device, (such as PDA, laptops or smart phones) and also in the computing elements permanently placed inside the hospital (such as the Hospital Central Server –HCS or the patient tracking system, composed by a set of sensors). It is important to highlight that every framework is fully operable by itself and, as seen in figure 3, they provide two interfaces to:

(i) Negotiate with other boundary frameworks: after the PDA gets out of the scope of the internal network, it will negotiate with the HCS the most secure encryption algorithm according to HCS security requirements or preferences.

(ii) Monitor the critical elements of the framework: the change of connection from a trusted network to a 3G mobile connection fires a monitor trigger alerting about the no-longer validity of the current encryption algorithm.

Case 1 shows how a typical scenario should work once the requirements have been studied, the system modeled, and the application successfully installed. The system takes advantage of SERENITY once the number of end users starts growing without being predicted; once a new and smart device is now used by the surgeons to communicate with the patients and between each other; and generally speaking, once a new unpredicted requirement not present at the early stages of the system development, appears.

In case 2, the scenario introduces more complex features. The monitor interface present in the SERENITY framework that runs in the mobile device recognizes the untrusted environment (e.g. a 3G internet connection) and
fires an event, alerting the system. As a result, the S&D manager collects all the information regarding with the new context of applicability and detects a violation in the protocol previously negotiated with the HSC framework. That protocol was valid only under the presumption of a local network, so the HSC will not allow the interchange of the patient data under the new circumstances. Quickly, the S&D Manager starts a negotiation with the HSC, and they establish that the connection will be possible if the mobile device is able to launch a VPN with the HSC. Again, the S&D Manager manages the situation and starts looking for a security pattern to achieve the new connection requirements. If a correct pattern is found, it is installed and placed with the rest of active patterns of the framework. The validity of the solution will be verified, and the monitoring mechanism for the new pattern will be activated.

Probably, as result of this adaptation the doctor will not be allowed to alter the treatment but he will still be able to check the patients’ condition.

Finally, it is also necessary to remark the configurability of the framework. From the point of view of the patient, it should be desirable to achieve the desired grade of confidentiality. In case of being a public person, the threat for the medical data to be stolen is higher than the usual case. Therefore, it should be desirable for the user to establish which kind of data his doctor can freely share and which other data should be restricted. In any case, we are talking about a different configuration for a pattern common to every patient of the hospital. Consequently, users can adjust some SERENITY framework settings by using some parameters of the applied pattern.

5. Conclusions and Future Work

The work presented in this paper describes our approach to face the problem of security and dependability in AmI environments. The proposed framework, based on a solid conceptual design, embeds (i) the best solutions proposed by security experts by means of S&D patterns; (ii) a continuous interaction with the surrounding AmI environment; (iii) and the mechanisms to offer a meeting point between the requirements imposed by the environment and the solutions offered by the S&D library.

We also present an application example, where new heterogeneity and ubiquitous needs enforces the use of innovative security solutions. The monitoring mechanisms along with the instantiation and adaptation capabilities of the SERENITY framework allow to react at run-time and to adapt security solutions and configurations according to the new contexts in which the elements of the hospital (from doctors to PDA, through sensors) have to work as they are supposed to.

We are currently in the process of implementing these concepts in the ongoing SERENITY project. Our interest focuses on the realization of the concept of S&D Pattern, the semantic definition of security and dependability properties in order to guarantee the interoperability of S&D Patterns from different sources, and the provision of a trust infrastructure for the patterns.

6. References


