Computers **2013**, *2*, 34-66; doi:10.3390/computers2010034

OPEN ACCESS

computers

ISSN 2073-431X

www.mdpi.com/journal/computers

Article

Architecture and Knowledge-Driven Self-Adaptive Security in Smart Space

Antti Evesti [1,*], Jani Suomalainen [2] and Eila Ovaska [1]

1 VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland;
E-Mail: eila.ovaska@vtt.fi (E.O.)

2 VTT Technical Research Centre of Finland, Vuorimiehentie 3, 02044 Espoo, Finland;
E-Mail: jani.suomalainen@vtt.fi (J.S.)

* Author to whom correspondence should be addressed; E-Mail: antti.evesti@vtt.fi;
Tel.: +358-20-722-2101; Fax: +358-20-722-2320.

Received: 26 November 2012; in revised form: 24 February 2013 / Accepted: 4 March 2013 / Published: 18 March 2013

Abstract: Dynamic and heterogeneous smart spaces cause challenges for security because it is impossible to anticipate all the possible changes at design-time. Self-adaptive security is an applicable solution for this challenge. This paper presents an architectural approach for security adaptation in smart spaces. The approach combines an adaptation loop, Information Security Measuring Ontology (ISMO) and a smart space security-control model. The adaptation loop includes phases to monitor, analyze, plan and execute changes in the smart space. The ISMO offers input knowledge for the adaptation loop and the security-control model enforces dynamic access control policies. The approach is novel because it defines the whole adaptation loop and knowledge required in each phase of the adaptation. The contributions are validated as a part of the smart space pilot implementation. The approach offers reusable and extensible means to achieve adaptive security in smart spaces and up-to-date access control for devices that appear in the space. Hence, the approach supports the work of smart space application developers.

Keywords: architecture; authentication; authorization; ontology; self-adaptation

---

1. Introduction

The smart space trend has initiated several research projects and publications, from technologies to applications. Smart spaces—such as smart homes, smart buildings and smart cities—offer available information and devices for end-users' purposes without a pre-defined configuration or application behavior. Smart spaces are dynamic, and moreover, utilized technologies are heterogeneous. Consequently, it is not possible to envision all the possible situations where the smart space application will be utilized, which creates a challenge for software designers. Self-adaptation is a possibility to respond to this challenge by postponing decision making from design-time to runtime. Self-adaptation is a software's capability to configure and tune its functionality at runtime, in order to tackle changing situations. Challenges related to smart space security are complex—due to the dynamicity and heterogeneity of smart spaces. Firstly, the required security objectives vary between situations, e.g., sometimes integrity is the essential security objective but in other situations the user's privacy is the first priority. Secondly, the security needs of a smart space application vary between situations. For instance, an application utilizing entertainment or critical control information has variable requirements for security effectiveness, *i.e.*, for the security level. Thirdly, smart spaces change continuously, as new devices appear and leave. These devices must be able to use the smart space's available security mechanisms. However, the same security mechanisms are not applicable in all smart spaces. For example, one smart space may support only one particular authentication mechanism while another provides three different mechanisms.

These challenges demand an adaptive security solution that is able to change and modify the used security mechanisms autonomously at runtime. The importance of self-adaptation in smart spaces is also recognized in [1]. A reference model called MAPE-K (Monitor, Analyze, Plan and Execute) has been introduced as a solution for self-adaptation. This model is generic, *i.e.*, it can be applied for various quality and functionality adaptations, and thus it was selected as the reference model for security adaptation in this work. In the MAPE-K model, the Monitor collects information that is analyzed to recognize adaptation needs. Thereafter, the Plan phase creates an adaptation plan for execution. These four phases constitute an adaptation loop supported by knowledge. Currently, several security-adaptation approaches exist [2,3]. The first survey reveals that the existing security-adaptation approaches concentrate on specific security objectives. The second survey shows that the existing approaches have a lack in the adaptation loop coverage, *i.e.*, the approaches do not define the whole MAPE loop. Moreover, Yuan *et al.* note that the abstract architecture for security adaptation is not presented in the existing approaches [3]. These architecture-level problems complicate the reusability and extensibility of the existing security-adaptation approaches. The MAPE-K model separates knowledge from the adaptation loop. However, existing security-adaptation approaches do not support this separation but utilize hard-coded adaptation decisions, which is not sufficient in dynamic smart spaces. Moreover, smart spaces require flexible access control, which is able to handle the situations when new devices and information constantly appear in the smart space. However, the existing approaches are not able to offer this flexibility for dynamically changing access control needs.

This paper concentrates on architecture, knowledge and access control problems. Hence, the paper makes the following contributions: (1) Architecture for the security-adaptation loop is defined and mapped to the MAPE model. Hence, the architecture covers all the adaptation phases in a clearly

defined form. The adaptation loop selects from the security mechanisms and configures the parameters of those mechanisms at runtime. (2) The knowledge is mapped to the security-adaptation architecture, in order to encompass the knowledge part of the MAPE-K reference model. In our solution, knowledge is made accessible to the security adaptation by means of ontologies. Compared with the existing approaches, our solution is novel because it makes it possible to minimize the amount of hard-coded knowledge and supports knowledge addition and modifications. (3) A runtime security-control model for dynamic authorization and access control is defined. The security-control model defines how shared information is effectively secured and controlled in smart spaces.

The novelty of contributions comes from the integration, which builds up the consistent security-adaptation architecture. The architecture covers all adaptation phases from monitoring to execution with dynamic access control. In contrast to the existing approaches, knowledge is separated from the architecture, and mappings from the architecture to knowledge are presented. Thus, the presented security adaptation does not require hard-coded rules for the analysis and planning phases because ontologies form the knowledge for the adaptation. Finally, the whole approach is validated as part of the wider smart space pilot implementation.

Background information and related work are described in Section 2. The whole adaptation approach is presented in Section 3. Section 4 describes the implementation of the approach by means of a use case. A discussion of the approach and future research are summarized in Section 5. Finally, the Conclusions Section closes the paper.

2. Background and Related Work

2.1. Smart Spaces

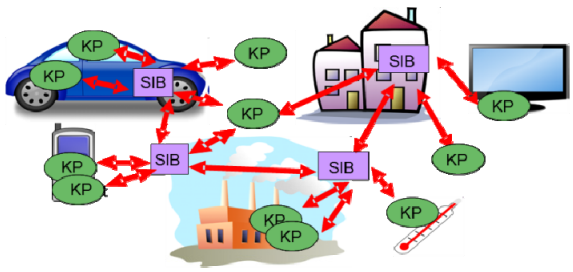
Smart spaces are physical spaces where devices cooperate and share information to intelligently provide services for the users. Cook *et al.* define a smart environment as one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment [4]. The terms smart space and smart environment are widely used interchangeably—this article uses the term smart space.

Cooperation and information sharing requires that devices and Smart Space Applications (SSAs) are able to interoperate. This interoperation is able to occur at different levels, which are called interoperability levels, defined in [5,6]. The presented interoperability levels from bottom to top are Connection, Communication, Semantic, Dynamic, Behavioral and Conceptual interoperability. The Connection interoperability level focuses on network connectivity, whereas the Communication level focuses on data syntax. However, these two lower-most interoperability levels are out of the scope of this paper. The Semantic interoperability level concentrates on understanding data from the communication level. Next, the Dynamic level focuses on context changes and the Behavioral level matches actions together. Finally, the Conceptual interoperability level focuses on abstracting; representing easy-to-use knowledge to the other interoperability levels, and making deductions based on data, context and actions. Thus, conceptual interoperability creates meaning from the information, context and behavior in the smart spaces. Hence, this is the level where the "smartness" is built for the smart space.

The SSA consists of a set of software agents that communicate and share information with each other. Therefore, the deployment of the SSA can be distributed to several smart space devices, *i.e.*, agents of the SSA are executed in different devices instead of one centralized device. These agents and the composed SSA act in dynamically changing smart spaces, which may offer a huge amount of information. Context-awareness is a means to handle this information flow in order to provide reasonable information and services for the user. Similarly, security in smart spaces requires context-awareness in order to provide reasonable security for different situations and actions. From the interoperability-level viewpoint, SSAs and context-awareness occur at the Dynamic and Behavioral levels.

Establishing a smart space requires an appropriate infrastructure—in this paper, the Smart-M3 concept [7] will be utilized. In the Smart-M3, the Semantic Information Broker (SIB) forms a backbone for the smart space. The SIB takes care of information sharing between agents—called Knowledge Processors (KPs). Agents are able to make queries and subscriptions, and insert semantic information in the SIB. Consequently, various devices are able to interoperate, *i.e.*, share semantic information, by means of the SIB and agents inside devices, as illustrated in Figure 1. The SIB utilizes semantic web technologies—especially Resource Description Framework (RDF) [8] and SPARQL query language. From the interoperability-level point of view, the SIB embodies the Semantic Interoperability level. In the Sofia project [9], three different Smart-M3 concept implementations were made for different usages. This paper utilizes the implementation called RIBS [10], which contains mechanisms to secure communication and is able to work in resource-restricted devices—such as a WLAN access point.

Figure 1. Smart spaces formed by Semantic Information Brokers (SIBs) and Knowledge Processors (KPs).



Security challenges caused by the dynamicity and heterogeneity of smart spaces are mentioned above. Moreover, traditional security challenges, e.g., key exchange and resource restrictions, are present in the smart spaces. Similarly, openness and free utilization, which are characteristics of smart spaces, affect security. Nevertheless, this paper focuses on security challenges due to dynamicity and heterogeneity by presenting a security-adaptation approach with a dynamic access control.

2.2. Security Adaptation

Kephart *et al.* define autonomic computing as computing systems that can manage themselves by using high-level objectives given by administrators [11]. The autonomic element contains the MAPE-K control loop composed of the Monitor, Analyze, Plan and Execute phases, supported by

Knowledge. A similar structure is also applied in [12,13] as a reference model for autonomic computing. However, the names of the phases vary. For instance, Dobson *et al.* use the terms Collect, Analyze, Decide and Act [12]. In contrast, Psaier *et al.* define a control loop that joins the Analyze and Plan phases into one Diagnosing phase [14]. The loop concentrates on self-healing, which is a form of autonomic computing. However, in this paper the MAPE-K loop will be utilized as a reference architectural model. The terms autonomic computing, self-management and self-adaptive are utilized interchangeably for instance in [13,15]. This article utilizes the term self-adaptive and its short-form adaptive to refer to software's ability to adapt itself at runtime. Adaptability has been defined as the ability of software to adapt its functionality according to the environment and user [16]. From the security viewpoint, functionality means security mechanisms intended to support the required security objectives.

Elkhodary *et al.* survey four approaches to adaptive security in [2]—namely Extensive Security Infrastructure [17], Strada Security API [18], Willow Architecture [19] and the Adaptive Trust Negotiation Framework (ATNAC) [20]. Moreover, the recent survey from Yuan and Malek [3] compares over 30 self-protection approaches. The extensible Security Adaptation Approach (ESAF) distinguishes security mechanisms from the application to the middleware layer [21]. Hence, the application communicates the required security to the middleware without any knowledge of the used security mechanism. In contrast, Context-sensitive adaptive authentication utilizes context information to replace static authentication mechanisms [22]. Hence, in situations where a lower authentication level is sufficient it is possible to utilize other attributes, e.g., location-based authentication, instead of passwords. The GEMOM (Genetic Messaging-Oriented Secure Middleware)—covered also in [3]—offers self-healing and adaptation features to ensure optimal security strength and uninterrupted operation in changing environments and threats [23]. The GEMOM applies the Monitoring part of the MAPE-K model by means of security measuring [24].

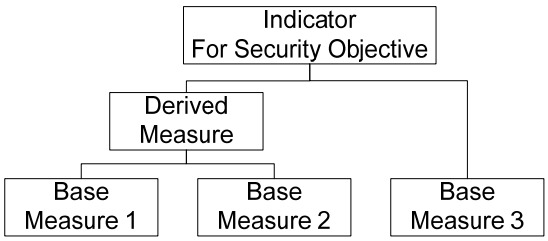
In the security adaptation, the Monitoring phase utilizes security measuring or observes events that affect security. Security measures can be produced by means of the decomposition approach, where security objectives are divided into smaller parts until the measurable components are found [25]. Garcia *et al.* presented a similar technique for generic software measurement in [26] by categorizing measures as Base Measures, Derived Measures and Indicators. Base Measures represent raw measures, which are further composed into Derived Measures. Indicators are on the highest abstraction level—and are able to compose Base Measures, Derived Measures and other Indicators. Figure 2 illustrates the structure of these measures. The indicator for the particular security objective is on the highest level. The Indicator is derived from the Derived measures, Base Measures and other indicators. Structuring measures hierarchically ensures that measures can be reused and extended. Hence, security measures will be applied to Monitoring in this article. The Monitoring phase uses base measures. The results of the base measures are composed in the Analysis phase to reveal the current security level.

The Analyze phase utilizes monitoring results. However, it is not enough to know the current security level but the required security level also has to be known. The required security level can be defined at design-time, or alternatively, the Analyze phase can reason the required level at runtime. Deciding on an appropriate security requirement set is a challenging task. Defining security requirements for design-time purposes is extensively covered in [27]. The presented security

requirements engineering framework takes into account assets, threats, business goals, and system context. All these aspects can be modeled in order to achieve the most extensive security adaptation approach. For example, Salehie *et al.* [28] concentrate on the variability of assets and how this affects security, and how adaptive security is able to deal with these challenges. This is an important viewpoint and starts from assets, which initially set the requirements for all security. It is said that an asset is an entity that someone places value upon [29], and thus, it needs protection. However, we do not model requirements and assets in this granularity. On the contrary, we will utilize context information to recognize the importance and role of handled data, *i.e.*, asset value, which in turn leads to required securities.

Many uncertainties relate to security, and thus, getting accurate numbers to describe security for adaptation purposes is challenging. Sahinoglu [30] utilizes variance values to cover uncertainties in risk analysis. Similarly, in security adaptation, results from monitoring and analysis contain variance in some range. Uncertainties affect the achieved security and the recognized adaptation need but uncertainties do not affect security adaptation architecture or the utilization of knowledge itself. Thus, these uncertainties are out of the scope of this article.

Figure 2. The Structure of Measures.



2.3. Adaptation Knowledge from Ontologies

The MAPE-K model does not define how the knowledge has to be offered. However, in order to follow the separation of the concerns principle, we introduce knowledge as a separately identifiable architectural element by utilizing ontology orientation to represent a self-sufficient model of security concepts. Ontology can be defined as a shared knowledge standard or knowledge model, defining primitive concepts, relations, rules and their instances, which comprise topic knowledge. It can be used for capturing, structuring and enlarging explicit and tacit topic knowledge across people, organizations and computer and software systems [31]. Several security ontologies have been listed in [32]. In addition, our earlier work [33] compared security ontologies from the runtime applicability and measuring viewpoints. Ontologies, designed for runtime usage, often concentrate on the service discovery and matchmaking, e.g., ontologies in [34,35]. However, these ontologies do not cover security measuring. In contrast, Savolainen *et al.* [36] present a security taxonomy for design time usage, which also contains a high-level security measuring part. At the moment, the most extensive security ontology is proposed by Herzog *et al.* [37], known as Ontology of Information Security (OIS). The OIS contains over 250 concepts, which describe security threats, countermeasures, assets and security goals, *etc*. In this paper, security goals and countermeasures are called security objectives and mechanisms, respectively. The OIS lists the following security objectives: confidentiality, integrity, availability, authentication, accountability, anonymity, authenticity, authorization, correctness,

identification, non-repudiation, policy compliance, secrecy and trust. Nevertheless, the OIS does not contain a security-measuring part. In contrast, Garcia *et al.* presented the measurement terminology in an ontology form called Software Measurement Ontology (SMO) in [26]. Consequently, we have combined the Information Security Measuring Ontology (ISMO) from OIS and SMO in [38]. The ISMO makes it possible to present security measures via a common vocabulary and map defined measures to security concepts, e.g., security objectives and mechanisms. In the ISMO, security measures are defined in detail—containing descriptions on how the particular measuring has to be performed and how the base measures can be further combined into indicators. Hence, the ISMO offers knowledge for design-time and runtime purposes, e.g., what kind of measuring probe to implement at design time and how to utilize measuring results at runtime. This paper utilizes knowledge from the ISMO. Furthermore, context knowledge is vital for security adaptation, in order to describe an environment and user actions. For this purpose, we utilize the Context Ontology for Smart Spaces (CO4SS) [39] in this paper. In [40] we defined the taxonomy of context information for security. The taxonomy maps security-related context information to physical, digital and situation context levels. The physical context describes an infrastructure where the SSA is running. The digital context presents the role of the smart space, e.g., public space. Finally, the situation context describes the user's role and activity within the smart space. Moreover, the role of the exchanged/stored data is described in the situation context.

Our earlier work in [41] presented ontology-based security adaptation. In that work, risk-based security measures were stored in the ontology to support security monitoring. Moreover, the ontology contained knowledge about how much each security mechanism decreases the particular security risk, which supports the Planning phase. In [40] we presented a micro-architecture for security adaptation. However, in that architecture the ontology usage was tightly coupled inside the architecture. In this article, the architecture is developed towards the MAPE-K reference model and the ontologies will be separated out to their own interoperability level, *i.e.*, to the Conceptual level.

2.4. Access Control over Semantic Information

To control access to shared semantic information, fine-grained authorization models have been introduced for RDF. These approaches include approaches where access control is implemented as an additional layer on top of the repository, as in [42], and approaches where access-control information has also been integrated into repositories. In the triple-level access control [43], RDF resources are protected with access-restriction properties. Essentially, these properties are links to access policy graphs that specify the owner of the RDF resource as well as those predicates to which this protection applies.

Some researchers have proposed models where RDF-level access control decisions are implicitly derived from existing higher-level policies and context information. The policy-based access control model [44] uses metadata to define permit or prohibit conditions. Jain and Farkas [45] introduce an access-control model where RDF class hierarchy is utilized to manage and derive access control policies. Flourish *et al.* [46] propose a high-level policy-specification language for annotation RDF triples with access-control information. Moreover, approaches for access-control reasoning, based on

concepts and their relations represented by ontologies, have been introduced by Kim *et al.* [47] and Cho *et al.* [48].

However, none of these reasoning solutions are directly applicable for smart spaces. In smart spaces, access control is enforced by information-brokering devices, which are not aware of application-specific policies. Also, semantic reasoning for real-time security control is a challenging task as the reasoning problems are, at the worst case, only solvable in exponential time with respect to the input size [49,50]. Consequently, to enable real-time security enforcement, efficient and scalable solutions are needed. These solutions should enable smart spaces to support different reasoning applications, which may be based on expressive and complex security ontologies.

Our earlier work defined solutions [10,51] for securing communication between smart space devices and controlling information sharing. In [51], we proposed an RDF node-level access-control model for a semantic information broker. The model is simpler than other RDF access-control models as each security policy can be expressed using a single RDF triplet and, in an optimized implementation, be presented with a single bit. In this article, the model is formally defined and generalized, and its granularity is enlarged to protect semantic relationships in addition to semantic information.

2.5. From Quality Variability to Quality Adaptation

In our approach, building a capability for quality adaptation begins at design-time. Consequently, our earlier work combines quality-, model-, and knowledge-driven software development. Our previous work presented the quality-variability model [52]. The variability model defines binding times for variations, *i.e.*, design, assembly, start-up and runtime, which defines the latest time point when quality can be changed. This work concentrates on the situation where security variation occurs at runtime—called security adaptation. Quality variability is closely related to architectures, and thus, the approach for the knowledge-based quality-driven architecture design and evaluation was presented in [53]. The approach contains three steps: (1) Modeling the quality requirement. (2) Modeling the software architecture and transforming the requirements to the models. (3) Quality evaluation. Steps one and two are divided into the knowledge and software engineering processes, whereas step three is divided into the quantitative and qualitative evaluation processes. In that study, quality ontologies for reliability and security were utilized as a knowledge base. Now, we will also bring ontology-based knowledge from design-time for automatic runtime usage.

Lastly, the design steps to produce an application with security adaptation features were presented [54]. The following steps were recognized: (1) Required security objectives—defines all security objectives for the application. (2) Adaptive security objectives—selects objectives, which will be adapted at runtime. (3) Mechanism variants for the selected objectives—selecting security mechanisms and their parameters, which can be adapted at runtime. (4) Measurements for triggering adaptation—selecting security measurements to monitor the adaptation needs of the selected security objectives. (5) Architecture design—designing the selected security mechanisms and measurements into the application architecture in a way that supports runtime adaptation. The contribution of this paper relates to step five, *i.e.*, presenting security-adaptation architecture and mapping it to the knowledge retrieved from the ontologies.

3. The Concept for Adaptive Security

The security-adaptation approach is presented in this section. Firstly, we give an overview of the approach, and thereafter, each part of the approach is presented in its own sub-section. The approach is not bound to any particular security objective or security mechanism. Nevertheless, the approach contains all the necessary components required to build adaptive security for smart spaces. The presented concept is instantiated by means of a case study in Section 4. The case study illustrates the approach from the authentication and authorization viewpoints.

The adaptation approach combines solutions from different interoperability levels. Figure 3 illustrates the proposed solutions—mapped to the interoperability levels and design-time and runtime phases. Sub-Sections 3.1—3.3 concentrate on these levels one-by-one.

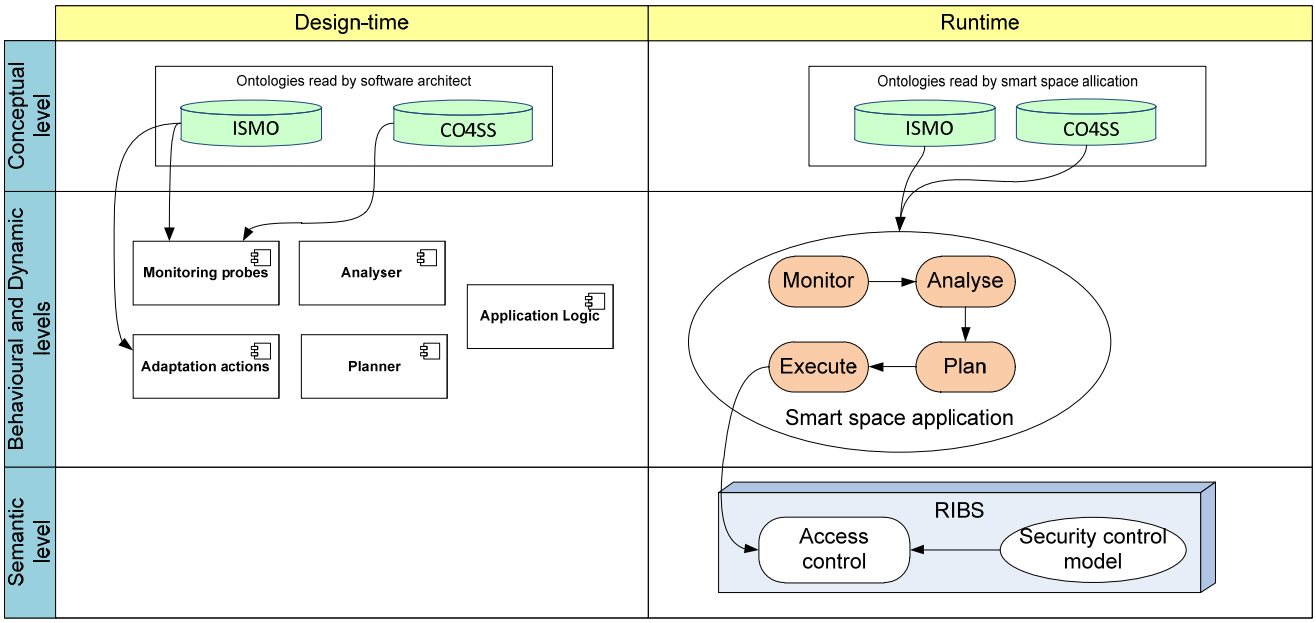
Knowledge, required in security adaptation, is set on the Conceptual interoperability level. The knowledge is described with ontologies—namely the Information Security Measuring Ontology (ISMO) and the Context Ontology for Smart Spaces (CO4SS). Hence, security- and context-related knowledge is offered to other interoperability levels from the Conceptual level at runtime. Ontology-based knowledge ensures that smart space applications (SSA) are able to understand security and context situations in a uniform way. Moreover, the architect utilizes knowledge from ontologies at design-time to implement the appropriate monitoring probes and adaptation actions for the SSA, *c.f.*, connections from ontologies to Monitoring probes and Adaptation actions in Figure 3. In other words, these components are specific for the designed SSA. In contrast, the Analyzer and Planner components search required knowledge at runtime.

The security-adaptation loop—based on the MAPE reference model—and the SSA are located on the Behavioral and Dynamic interoperability levels. The architect designs and implements the required software components at design-time. These components are as follows: Pure application logic, monitoring probes, security mechanisms with adaptation actions, analyzer and planner. The adaptation logic is located in the Planner component. However, the architecture does not dictate the utilized adaptation logic, *i.e.*, the internal functionality of the Planner component. The plan phase is described in Subsection 3.2.3. At runtime, the SSA utilizes monitoring probes to observe security and context changes. The Analyzer component analyses the consequences of the changes based on knowledge retrieved from the ontologies at runtime. Sequentially, the Plan component creates an adaptation plan for responding to changes. Finally, the adaptation plan is enforced by the executors. In the security adaptation these executors are the selected security mechanisms. For example, "a new mobile device has arrived in the smart space" is a change observed by means of monitoring. The SSA analyzes the security consequences of a new device and decides how to adapt to this change. Adaptation actions can, e.g., affect information sharing on the semantic level or reset the communication parameters.

On the Semantic interoperability level, the RIBS constitutes the smart space infrastructure. The RIBS takes care of information sharing between smart space devices and SSAs. Hence, various devices are able to interoperate via the RIBS by sharing semantic information, which is presented by means of RDF. The Semantic level has to contain security solutions, which protect and control the sharing of semantic information. Therefore, we propose a semantic level security-control model to enable efficient access control. The security-control model enables SSAs to prepare security policies so that the RIBS does not have to support complex ontologies or perform runtime security reasoning.

Applications requesting and providing information must unambiguously define the semantics of shared information. However, the presented dynamic access control ensures that variations are handled at runtime, without design-time rules.

Figure 3. Interoperability levels and proposed solutions.



At this point, it is necessary to emphasize the difference between semantic and conceptual interoperability levels. However the semantic technologies, *i.e.*, RDF and OWL, are applied on both levels the difference comes from the abstraction level of the knowledge. The Semantic level contains separated pieces of information, e.g. temperature is minus five or a password length is seven characters. In contrast, the Conceptual level makes it possible to deduct the causes of the semantic information, e.g., water will freeze or the authentication level is low.

3.1. Security Adaptation Concepts from Ontologies

The right knowledge is an essential part of the security adaptation. In our solution, knowledge will be offered from the Conceptual interoperability level by means of ontologies. Ontologies make it possible to update and extend the existing knowledge. Moreover, ontologies support reusability and offer knowledge in a machine-readable form. Knowledge requirements for the security adaptation are threefold, *c.f.*, Figure 4. Firstly, security knowledge is needed to describe security mechanisms, security objectives, threats and their relationships. Secondly, measuring concepts are needed to monitor the achieved security, *i.e.*, the current and/or past security. Thirdly, context knowledge is needed to describe the state of the smart space from situational, digital and physical viewpoints. The situational context is intended to describe the user's role in the smart space and the role of the exchanged/stored data. In contrast, the digital context depicts the role of the smart space. Lastly, the physical context presents the execution platform and smart space infrastructure. The knowledge is arranged in two separate ontologies, *i.e.*, ISMO [38] and CO4SS [39], which together contain over

300 concepts and their connections. The ISMO describes security and measuring knowledge, while CO4SS contains context knowledge.

Figure 4 summarizes the main dependencies of these ontologies, and thus, it contains only the main concepts. The additional concepts and connections are presented in Figures 6–10 in Subsection 3.2. It is notable that the content in Figure 4 is laid in the conceptual level in Figure 3 (the highest level). The context knowledge from the CO4SS sets the required security objectives and levels. For example, dealing with professional information in an office environment has different security requirements than handling the same information in a public environment, which may contain additional threats. The ISMO describes which security mechanism supports the particular security objective and how the security objectives mutually relate. However, the applicable mechanisms depend on the physical context of the smart space, which describes the execution platform and operating system. For instance, the used operating system supports only a particular security mechanism, therefore, in Figure 4 the Physical context is connected to Security mechanisms with an offers connector.

Triggering the security adaptation requires that both security and context concepts are monitored. The monitoring focuses on measurable attributes (attribute), and thus, security and context are connected to the Attribute, *c.f.*, Figure 4. In Figure 4, the connections to Attribute start from the Security concepts and Context concepts frames, which means that all of those concepts can contain measurable attributes. Examples of attributes are a key length from the security side and the number of smart space devices from the context side. Naturally each attribute contains its own measures.

Figure 4. Dependencies of security and context ontologies.

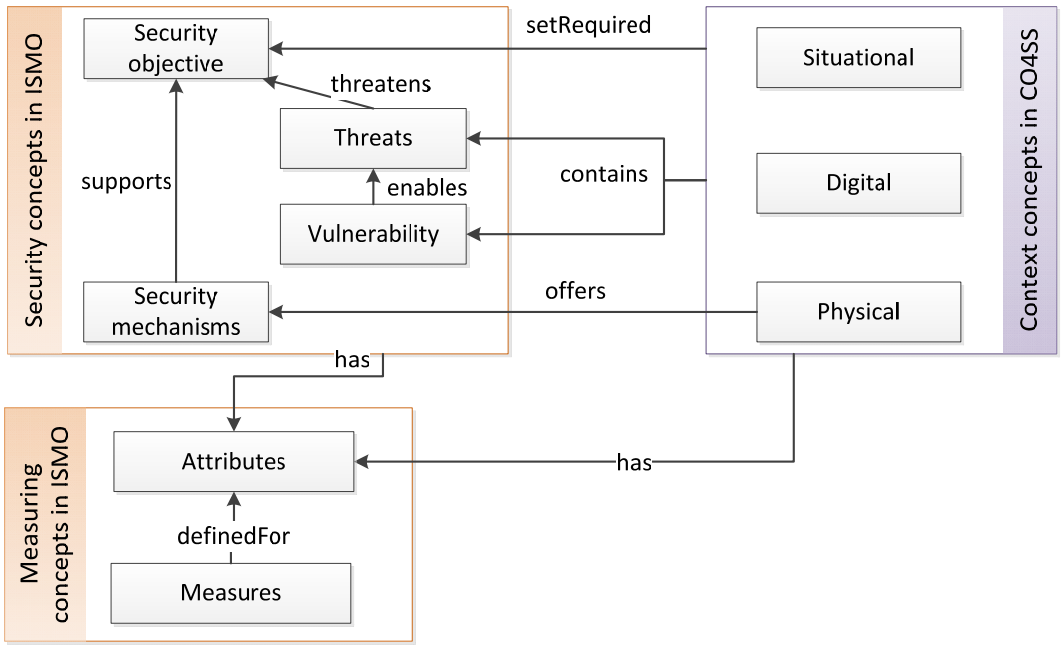


Figure 3 shows that knowledge from ontologies is utilized at design-time and runtime alike. At design-time, the software architect implements a set of monitoring probes, which require knowledge of the measures. Moreover, the architect searches which security mechanisms to implement from ISMO. Additional details of the design-time use of ISMO are presented in [38,53]. At runtime, the SSA automatically utilizes knowledge from the ISMO. The application has to know what measures to

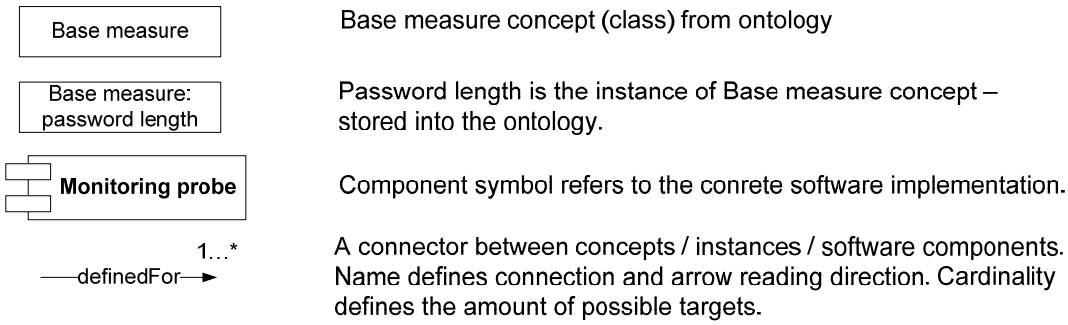
utilize, how to analyze results, and finally, how to create an adaptation plan. These knowledge requirements are described in detail in the following sections.

Smart spaces create a need to update knowledge every now and then. There can be several reasons for updates: new security mechanisms appear, vulnerabilities are found, the threat landscape changes or the execution environment changes. The SSA has to be aware of these changes, which is ensured by up-to-date knowledge. Utilizing ontologies as the knowledge source offers an advantage from a knowledge updating and enhancement view point. ISMO and CO4SS are presented in an OWL format, and thus, updates can be made to ontologies without modifying SSAs, which only retrieve knowledge from ontologies.

3.2. Architecture for Security Adaptation

The security-adaptation approach follows the MAPE model. The Monitor phase collects information by means of monitoring probes. The Analyze phase calculates the security level achieved, reasons the required security level, and calls the Plan phase if the required securities are not achieved. The Plan phase creates a plan to adapt, and finally, the Execute phase enforces the adaptation plan. The following descriptions of these phases show, how they utilize knowledge from the Conceptual interoperability level. Figure 5 presents a legend for figures used in the next sections.

Figure 5. Symbol definitions.



3.2.1. Monitor

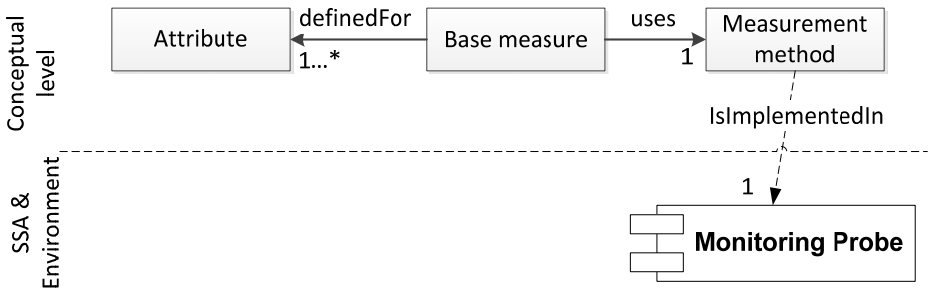
The Monitor phase gathers information from the environment and the SSA itself. The purpose of monitoring is to collect those small information pieces, which are then utilized to reveal an adaptation need. The target of monitoring can be at any level—from a low-level network technology to a high-level description of the user situation. These monitoring targets are called attributes—the encryption key length and the number of unknown devices in the smart space are examples of monitored attributes. Monitoring utilizes security measures, and thus, the measuring related terminology is utilized in the Monitoring phase. The Monitor phase uses Base measures to collect raw data by means of Monitoring probes. It is notable that Figure 4 presents the Measure concept, and the Base measure is one subclass of the Measure.

At design time, the software architect implements Monitoring probes into the SSA and environment based on the Measurement method descriptions from the ISMO, *c.f.*, Figure 6. If the architect creates a new Base measure and Monitoring probe, knowledge about what attribute the probe measures is added

into the ISMO. The Monitoring probe is a concrete code snippet, which is able to observe the particular attribute, e.g., by retrieving a key length from the utilized encryption library. In other words, the Monitoring probe is the implementation of the Measurement method. Each Monitoring probe is intended to observe only one attribute from the environment or SSA. Therefore, implemented probes can be easily reused.

Figure 6 depicts knowledge from the ISMO, *i.e.*, the Conceptual level, and its relation to the SSA and the environment in the Monitoring phase. The SSA and environment contain several attributes and each attribute has its own Monitoring probe implementation. At runtime, the ISMO provides knowledge about useful Base measures. The environment contains several probes but only a certain set is needed in each particular situation. For instance, in a situation where communication integrity is not needed it is useless to utilize integrity-related Base measures. Hence, the knowledge from the ISMO reveals which probes to use. Consequently, it is vital that the ISMO contains the connections depicted in Figure 6 because that information shows what attribute each probe is able to measure.

Figure 6. Knowledge from the Information Security Measuring Ontology (ISMO) for the Monitoring phase.



3.2.2. Analyze

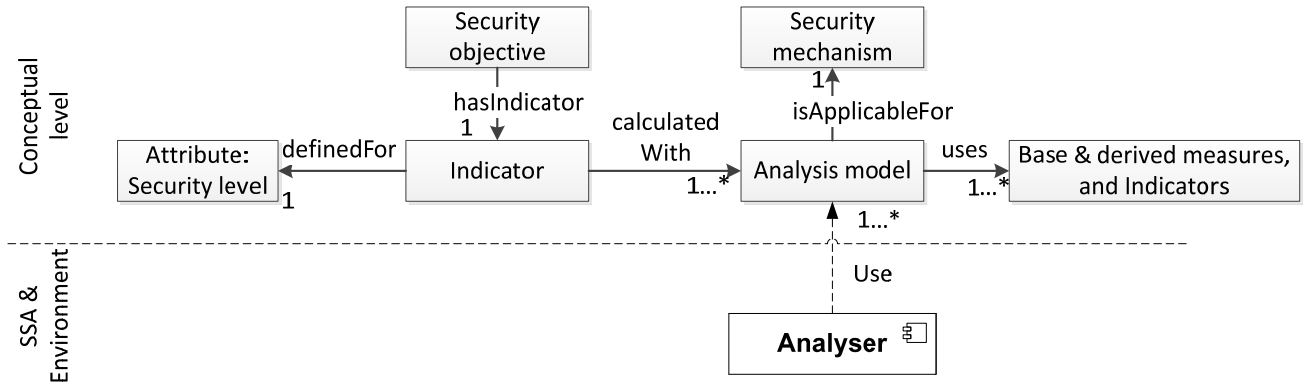
The Analyze phase reveals the current security levels for the required security objectives and decides which levels are sufficient for the current situation. Hence, the Analyze phase combines the Base measure results from the Monitoring phase to Indicators. The Indicator presents the security level of the security objective in that particular smart space situation. In other words, Base measures indicate that some changes have occurred in the smart space, whereas, the Analyze phase reveals the consequences of these changes, *i.e.*, it builds a meaning for the information in that situation.

The ISMO contains Derived measures and Indicators, which combine the Base measure results. Base measures, Derived measures and Indicators are sub-classes of the Measure concept, depicted in Figure 4. The ISMO follows a terminology defined in [26], and thus, Derived measures use the Measurement function and Indicators use Analysis models to perform the combining process. The Measurement function can be a simple mathematical operation, e.g., base_measure_1 + base_measure_2. In contrast, Analysis models contain more complex structures including conditional clauses and Boolean operations. For instance, the result from the Analysis model can be the integrity level of communication.

Figure 7 shows concepts for the Analyze phase from the ISMO. Each indicator has Analysis models, which use Base measures, Derived measures and Indicators. Every Security objective has its

own Indicator—e.g., authentication is a security objective, which has an Indicator called the authentication level. However, the Indicator is able to use several Analysis models, depending on the situation in hand. For example, a different Analysis model has to be used when authentication is based on fingerprints, passwords or a multi-factor authentication mechanism. In the multi-factor authentication case, the Analysis model, which combines analysis models from a single mechanism, is applied. Therefore, each Analysis model has a property that binds it to the security mechanism—from the above example, multi-factor authentication is seen as an individual security mechanism. Similarly as the Base measures, Indicators are also defined for Attributes. At design-time, the architect brings the Analyzer component into the SSA and the Analyzer component searches knowledge from the Analysis models to calculate security-level indicators for the SSA at runtime.

Figure 7. The concepts from the ISMO for the Analyze phase.

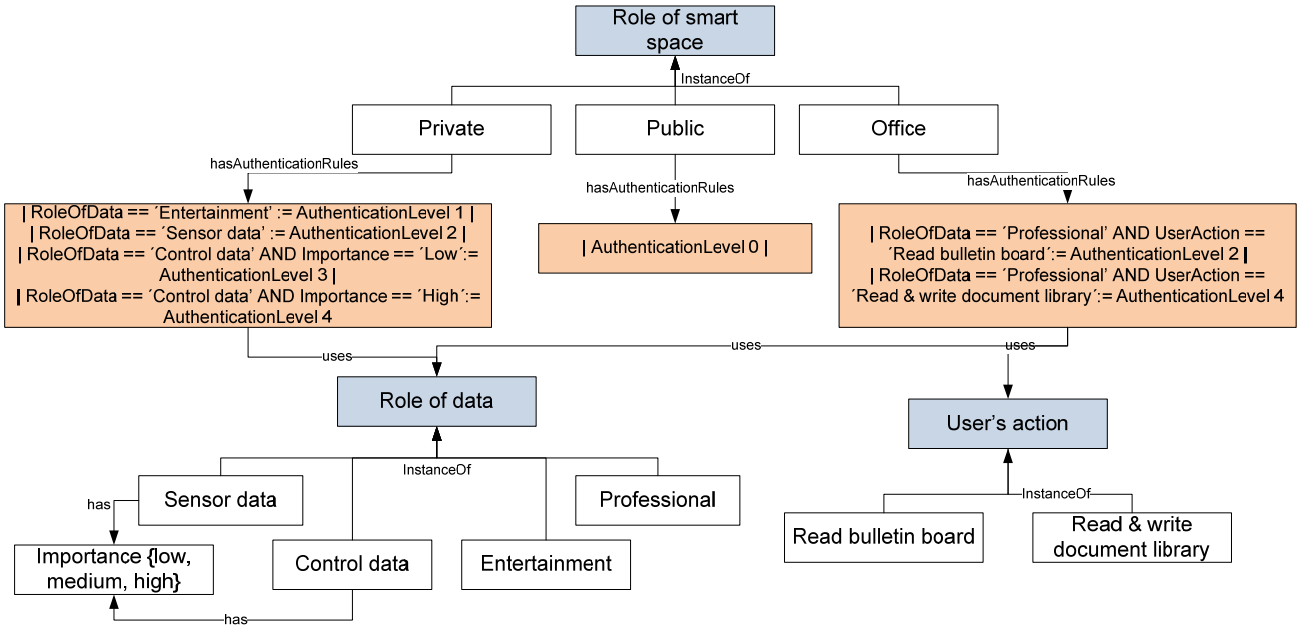


The security level indicators are compared to the required security levels. As depicted in Figure 4, the context information sets the required security objectives and levels. In [40], we defined the context concepts, which affect the required security as follows: (i) The user's role in the smart space. (ii) The actions performed in the smart space. (iii) The role and importance of the data. (iv) The role of the smart space. Furthermore, user preferences are able to affect the required security level.

Figure 8 shows the context information rules that define the required authentication level for different situations. However, the rule sets are an example—not the extensive rule sets to define authentication requirements. In these rules, the role of the smart space, *i.e.*, Private, Public or Office, constitutes the main categorization between the rule sets. In the private smart space, e.g., home smart space, the role of the data and its importance define how strong authentication has to be used. Thus, in a situation where the user consumes entertainment data—like news—authentication level 1 is sufficient. However, in a situation where the high importance level control information is handled, e.g., a home alarm system, authentication level 4 is required. In contrast, in public smart spaces, like freely available smart city services, authentication is not required at all. Finally, in an office smart space the user's action is also taken into account in the rules. Similar context-based rule sets can be defined for other security objectives, *i.e.*, confidentiality, integrity, availability, *etc*. Based on these rules, the SSA is aware of the required security objectives and levels in different situations.

The final step in the Analyze phase is to compare the Indicator results to the required security objectives and levels. If some indicators show that the required security level is not achieved, the Planning phase will be triggered.

Figure 8. An example of context information to define security objectives and levels.



As a whole, the Analyze phase requires a lot of knowledge, which is available from the ontologies. Based on the knowledge, the SSA builds an individual view point of the environment and achieved security. In order to produce the correct results, knowledge maintenance is important. Analysis models and the rule sets for the definition of requirements can be added and updated without modifying the application logic. The utilization of ontologies makes this flexibility possible, *i.e.*, the knowledge is not hard coded inside the application.

3.2.3. Plan

The Plan phase decides how to adapt the SSA when the required securities are not achieved. The Plan phase uses the results from the Analyze phase as input information, *i.e.*, (i) the unfulfilled security objective. (ii) The current and required security level. (iii) The used security mechanism. Furthermore, ontologies offer knowledge to make an adaptation plan. In some situations, only certain security mechanisms are supported, and thus, the Plan phase has to take these restrictions into account.

We have recognized three alternative ways to create the adaptation plan:

(1) The SSA utilizes pre-defined configuration alternatives.

(2) The SSA searches alternative security mechanisms or individual attributes to adapt, based on knowledge from the ISMO.

(3) The SSA asks the user how to proceed.

The first one is the simplest case. At design-time, the architect implements configuration alternatives inside the SSA. Thus, the Plan phase selects one of these pre-defined configurations at runtime. Naturally, dynamism is restricted in this alternative. However, this is enough for simple

devices and applications. As an example, for a situation where the required level of communication confidentiality is not achieved, the pre-defined configuration can be *"Start to use the TLS connection"*.

The second planning alternative changes the security mechanism or adapts individual attributes. When changing the whole security mechanism, the Planner component searches alternative mechanisms from the ISMO. In the ISMO, each security mechanism contains a link to the supported security objectives. For the required security objective, the ISMO can contain several security mechanisms. However, it is probable that only few of those are applicable in the current situation. For instance, the user's device supports fingerprint authentication but the smart space infrastructure does not offer this possibility. Hence, the adaptation plan has to take into account these restrictions from the context information. Alternatively, the Planner component can adapt individual attributes. To achieve this, the SSA searches the causes for the current security level by means of the same Analysis model, which showed that the adaptation was needed. In other words, the Analysis model is used to search Attributes, which have affected the current security level. Therefore, the SSA knows which attribute to adapt in order to affect the security level. Figure 9 illustrates this alternative—colored rectangles refer to the concepts presented in Figure 4. The Analysis model uses Base measures to observe Attributes. If the Attribute can be adapted its *adaptableWith* property shows an action for how to adapt the Attribute. The smart space may contain attributes that affect the achieved security level but all of these cannot be adapted. For instance, if the smart space contains an external threat that cannot be removed other attributes have to be adapted to mitigate threat effects. Finally, the Planner component decides on the adapted Attribute and the Action to be performed. To make this decision the Planner component may utilize goal, constraint or utility function based decision-making. At the moment, our approach is clearly goal orientated, *i.e.*, the context sets the required security objectives and levels (goals) and the purpose of the planning is to find a configuration that satisfies the goal. However, the decision-making algorithm is out of the scope of this paper but the architecture does not restrict decision-making algorithms utilized internally in the Planner component. When the Planner has to take trade-offs into account more sophisticated decision making will be needed, for instance for utility functions.

The third case is for a situation where the SSA is not able to create an enforceable adaptation plan. The following reasons can lead to this alternative: (i) The required knowledge is not defined in the ontologies. (ii) Knowledge is available but creating an adaptation plan would consume too many resources. Therefore, the only way to proceed is to give a warning message to the user and ask for instructions on how to continue.

From these alternatives, the second one is the most dynamic and autonomous without hard-coded adaptation plans. Hence, it is the preferred alternative.

3.2.4. Execute

The final step in the adaptation is the Execute phase, where the created adaptation plan is executed. In other words, it is the straightforward realization of the adaptation plan. At design-time, the software architect has to design variation points inside the SSA. The variation point ensures that it is possible to adapt security mechanisms and attributes at runtime. Figure 9 contains the Action to adapt concept. From the Execute phase point of of view the action is a component, which modifies the related

Attribute. Figure 10 shows a runtime situation, where the Planner component calls the Executor and offers the adaptation plan. Based on the adaptation plan the Executor signals an implementation component.

Figure 9. Planning using an Analysis Model.

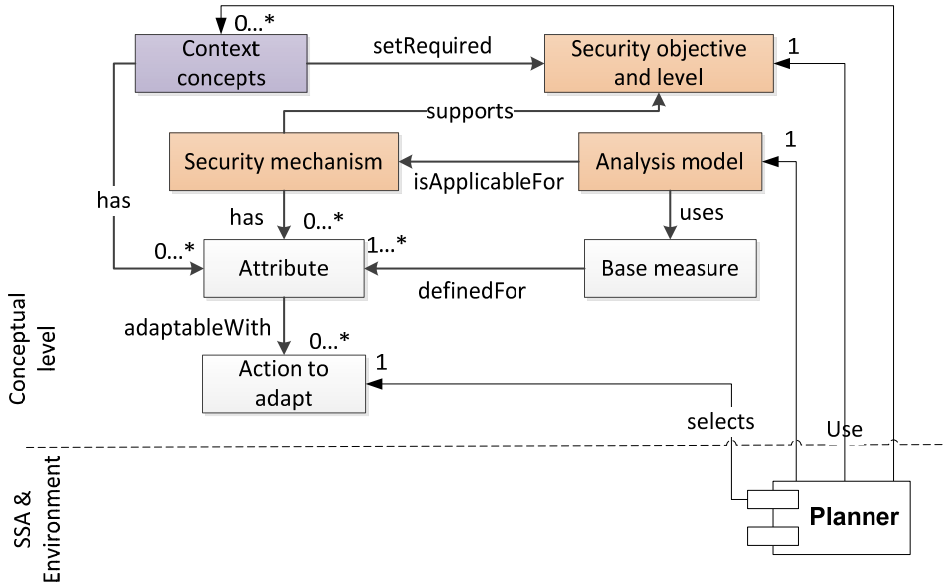
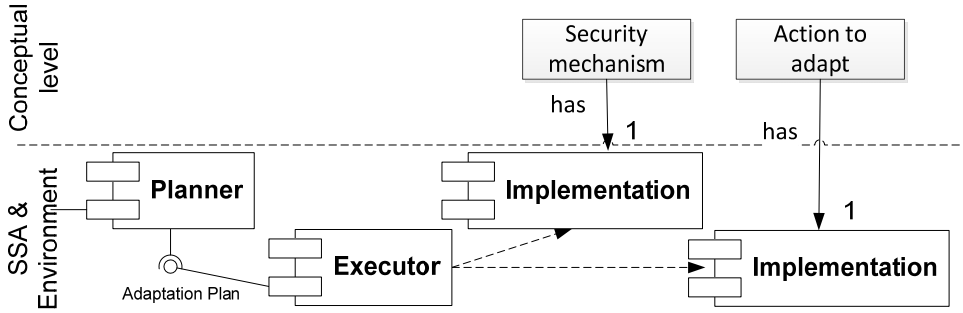


Figure 10. Execute phase.



The execution can affect different layers in a device where the SSA is running. Moreover, the execution can indirectly affect the whole smart space infrastructure. For example, when the SSA adapts a communication protocol the application has to establish a new connection to the smart space infrastructure by using new parameters or a new mechanism. The execute phase can, for example, control how information can be shared by defining security policies according to the Security Control Model described in the next section.

3.3. Runtime Security Control Model

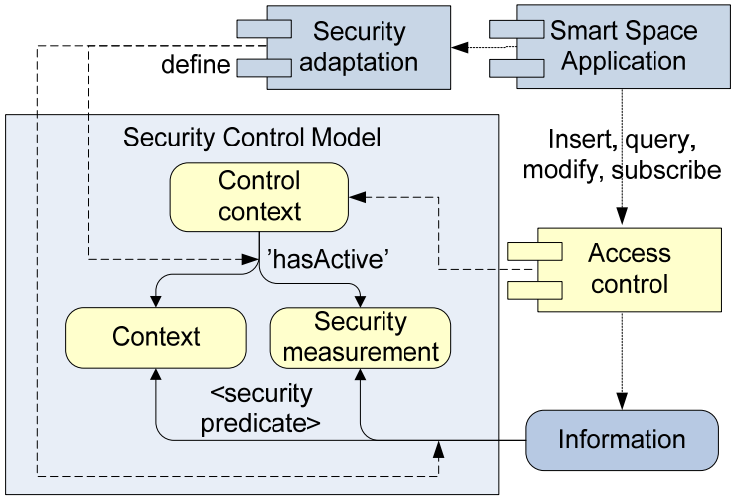
Security knowledge in the conceptual interoperability level provides a means to present security policies, which control the behavior of smart space devices and applications. However, analyzing and planning access-control decisions at runtime, when information is queried and modified, can be

computationally costly. In smart spaces the information is shared using SIBs, which are unaware of applications' conceptual policies and hence unable to enforce these policies. SIBs can be assumed to be aware only of a minimal set of standard security primitives, which are associated to information elements instead of the meaning of this information. In addition, as smart space devices may have limited computing capabilities, solutions based on cryptography are often unfeasible. Therefore, efficient solutions are needed to protect information sharing and to control information access in a fine-grained manner at the level of semantic data.

This subsection generalizes and formalizes our previously presented RDF access control approach [51] into a conceptual security control model. The control model has been verified with RDF but it can be applied to any information presentation system, which is based on subject-predicate-object triples. It specifies how access-control policies and security control information over resources are structured and presented. Runtime costs are minimized by requiring that each policy is presented with a single information triple. The security control model is based on context and security measurement concepts, which are used to authorize actions. Hence, the model can be applied efficiently and flexibly in various dynamic security-control situations.

Figure 11 depicts the security relationships in the security control model. The model has a relationship with three software components, presented in the top-right corner of the figure. *Smart space applications* insert, query, modify or subscribe to information resources. The *access control* component authorizes and controls these operations. Application specific *security adaptation* components administer the behavior of the access-control component. This administering is done by controlling the relationships between resources as specified by the security control model.

Figure 11. Runtime security-control model for smart spaces.



Each piece of information, *i.e.*, each *information* resource, can have a relationship with one or several *context and security measurement* resources. Each relationship presents one access control statement and is described using triplets in the form: "*Information, SecurityPredicate, Context/Measurement*". *Security predicates* are properties that define authorizing or accounting relationships for the security control. Predicates, which are to be used when authorizing transactions, are presented in Table 2. Predicates for accounting can be found in Table 2. *Context/measurement*

refers to any resource that the security adaptation component selects based on the ontologies and policy information from the conceptual level.

When the SSA queries or modifies information, only some contexts and measurements are active. The access control component uses resources, which are active for the application in the current situation. Active resources are found through the *control context* concept, which can be realized as a resource. Security adaptation components define which measurement and context resources are active with triplets: "*ControlContext, 'hasActive'*, *Context/Measurement*". Determination of what resources are active is a dynamic and constantly running process, which may involve different security-adaptation applications. ControlContext resources are fixed in the sense that the access control and security adaption components must know them. For instance, each SSA that is connected to a SIB and has an open communication socket may have a dedicated ControlContext resource. In this case, the active resources could be URIs representing the end-user's identity or security level. These URIs can be resolved and activated by the security adaptation component in the Monitor and Analyze phases, when the user authenticates.

3.3.1. Authorization Predicates

An important use case for the security-control model is authorization over resource access. Policy predicates enabling authorization are defined in Table 1. The granularity of the security-control model protects individual resources and also semantic relationships because of control over access to the resource properties. The security-control model supports the use of allow and disallow policies. Different policies can be used in conjunction to the set conditions of the authorizations (e.g., a user can access information but only if a contextual requirement is met). To prevent contradictory behavior due to the simultaneous use of allow and disallow policies, the proposed approach is that 'disable' policies override "allow" policies.

The security control model enables efficient runtime access control. An access-control component does not need to do heavy reasoning at the time applications are querying or modifying information, instead, security adaptation Analysis and Planning phases can be done in advance when adaptation-triggering events occur. The access control component needs to locate the relevant security relationships, presented with triples, between context or measurement resources and a target sources. When an SSA queries or modifies information, the access control component checks whether there are active policies allowing or denying the action.

When the amount of active and authorizing context and measurement resources is n, the access control component must do at most 2*n truth queries ("is there an allow or deny relationship between the active resource and the accessed resource?") to resolve the authorization of a transaction on a target. The access control component must also find active resources for each used control context resource. Implementations may further speed this up by keeping the list of control context specific active resources in the cached memory.

Table 1. Authorisation policy predicates.

Predicate	Description
GetAllowedFor	Authorizes reading a URI or literal value
SetAllowedFor	Authorizes modifying a URI or literal value
PropertyCreationAllowedFor	Authorizes adding a new URI or literal node under the URI node
PropertyRemovalAllowedFor	Authorizes the removal of a URI or literal node from the URI node
UseAsPropertyAllowedFor	Authorizes use of this node under other URI nodes
GetDisabledFor	Prevents reading a URI or literal value
SetDisabledFor	Prevents modifying a URI or literal value
PropertyCreationDisabledFor	Prevents adding a new URI or literal node under the URI node
PropertyRemovalDisabledFor	Prevents the removal of a URI or literal node from the URI node
UseAsPropertyDisabledFor	Prevents the use of this node under other URI nodes
IsAuthorisedBy	Sets a node under access control and specifies authority. There may be several authorities in one broker.

Table 2. Predicates for access control accounting.

Predicate	Description
HasBeenAuthoredBy	Identifies a resource's author
HasAddedPredicate	Identifies authors who have added predicates under the resource
IsSignedWith	Link to a signature proving authenticity and the origin of the resource
HasSecurityContext	Link to any security measurement or context resource which was active when the data was stored (needed to verify e.g. trustworthiness of data)
IsAuthorisedBy	Specifies the authority that controls security. If such relationship to a known security authority is missing, access can be directly authorized without any other checks.
CanBeMonitored	Allows or disallows logging (e.g. due to performance or privacy)
HasBeenReadBy	Identifies contexts (users) where data has been successfully queried
HadInvalidReadAttemptBy	Identifies contexts (users) with rejected read requests
HadInvalidWriteAttemptBy	Identifies contexts (users) who have made rejected write requests

3.3.2. Accounting Predicates

In addition to authorization, the security control model supports other real-time security control situations. Table 2 presents predicate definitions for access accounting activities, which are needed to determine the authenticity or trustworthiness of information. The table defines the relationships for accounting predicates, which are used to log access requests, both successful and unsuccessful. This

information is needed, e.g., when trying to detect malicious or harmful modifications and intrusions and when reasoning which nodes may have been potentially compromised due to harmful information. The table also lists *IsSignedWith* and *HasSecurityContext* predicates, which users can use to verify the authenticity and trustworthiness of information. Trustworthiness may depend on context or measurement, which were active when the information was stored.

4. Implementation of Adaptive Security

4.1. Case Description

The validation was based on a use case from the smart space pilot—called Seamless Usage of Multiple Smart Spaces (SUM-SS) [55]. The SUM-SS pilot combines four smart spaces, *i.e.*, smart personal space, smart home, smart office and smart city. An end user facilitates information from these smart spaces via his/her mobile phone, which constitutes his/her personal smart space. In the personal smart space, the user is able to store information from other smart spaces, like calendar information and documents from the office space. The home smart space offers capabilities to monitor energy consumption; control light and wall sockets and control home automation via the Lon network etc. The smart city offers public information and facilitates everyday life in an urban environment—for instance, by offering information on parking areas and traffic jams. Furthermore, mobile devices and televisions are able to consume entertainment content from a cloud, which is supported by the Cam4Home platform [56]. Consequently, the SUM-SS pilot opens up possibilities to select a smart space use case for the validation purposes. Hence, the use case selected for the validation purposes concentrates on illustrating the following issues:

1. The SSA running in an end user's mobile phone utilizes the adaptation loop to ensure an appropriate authentication level in different situations.
2. Security- and context-related knowledge is retrieved from ontologies.
3. RIBS controls access over shared information by using the security control model.

Use case description: The homeowner leaves the home in the morning, by car. During the drive she wants to check that the front door of her house is locked properly. The owner is able to check a lock status via her mobile phone without any additional authentication.

During the working day, a maintenance man arrives on the front door of the house and rings a doorbell. The doorbell sound is played in the owner's mobile phone and a video stream from the front door of the house is delivered. Hence, when she recognizes the maintenance man at the door, she is able to open the door remotely. However, her current authentication level is not strong enough for the remote door opening, and thus, re-authentication is requested.

After a while, the maintenance man is ready and leaves the house. The owner is informed and she locks the front door again. Now the time has passed and the authentication level is dropped. Nevertheless, re-authentication is not needed because the door can be locked with a lower authentication level.

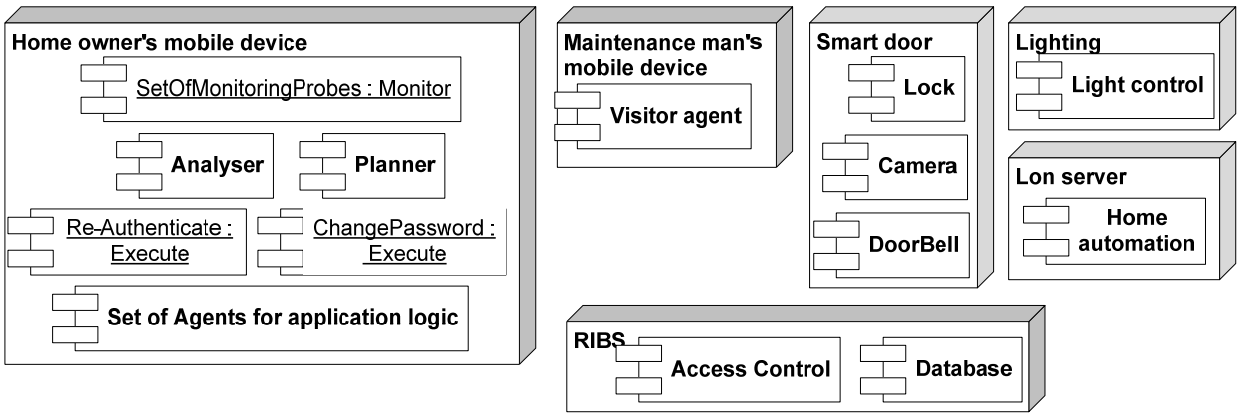
The home owner arrives home and opens the front door locally by utilizing the NFC (Near Field Communication) feature of her mobile phone. The mobile phone authenticates the user and then the mobile is authenticated through NFC. Inside the house, the owner adjusts the lighting—and the earlier

achieved authentication level is enough for these actions. After a while, the owner wants to turn down the heating system in the house. However, controlling the home automation system is a critical action and too much time has passed from the last authentication. Thus, authentication with a stronger authentication level is required.

4.2. Case Implementation

Figure 12 shows the deployment of the use case from the security adaptation viewpoint. The use case implements user authentication in an adaptive manner by using password-based authentication. Our previous demonstration utilized a gait-based authentication [57]. The construction contains six nodes, *i.e.*, homeowner's mobile device, RIBS, smart door, lighting, Lon server and the maintenance man's mobile device. The main actions in the use case are performed with the homeowner's mobile device Nokia C7. Hence, it contains application logic agents to retrieve and insert information into the RIBS. These agents are implemented with Qt C++. In this case, the adaptation will be performed from the homeowner's viewpoint, and thus, adaptation-related components are located in her device. The smart door node contains lock, camera and doorbell agents, which offer related functionalities. Similarly, the lighting node and the Lon server node contain agents to utilize those devices. The RIBS is executed inside a WLAN access point—in order to offer a good connectivity. The RIBS supports the Transport Layer Security (TLS) protocol [58] to secure communication between agents and the RIBS. The last node is the maintenance man's mobile device that contains a visitor agent. The visitor agent makes it possible to ring the doorbell that is available publicly from the home smart space.

Figure 12. The deployment of the use case.

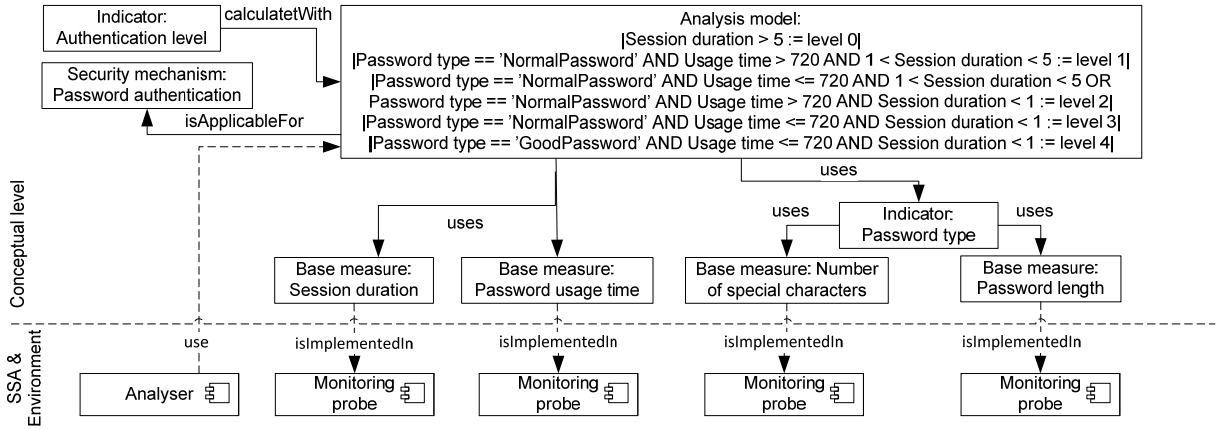


Monitoring probes are components that implement measurement methods for base measures (*c.f.*, Figure 6). For password-based authentication the use case utilizes the following base measures: (1) Password length. (2) The number of special characters in the password. (3) Password usage time. (4) Session duration. The change of these base measures is informed to the Analyze component. Naturally, Base measures 1 and 2 change when the password is changed. In contrast, Base measures 3 and 4 change constantly, and thus, the changed values are informed to the Analyze component at a certain intervals.

The Analyze component combines the monitoring results to authentication level indicator by means of the Analysis model (*c.f.*, Figure 7). The Analyzer component retrieves the right analysis model from

the ISMO. Analysis models are described by a natural language, which combines English and Boolean algebra. We made this decision in our previous work [38] in order to facilitate the preparation of Analysis models. Thus, Analysis models can be modified and added without experience of ontology query languages. The Analysis model utilized in this case is presented in Figure 13. The analysis model uses four base measures—either directly or via the Password type indicator—as depicted in the figure. Moreover, the figure presents Monitoring probes, which implement measurement methods for base measures.

Figure 13. Analysis model used in the use case.



The above-presented Analysis model produces the achieved authentication level. Moreover, the Analyze component analyzes the current situation in order to decide the required authentication level. For this purpose, rules presented in Figure 8 are utilized.

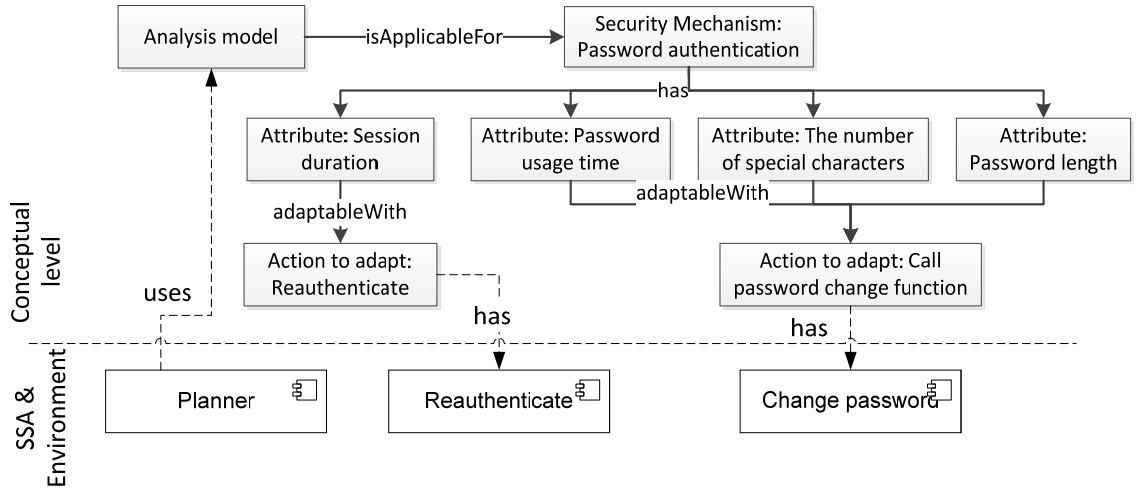
Table 3 summarizes user actions, situations and the achieved and required authentication levels. The achieved authentication level means the level before adaptation. Hence, Actions 1, 3 and 5 do not require adaptation because the achieved level is higher than the required level. Other actions require adaptation and the Planner component is called.

The Plan component selects how the adaptation is performed. In this case, the Planner component searches the causes of the current security level from the analysis model (*c.f.*, Figure 9). Figure 14 lists the attributes of password authentication. In the ISMO base measures from Figure 13 are connected to these attributes by means of a *definedFor* property. Now there are two possible ways to adapt. Firstly, the re-authentication of the user affects the session duration attribute. Secondly, calling the change password function affects password usage time—and depending on a new password—the length and number of special characters attributes. Based on this knowledge, the Planner component selects an appropriate action to adapt. Table 3 listed user actions. Action numbers 2 and 4 are handled with re-authenticate adaptation. The user action number 6 leads to the change-password adaptation action because the one-month usage time, *i.e.*, 720 h, boundary has been exceeded.

Table 3. Achieved and required authentication levels in different situations.

Action	Situation and Input Information for the Analysis Model	Achieved Auth. Level	Required Auth. Level
1. Check the lock status	Check the lock status Normal password selected 708 h ago. Session duration: 2 h	2	0
2. Open the front door	Remote opening Normal password selected 710 h ago. Session duration: 4 h	2	3
3. Lock the front door	Remote locking Normal password selected 712 h ago. Session duration: 2 h	2	1
4. Open the front door	Local opening Normal password selected 719 h ago. Session duration: 7 h	0	3
5. Modify lighting	Local modification Normal password selected 719 h ago. Session duration: 0.02 h	3	1
6. Modify home automation	Local modification Normal password selected 721 h ago. Session duration: 2 h	1	2

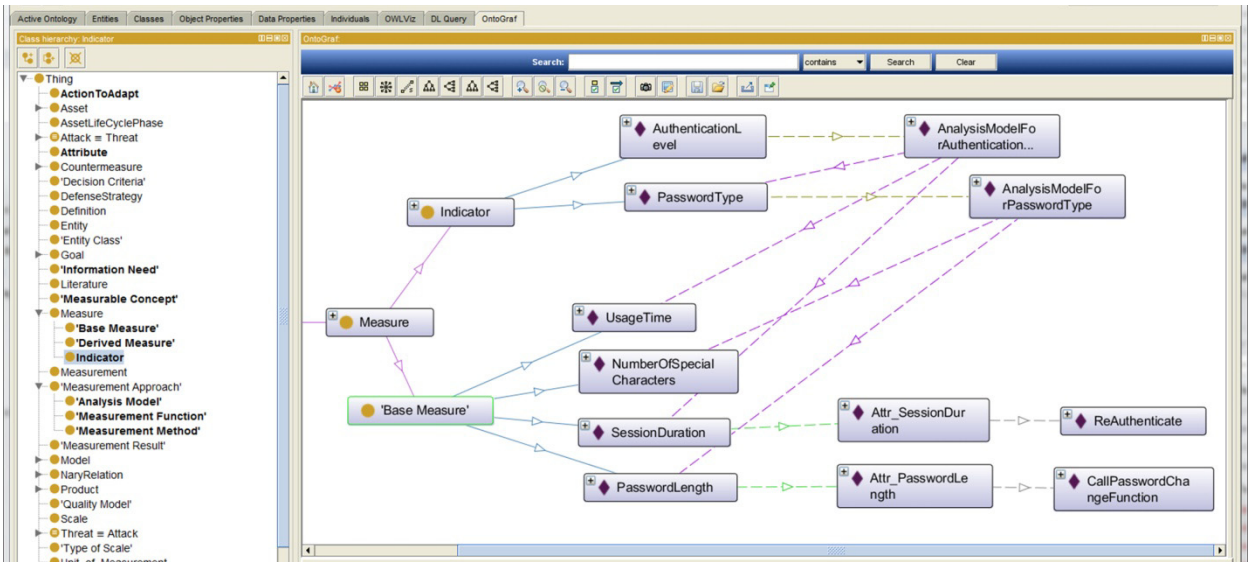
Figure 14. Actions to adapt different parameters.



Lastly, the Execute component performs the created adaptation plan. Both adaptation actions also have an effect outside of the homeowner's device: It requires re-authentication into the RIBS or calling the password-change function from the RIBS. However, in both cases the initiative for these actions is made in the Execute component inside the homeowner's device.

Figure 15 shows the screenshot of the ISMO in the Protégé ontology tool. The screenshot contains concepts instantiated for the use case purposes. The ISMO is also available in web: https://sofia-community.com/projects/sontologies/—needs registration. The page contains ISMO and links to imported ontologies.

Figure 15. ISMO in Protégé.



The RIBS is responsible for enforcing access control over brokered information. It enforces that only authenticated and authorized users can insert and modify information. The authorization checks follow the runtime security-control model, as illustrated in Figure 16. The agents in the homeowner's terminal are responsible for administering the authorization policies, which are stored in the RIBS for each RDF resource. When a user is authenticated, appropriate context and measurement resources are activated. In the use case, the maintenance man is mapped to a visitor context and the homeowner is mapped to a resource, which represents owner's identity. Further, all users are mapped to security measurement resources, which describe the authentication level. When users query or modify information, the SIB checks whether these active RDF resources authorize access to the requested resources. All authenticated users are given access to non-critical information inside the home e.g., the lighting. The homeowner has access to every piece of information. However, access to the most critical information requires that the owner have a sufficient authentication level.

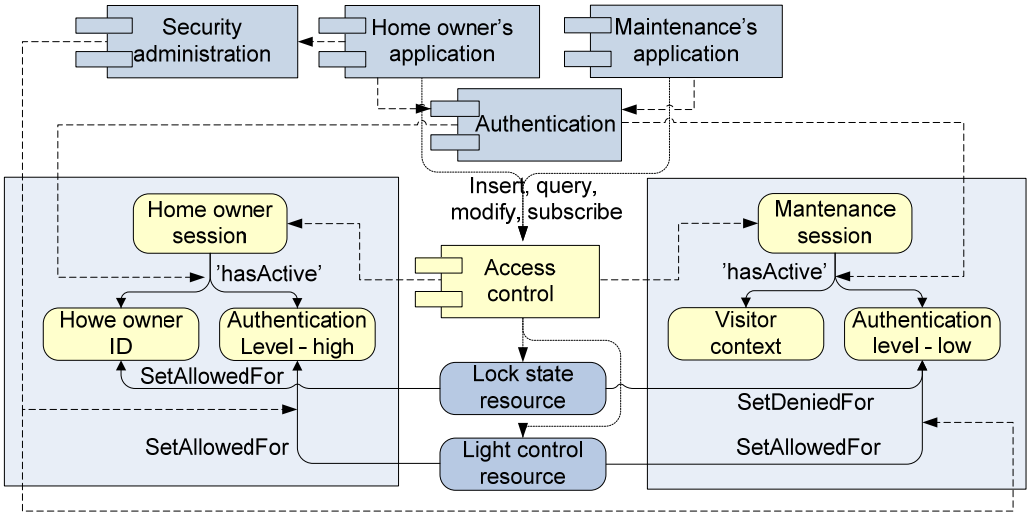
The RIBS has been optimized to provide fast and low-power consuming information access. The implementation indexes all incoming RDF data, and thus, enables RDF URIs as well as literals to be directly addressed. Relationship information is stored to a three-dimensional (subject-predicate-object) array, *i.e.*, to a bit cube. As security policies are presented with a single bit, which is either on or off, they can be quickly checked and the amount of required memory will not increase even when the security configuration becomes more complex.

4.3. Lessons Learned

Implementing and performing these use cases showed several advantages from the presented approach. Firstly, all knowledge is retrieved from ontologies. Thus, content from the analysis model (*c.f.*, Figure 13) and rules for setting the required authentication levels (*c.f.*, Figure 8) do not need to be coded into the SSA. Hence, knowledge can be modified and extended without coding work. Secondly, adaptive user authentication enhanced usability. However, as the authentication level decreased the user was able to perform Action 3 (see Table 3). In contrast, using a static authentication requires that

all actions be set as equally critical, which in turn means that all actions require the highest authentication level. Thirdly, from the implementation view point the presented adaptation loop clearly defined the required components. In addition, knowledge—required in each component—was defined explicitly. Therefore, the approach description can act as an implementation guideline. Finally, the security-control model in the RIBS supported flexible access control. The access control information is described in the RIBS as RDF information, and thus, separated access-control lists are not needed.

Figure 16. Examples of authorizing relations in the runtime security-control model.



However, improvement ideas are also recognized. The first consideration relates to the selected security objective—in the use case user authentication was in the focal point. Supporting other security objectives does not require changes to the adaptation approach and utilization of knowledge from the ontologies. Nevertheless, the content of the ontologies has to be extended with new knowledge related to security objectives. This knowledge does not need to be created from scratch but the existing knowledge has to be described in a machine-readable form by means of the ontologies. The second issue relates to the definition of the Analysis model. During the case implementation it was noticed that defining the Analysis model is a complex and time-consuming task—although only three variables were used. Thus, automation is needed in the future in order to define extensive analysis models.

5. Discussion

5.1. The Advantages of the Approach

Achieving security in smart spaces requires dynamic security solutions. The presented adaptation approach for security ensures that the achieved security corresponds to each situation faced in the smart space. Further, the introduced security-control model ensures the appropriate information access in changing smart spaces. Thus, the introduced security-adaptation approach offers several advantages:

Firstly, the approach is generic—security objectives and mechanisms can be freely selected. Hence, the approach is able to offer adaptation for one security objective, or alternatively, a set of security

objectives can be implemented in an adaptive manner. Similarly, the adaptation approach is independent on the used security mechanism.

Secondly, the adaptation approach is based on knowledge from ontologies, which cover the knowledge part of the MAPE-K model. This separation of concerns, *i.e.*, separation of generic knowledge from application logic, improves reusability. The knowledge can be updated and extended easily and quickly without modifying the SSA. Moreover, the ontologies ensure that there is a uniform way to present security terminology.

Thirdly, the commonly known MAPE model ensures that the required components are clearly defined, *i.e.*, Monitoring, Analyzing, Planning and Executing. Clearly defined components support reusability, which facilitates the architects' work. The Monitoring utilizes security measures and the Analysis utilizes extendable analysis models. In each phase the required knowledge is retrieved from the ontologies.

Fourthly, smart spaces consist of devices and applications that can enter and leave the space at any time. Various situations require dynamic access-control policies that are enforced automatically. The security-control model provides an approach for enforcing the access control for different situations. The model is expressive but still straightforward, and hence, more suitable for runtime enforcement than the previous RDF level access-control proposals.

5.2. The Challenges of the Approach and Future Research

Firstly, the use cases showed that creating analysis models is a complex task. Therefore, a tool, or at least guidelines, for the analysis model definition is needed in the future. The tool has to present the possible variables and their value ranges. In addition, the tool has to check that contradicting analysis models are not created.

Secondly, in the future mutual relationships between security objectives have to be taken into account during the Analyze and Plan phases. The ISMO already contains basic connections between security objectives, e.g., authorization demands identification. Therefore, analysis models have to notice these dependencies when recognizing adaptation needs. Similarly, the Plan phase is able to utilize this knowledge when searching applicable adaptation actions. For instance, the need to adapt authorization causes the utilized identification scheme to change. Furthermore, trade-offs and dependencies on other qualities like performance, reliability and usability are topics for future research.

Thirdly, the performance cost of the approach is a natural question. However, end users were not able to recognize decreased performance during the case study. It is clear that the performance overhead can be noticed if a huge amount of base measures and complex analysis models are used. Therefore, it is important to adjust the analysis models and the number of base measures for device resources. The performance penalty caused by the RDF access control depends on the amount of requested RDF resources. In a performance test case with the RIBS implementation, the average request times were around one per cent longer when compared to a case where all requests were authorized without any checks [51].

The last issue relates to the security of the adaptation approach. It is possible that an attacker may try to modify some parts of the adaptation loop in order to attack the smart space. For instance,

manipulating the Monitoring or Analyzing parts might mean that a decreased security level is not recognized. Or alternatively, the Plan phase may create an inappropriate adaptation plan, which is advantageous for the attacker. Hence, it is extremely important to protect these components and ensure the authenticity and integrity of knowledge.

5.3. The Maturity of the Approach

The approach was developed during a three-year research project. The development was performed incrementally by utilizing different use cases and smart space set-ups. In previous cases, we have experienced different adaptation ideas and worked on different phases of security adaptation. Table 4 summarizes the previous validation cases.

Table 4. Previous validation cases.

Validation Case	Description
Risk-based security adaptation in a greenhouse [41,57].	A greenhouse with a shopping area constitutes a public smart space. In the smart space threats increase the risk levels and security mechanisms decrease risks. Hence, the monitoring concentrates on recognizing threats. In this case, confidentiality and integrity were considered. Furthermore, users authenticated by means of gait information identified from the measurements of acceleration sensors inside the mobile phone.
Adaptive user authentication [38].	The first case that utilized knowledge from the ISMO. It adapts user authentication by monitoring authentication-related measures: password length, age, variation of characters and session duration. Important information was available only when an acceptable authentication level was reached. The user's re-authentication was requested when the session duration was exceeded.
Role- and popularity-based access-control simulations [51,59].	Controlling access to information according to the user's role or popularity of information. Popularity is a measure that indicates how many readers or how many authors an RDF resource has. These adaptation cases were simulated with the smodels logic solver.
Adding new knowledge into the ISMO [54].	The paper and related case example showed how easily knowledge in the ISMO can be extended. Moreover, design steps to develop adaptive security were presented.

6. Conclusions

In smart spaces, it is not possible to take all the security requirements into account at design-time. Hence, this paper presented a self-adaptation approach for smart space security. The presented approach contains an adaptation loop the Monitor, Analyze, Plan, and Execute model—in a clearly defined form. The monitor phase utilizes monitoring probes to observe security-relevant attributes from the smart space and smart space application. The monitored results are analyzed in order to reveal if the required security is not achieved. The Plan phase creates an adaptation plan, which will be enforced in the Execute phase. The adaptation approach requires a lot of knowledge, which is retrieved from the ontologies. The utilization of ontologies ensures a flexible and extensible way to manage and

use knowledge in machine-readable form. The ISMO provides security- and measuring-related knowledge and CO4SS offers context knowledge. For access control, the security-control model was presented, which utilizes context information and provides dynamic access control. Hence, the security-control model provides a flexible and efficient mechanism to control information sharing in smart spaces.

The presented approach was validated by means of a use case. The use case illustrated (i) all the phases of the adaptation loop, (ii) how ontologies offer knowledge for adaptation at runtime, (iii) that access control enforced the semantic information. The advantages of the presented approach are evident. Firstly, the approach is independent of security objectives and mechanisms. Second, the approach provides a reusable architecture to develop adaptive security applicable for different kinds of smart spaces. Thirdly, the components for the security adaptation are clearly defined, which help in adopting the approach. Finally, the utilization of ontologies ensures that knowledge can be updated and extended easily.

In the future, new and wider analysis models are needed. Thus, a tool will be developed for defining a wider set of analysis models for various security objectives.

Acknowledgments

This work has been carried out in the SOFIA ARTEMIS project (2009-2011) and SASER-Siegfried Celtic-Plus project (2012-2015) funded by Tekes (the Finnish Funding Agency for Technology and Innovation), VTT Technical Research Centre of Finland and the European Commission.

Conflict of Interests

The authors declare no conflict of interests.

References

1. Conti, M.; Das, S.K.; Bisdikian, C.; Kumar, M.; Ni, L.M.; Passarella, A.; Roussos, G.; Tröster, G.; Tsudik, G.; Zambonelli, F. Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence. *Pervasive Mob. Comput.* **2012**, *8*, 2–21.
2. Elkhodary, A.; Whittle, J. A Survey of Approaches to Adaptive Application Security. In *Proceedings of the International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, Minneapolis, USA, 20-26 May, 2007; pp. 16–23.
3. Yuan, E.; Malek, S. A taxonomy and survey of self-protecting software systems. In *Proceedings of the IEEE Software Engineering for Adaptive and Self-Managing Systems*, Zürich, Switzerland, 4-5 June, 2012; pp. 109–118.
4. Cook, D.J.; Das, S.K. How smart are our environments? An updated look at the state of the art. *Pervasive Mob. Comput.* **2007**, *3*, 53–73.
5. Ovaska, E.; Salmon Cinotti, T.; Toninelli, A. Design principles and practices of interoperable smart spaces. In *Advanced Design Approaches to Emerging Software Systems: Principles, Methodologies, and Tools*; Liu, X., Li, Y., Eds.; IGI Global, 2011; pp. 18–47.

6. Pantsar-Syväniemi, S.; Purhonen, A.; Ovaska, E.; Kuusijärvi, J.; Evesti, A. Situation-Based and Self-Adaptive Applications for Smart Environment. *J. Ambient Intelligence Smart Environ.* **2012**, *4*, 491–516.
7. Honkola, J.; Laine, H.; Brown, R.; Tyrkkö, O. Smart–M3 information sharing platform. In *Proceedings of the IEEE Symposion on Computers and Communications*, Riccione, Italy, 22–25 June, 2010; pp. 1041–1046.
8. RDF Primer. 2004. Available online: http://www.w3.org/TR/rdf-primer/ (accessed on 23 November 2012).
9. SOFIA Smart Objects For Intelligent Applications. www.sofia-project.eu, 2012. Accessed 23 November 2012.
10. Suomalainen, J.; Hyttinen, P.; Tarvainen, P. Secure information sharing between heterogeneous embedded devices. In *Proceedings of the 4th European Conference on Software Architecture: Companion Volume*, Copenhagen, Denmark, 23-26 August, ACM, 2010; pp. 205–212.
11. Kephart, J.O.; Chess, D.M. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50.
12. Dobson, S.; Denazis, S.; Fernández, A.; Gaïti, D.; Gelenbe, E.; Massacci, F.; Nixon, P.; Saffre, F.; Schmidt, N.; Zambonelli, F. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.* **2006**, *1*, 223–259.
13. Salehie, M.; Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.* **2009**, *4*, 1–42.
14. Psaier, H.; Dustdar, S. A survey on self-healing systems: approaches and systems. *Computing* **2011**, 91, 43–73.
15. Huebscher, M.C.; McCann, J.A. A survey of autonomic computing-degrees, models, and applications. *ACM Comput. Surv.* **2008**, *40*, 1–28.
16. Matinlassi, M.; Niemelä, E. The impact of maintainability on component-based software systems. In *Proceedings of the 29th Euromicro Conference*, Belek-Antalya, Turkey, 3–5 September, IEEE, 2003; pp. 25–32.
17. Hashii, B.; Malabarba, S.; Pandey, R.; Bishop, M. Supporting reconfigurable security policies for mobile programs. *Computer Networks* **2000**, *33*, 77–93.
18. Hu, W.; Hiser, J.; Williams, D.; Filipi, A.; Davidson, J.W.; Evans, D.; Knight, J.C.; Nguyen-Tuong, A.; Rowanhill, J. Secure and practical defense against code-injection attacks using software dynamic translation. In *Proceedings of the 2nd international conference on Virtual execution environments*, Ottawa, Canada, 14-16 June, ACM, 2006; pp. 2–12.
19. Knight, J.C.; Strunk, E.A. Achieving critical system survivability through software architectures. In Architecting Dependable Systems II, Lemos R., Gacek C., Romanovsky A., Eds.; Springer: Berlin-Heidelberg, Germany, 2004; pp. 51–78.
20. Ryutov, T.; Zhou, L.; Neuman, C.; Leithead, T.; Seamons, K.E. Adaptive trust negotiation and access control. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*, Stockholm, Sweden, 1-3 June 2005; pp. 139–146.
21. Klenk, A.; Niedermayer, H.; Masekowsky, M.; Carle, G. An architecture for autonomic security adaptation. *Ann. Telecommun.* **2006**, *61*, 1066–1082.

22. Hulsebosch, R.; Bargh, M.; Lenzini, G.; Ebben, P.; Iacob, S. Context sensitive adaptive authentication. In *Smart Sensing and Context*; Kortuem, G., Finney, J., Lea, R., Sundramoorthy, V., Eds.; Springer: Berlin-Heidelberg, Germany, 2007; pp. 93–109.
23. Abie, H.; Savola, R.M.; Bigham, J.; Dattani, I.; Rotondi, D.; Da Bormida, G. Self-Healing and Secure Adaptive Messaging Middleware for Business-Critical Systems. *Int. J. Adv. Se.* **2010**, *3*, 34–51.
24. Savola, R.; Abie, H. Development of measurable security for a distributed messaging system. *Int. J. Adv. Se.* **2009**, *2*, 358–380.
25. Wang, C.; Wulf, W. A. Towards a Framework for Security Measurement. In *Proceedings of the 20th National Information Systems Security Conference*, Baltimore, Maryland, USA, October 1997; pp. 522–533.
26. García, F.; Bertoa, M.F.; Calero, C.; Vallecillo, A.; Ruíz, F.; Piattini, M.; Genero, M. Towards a consistent terminology for software measurement. *Inf. Softw. Technol.* **2006**, *48*, 631–644.
27. Haley, C.B.; Laney, R.; Moffett, J.D.; Nuseibeh, B. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Trans. Softw. Eng.* **2008**, *34*, 133–153.
28. Salehie, M.; Pasquale, L.; Omoronyia, I.; Ali, R.; Nuseibeh, B. Requirements-driven adaptive security: Protecting variable assets at runtime. In *Proceedings of the 20th International Requirements Engineering Conference (RE)*, Chicago, USA, 24–28 September, IEEE, 2012; pp. 111–120.
29. ISO/IEC 15408-1:2009 Standard. Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and general model, International Organization of Standardization, 2009.
30. Sahinoglu, M. Security meter: a practical decision-tree model to quantify risk. *Security Privacy* **2005**, *3*, 18–24.
31. Zhou, J. Knowledge Dichotomy and Semantic Knowledge Management. In *Proceedings of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web*, Jyväskylä, Finland, 25–27 August, Springer, USA, 2005; pp. 305–316.
32. Blanco, C.; Lasheras, J.; Valencia-García, R.; Fernández-Medina, E.; Toval, A.; Piattini, M. A systematic review and comparison of security ontologies. In *Proceedings of the 3rd International Conference on Availability, Security, and Reliability*, Barcelona, Spain, 4–7 March, IEEE, 2008; pp. 813–820.
33. Evesti, A.; Ovaska, E.; Savola, R. From security modelling to run-time security monitoring. In *Proceedings of the European Workshop on Security in Model Driven Architecture, CTIT Centre for Telematics and Information Technology*, Enchede, Netherlands, 23–26 June, 2009; pp. 33–41.
34. Kim, A.; Luo, J.; Kang, M. Security Ontology for annotating resources. In *Proceedings of the On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, Agia Napa, Cyprus, 31 October–4 November, Springer: Berlin-Heidelberg, Germany, 2005; pp. 1483-1499.
35. Denker, G.; Kagal, L.; Finin, T. Security in the Semantic Web using OWL. *Inform. Sec. Tech. Rep.* **2005**, *10*, 51–58.

36. Savolainen, P.; Niemelä, E.; Savola, R. A taxonomy of information security for service centric systems. In *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, Lübeck, Germany, 27–31 August, IEEE, 2007; pp. 5–12.
37. Herzog, A.; Shahmehri, N.; Duma, C. An ontology of information security. *J. Inform. Sec. Privacy* **2007**, *1*, 1–23.
38. Evesti, A.; Savola, R.; Ovaska, E.; Kuusijärvi, J. The Design, Instantiation, and Usage of Information Security Measuring Ontology. In *Proceedings of the 2nd International Conference on Models and Ontology-based Design of Protocols, Architectures and Services*, Budapest, Hungary, 17-22 April, IARIA, 2011; pp. 1–9.
39. Pantsar-Syväniemi, S.; Kuusijärvi, J.; Ovaska, E. Supporting Situation-awareness in Smart Spaces. In *Proceedings of the International Workshops, S3E, HWTS, Doctoral Colloquium, Held in Conjunction with GPC 2011*, Oulu, Finland, 11–13 May 2011, Springer: Berlin-Heidelberg, Germany, 2012; pp. 14–23.
40. Evesti, A.; Pantsar-Syväniemi, S. Towards micro architecture for security adaptation. In *Proceedings of the 4th European Conference on Software Architecture, Companion Volume*, Copenhagen, Denmark, 23–26 August, ACM, 2010; pp. 181–188.
41. Evesti, A.; Ovaska, E. Ontology-Based Security Adaptation at Run-Time. In *Proceedings of the 4th International Conference on Self-Adaptive and Self-Organizing Systems*, Budapest, Hungary, 27 September–1 October 2010, IEEE, 2010; pp. 204–212.
42. Dietzold, S.; Auer, S. Access control on RDF triple stores from a semantic wiki perspective. In *Proceedings of the Scripting for the Semantic Web Workshop at 3rd European Semantic Web Conference*, Budva, Montenegro, 11–14 June 2006, CEUR Workshop Proceedings, 2006; pp. 1–9.
43. D'Elia, A.; Honkola, J.; Manzaroli, D.; Salmon Cinotti, T. Access Control at Triple Level: Specification and Enforcement of a Simple RDF Model to Support Concurrent Applications in Smart Environments. In *Proceedings of the 11th International Conference, NEW2AN 2011, and 4th Conference on Smart Spaces, ruSMART 2011*, St. Petersburg, Russia, 22–25 August 2011, Springer: Berlin-Heidelberg, Germany, 2011; pp. 63–74.
44. Reddivari, P.; Finin, T.; Joshi, A. Policy-based access control for an RDF store. In *Proceedings of the Policy Management for the Web*, Chiba, Japan, 10–14 May 2005; pp. 78–81.
45. Jain, A.; Farkas, C. Secure resource description framework: an access control model. In *Proceedings of the 11th symposium on Access control models and technologies*, Lake Tahoe, CA, USA, 7–9 June 2006, ACM, 2006, pp. 121–129.
46. Flouris, G.; Fundulaki, I.; Michou, M.; Antoniou, G. Controlling access to RDF graphs. In *Proceedings of the Future Internet - FIS 2010*, Berlin, Germany, 20–22 September 2010, Springer: Berlin-Heidelberg, Germany, 2010, pp. 107–117.
47. Kim, J.; Jung, K.; Park, S. An Introduction to Authorization Conflict Problem in RDF Access Control. In *Proceedings of the Knowledge-Based Intelligent Information and Engineering Systems*, Zagreb, Croatia, 3-5 September 2008, Springer: Berlin-Heidelberg, Germany, 2008; pp. 583–592.
48. Cho, E.; Kim, Y.; Hong, M.; Cho, W. Fine-Grained View-Based Access Control for RDF Cloaking. In *Proceedings of the 9th International Conference on Computer and Information Technology*, Xiamen, China, 11-14 October 2009, IEEE, 2009; pp. 336–341.

49. Bock, J.; Haase, P.; Ji, Q.; Volz, R. Benchmarking OWL reasoners. In *Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*, Tenerife, Spain, CEUR Workshop Proceedings, 2008; pp. 1–15.
50. Dentler, K.; Cornet, R.; Ten Teije, A.; De Keizer, N. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web* **2011**, *2*, 71–87.
51. Suomalainen, J.; Hyttinen, P. Security Solutions for Smart Spaces. In *Proceedings of the 11th International Symposium on Applications and the Internet*, Munich, Germany, 18-21 July 2011, IEEE, 2011; pp. 297–302.
52. Niemelä, E.; Evesti, A.; Savolainen, P. Modeling quality attribute variability. In *Proceedings of the 3rd International Conference on Evaluation of Novel Approaches to Software Engineering*, Funchal, Portugal, 4-7 May 2008; pp. 169–176.
53. Ovaska, E.; Evesti, A.; Henttonen, K.; Palviainen, M.; Aho, P. Knowledge based quality-driven architecture design and evaluation. *Inf. Softw. Technol.* **2010**, *52*, 577–601.
54. Evesti, A.; Ovaska, E. Design Time Reliability Predictions for Supporting Runtime Security Measuring and Adaptation. In *Proceedings of the 3rd International Conference on Emerging Network Intelligence*, Lisbon, Portugal, 20–25 November 2011, IARIA, 2011; pp. 94–99.
55. Sofia Pilot Brochure. 2012. Available online: http://www.slideshare.net/sofiaproject/sofia-project-brochure-pilots-set (accessed on 23 November 2012).
56. Cam4Home Project. 2012. Available online: http://www.cam4home-itea.org/ (accessed on 8 May 2012).
57. Evesti, A.; Eteläperä, M.; Kiljander, J.; Kuusijärvi, J.; Purhonen, A.; Stenudd, S. Semantic Information Interoperability in Smart Spaces. In *Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia*, Cambride, UK, 22-25 November 2009, ACM, 2009; pp. 158–159.
58. Dierks, T. and Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. 2008. Available online: http://www.ietf.org/rfc/rfc5246.txt (accessed on 23 November 2012).
59. Suomalainen, J. Flexible Security Deployment in Smart Spaces. In *Proceedings of the International Workshops, S3E, HWTS, Doctoral Colloquium, Held in Conjunction with GPC 2011*, Oulu, Finland, 11–13 May 2011, Springer: Berlin-Heidelberg, Germany, 2012; pp. 34–43.