Project no: 100204

pSHIELD

pilot embedded Systems architecture for multi-Layer Dependable solutions

Instrument type: Capability Project

Priority name: Embedded Systems (including RAILWAYS)

M0.1: Formalized conceptual models of the key pSHIELD concepts

Due date of deliverable: 15th April 2011
Actual submission date: 15th April 2011

Start date of project: 1st June 2010 Duration: 12 months

Organisation name of lead contractor for this deliverable: pSHIELD Consortium

Revision [Final]

Project co-funded by the European Commission within the ARTEMIS-JU (2007-2013)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Andrea Fiaschetti	University of Rome		
Vincenzo Suraci	University of Rome		
Francesco Delli Priscoli	University of Rome		
Andi Palo	University of Rome		
Fabrizio M. de Seta	Elsag Datamat		
Renato Baldelli	Elsag Datamat		
Andrea Morgagni	Elsag Datamat		
Emilio Bisbiglio	SESM		
Ezio Rossi	SESM		
João Cunha	SESM		
Fabio Giovagnini	SESM		
Mohammad M. R. Chowdhury	CWIN		
Jose Verissimo	Critical Software		
Gareth May-Clement	Critical Software		
Josef Noll	Movation AS		
Spase Drakul	THYIA		
Gordana Mijic	THYIA		
Nastja Kuzmin	THYIA		
Ljiljana Mijic	THYIA		
Andrea Taglialatela	TRS		
Luca Bixio	University of Genova		
Mirco Raffetto	University of Genova		
Carlo Regazzoni	University of Genova		
Paolo Azzoni	Eurotech S.p.a.		
Marco Cesena	Selex Communications		
Reviewed by			
Name	**Company**		
Antonio Pietrabissa	University of Rome		
Claudio De Persis	University of Rome		
Alessandro De Carli	University of Rome		
Approved by			
Name	**Company**		
Fabrizio M. de Seta	Elsag Datamat		

Modification History

Issue	Date	Description
Draft A	25/03/2011	First issue for comments
V0.1	01/04/2011	Draft version
V1.0	08/04/2011	First release
V2.0	12/04/2011	Second release after a first round of internal review
Final	15/04/2011	Final version – official release

Contents

Figures

Tables

Acronyms

ESD	Embedded System Device
ESs	Embedded Systems
L-ESD	Legacy Embedded System Device
MS	Middleware Service
MwA	Middleware Adapter
NC	Node Capability
NoA	Node Adapter
NS	Network Service
NwA	Network Adapter
pS-ESD	pSHIELD Embedded System Device
pS-MS	pSHIELD Middleware Service
pS-OS	pSHIELD Overlay Service
pS-P	pSHIELD Subsystem
pS-P	pSHIELD Proxy
pS-SPD-ESD	SPD Embedded System Device
SPD	Security Privacy Dependability
CC	Common Criteria
FUA	Faults with Unauthorized Access
HMF	Human-Made Faults
NFUA	Not Faults with Unauthorized Access
NHMF	Nonhuman-Made Faults
SoC	System on Chip

Applicable Documents

Ref	Document Title	Issue/Date	Responsible
TA	pSHIELD Technical Annex	1	ALL
M0.2	SPD Metrics Aggregation	1	ED
D2.1.1	pSHIELD Systems requirements specifications	1	ASTS
D2.2.1	pSHIELD metrics definition	Draft	ESI

- This page intentionally left blank -

1 Executive Summary

The pSHIELD key concepts, enabling technologies and their formalised conceptual models are described as a fundamental **pSHIELD framework** within the Security, Privacy and Dependability (SPD) design constraints for future secure and fault-tolerant networked Embedded Systems Devices (ESDs). The next generation ESDs that are designed for many industrial applications should satisfy the requirements for availability, confidentiality, integrity, reliability, survivability, maintainability, safety, etc., and bring significant advantages into the ever growing competitive Embedded System world market.
The **overall objective** of the pSHIELD project is to design and implement a modular and composable framework able to provide enriched Security, Privacy and Dependability (SPD) within the context of heterogeneous Embedded Systems. The aim is to achieve optimised SPD performance within the boundaries of the requirements given by the considered application scenarios. In addition, the pSHIELD project has the ambitious long-term objective of enabling and defining the SPD certification for the future systems developed in compliance with the pSHIELD requirements.
The pSHIELD consortium proposes to achieve the above-mentioned overall objective through the development of **six innovative key concepts**;

1- A **Multi-layered approach** allowing the definition, for each layer, of the pSHIELD-specific SPD functionalities necessary to be installed, in a legacy Embedded System Device (ESD), so that it becomes "pSHIELD-compliant" ESD and can take advantage of the features introduced by pSHIELD;

2- The introduction of a pSHIELD-specific layer referred to as an **Overlay**, able, on the basis of the monitoring of SPD related multi-layer information, to take consistent and coordinated SPD decisions impacting on all the system;

3- A so-called **Seamless approach** able to overcome the intrinsic heterogeneity of Embedded Systems;

4- **Composability**, intended as the possibility to compose different (even heterogeneous) SPD functionalities both at design time as well as at runtime, so that the resulting system achieves a target SPD level;

5- The introduction of **innovative SPD functionalities** able to enrich the SPD effectiveness of the legacy Embedded System Devices enhancing their capabilities as well as making them composable;

6- The definition of common shared, measurable **SPD Metrics** fundamental to assess (and to certificate) the achieved SPD enhancements, for comparing the actual system SPD level with the target one.

In order to realise the above-mentioned six key concepts, the pSHIELD consortium will make use of **eight "key enabling technologies"** (KETs)**,** that are fundamental building blocks for developing targeted node, network, middleware and overlay layers of the pSHIELD framework. These KETs are: 1) the pSHIELD node, 2) cryptographic algorithms, 3) smart SPD driven transmission, 4) semantic models, 5) core SPD services, 6) hybrid automata, 7) common criteria approach and 8) the pSHIELD demonstrator.

Since pSHIELD is a pilot project, it will only design and implement a subset of the pSHIELD framework functionalities necessary to realise the above-mentioned six key concepts. Nevertheless, this subset of functionalities will be carefully selected in order to provide a preliminary, but exhaustive demonstration of the effectiveness of these six concepts. To enable the consortium to validate the overall objective, a Railway scenario (including hazardous material monitoring) has been selected. The pSHIELD demonstrator aims to achieve the pSHIELD overall objective through a proof of concept development of the above six key concepts.

In this document, after a short introduction within Section 2, the functional architecture of the pSHIELD framework is presented in Section 3. Section 4 details how the six key concepts are achievable by the pSHIELD functional architecture. The six key concepts in question are the subject of the Sections 4.1-4.6, respectively. For each key concept a description and the formalised conceptual model is presented. Section 5 highlights the major enabling technologies and methodologies investigated in pSHIELD to improve the state of the art and to achieve and demonstrate the overall project objective.

To better understand the document structure, the synoptic Figure 1.1 shows how starting from the pSHIELD functional architecture (Section 3) can be derived the six key concepts (from Section 4.1 to 4.6) which can be achieved and demonstrated through the development of eight enabling technologies (from Section 5.1 to 5.8).

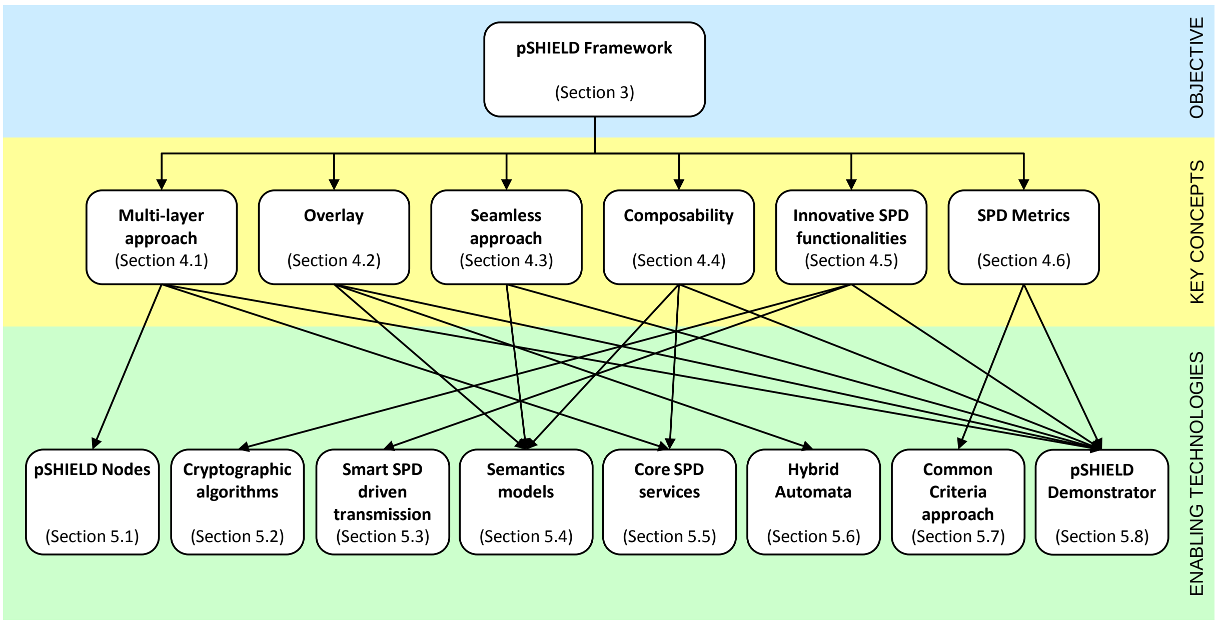

OBJECTIVE
KEY CONCEPTS
ENABLING TECHNOLOGIES
pSHIELD Framework (Section 3)
Multi-layer approach (Section 4.1)
Overlay (Section 4.2)
Seamless approach (Section 4.3)
Composability (Section 4.4)
Innovative SPD functionalities (Section 4.5)
SPD Metrics (Section 4.6)
pSHIELD Nodes (Section 5.1)
Cryptographic algorithms (Section 5.2)
Smart SPD driven transmission (Section 5.3)
Semantics models (Section 5.4)
Core SPD services (Section 5.5)
Hybrid Automata (Section 5.6)
Common Criteria approach (Section 5.7)
pSHIELD Demonstrator (Section 5.8)

Figure 1.1 pSHIELD synoptic representation

2 Introduction

The complex network of Embedded System (ES) on which society relies today are highly vulnerable. This vulnerability comes from a large number of failures and interference that can be intentional or non-intentional. Nowadays, it is required that the service delivered by an embedded system should be both dependable and secure and this is especially true for the rail transportation scenario selected [01] - [05] . This document presents a pSHIELD framework for integrated SPD assessment of ESs with emphasis on; the SPD topics [06] - [15] , pSHIELD related projects mentioned within the Technical Annex (TA) [16] - [21] , conceptual models [22] - [30] , certification and verification of ESs and services [32] - [37] , node layer [38] - [44] , network layer [45] - [49] , middleware layer [50] - [54] , overlay/cross-layer [55] - [65] , semantic ontology [66] - [73] , composability [74] - [81] , interoperability of heterogeneous networks [82] - [85] , and smart objects [86] - [91] .

In order to achieve the pSHIELD overall objective, the following present limitations of ESs should be overcome:

(i) the **lack of composable SPD Metrics** which prevent measurements to be taken in an univocal way of the SPD level of the Embedded System Devices (ESDs) included in an Embedded System, as well as of the Embedded System as a whole. In addition, the lack of such SPD Metrics precludes the univocal definition of the target SPD level of a given scenario;

(ii) the **lack of appropriate SPD functionalities** due to their complexity for critical applications that are not well optimised, i.e., lack of appropriate conceptual models for the selected industrial application scenario;

(iii) the fact that, even though different SPD functionalities provide satisfactory performance when considered separately, when they are integrated (i.e. composed) together in the Embedded System, due to the **lack of intelligent coordination of the composed SPD functionalities**, they do not succeed in achieving the expected SPD level. As a matter of fact, even when such coordination is attempted, the different SPD functionalities are composed together on the basis of the application of empirical, static, open-loop rules. The result of this is that guaranteeing a target SPD level for a certain Embedded System is not possible;

(iv) the lack of **logical formalism and semantic-based requirement engineering** needed to overcome the intrinsic Embedded System complexity and heterogeneity. This results in a high number of incomparable, inhomogeneous and incompatible SPD attributes that are required to be integrated together, making it an extremely challenging task to measure and certificate the achieved SPD level of the composed system.

In order to successfully overcome the above-mentioned limitations, the pSHIELD project proposes the following ESs solutions (each corresponding to the limitation with the same number):

(i) introducing **appropriate innovative SPD metrics**; this will be fundamental even for enabling and easing the SPD certification for all the future systems developed in compliance with the pSHIELD framework specification (this is just another important objective of the pSHIELD project);

(ii) Introducing **innovative SPD functionalities;** which can be transparently installed within legacy ESDs, according to a modular approach based on the pSHIELD multi-layer architecture;

(iii) introducing **innovative multi-layer functionalities;** including Control Algorithms which, on the basis of monitored multi-layer SPD-related parameters and by adopting advanced close-loop methodologies, take consistent decisions concerning the rules for selecting the SPD functionalities to be composed, as well as for their configuration and composition. This means that the various SPD functionalities are composed together on the basis of the outcome of a closed-loop reasoning accounting for dynamically measured parameters characterising the available SPD functionalities. This approach aims at guaranteeing optimised SPD performance for a defined Embedded System, thus satisfying the SPD requirements of the targeted scenario;

(iv) Introducing an abstract, **comprehensive semantic model** that provides a homogeneous representation of heterogeneous SPD-related parameters and conceptual models. This allows to dominate the complexity and the heterogeneity of the Embedded Systems abstracting from their peculiarities and focusing only on their abstract SPD semantic. In other words, the heterogeneous SPD functionalities are seen by the Control Algorithms as if they were a Seamless homogeneous pool.

With a clear focus on the current ESs limitations to guarantee SPD, it should be evident that, in order to overcome these limitations according to the above innovative system solutions, the pSHIELD project must focus on the development of six driving key concepts, namely the ones underlined in the previous discussion: pSHIELD multi-layered approach, Overlay, Seamless approach, Composability, innovative SPD functionalities, and SPD Metrics. For each key concept a formalised conceptual model must be selected to achieve the overall objective. The remainder of this document expands upon these six key concepts, as well as on the enabling technologies allowing the realisation of the concepts in question.

3 From objectives to functional architecture

The pSHIELD **overall objective** is to *design and develop a modular, composable framework able to provide enriched Security, Privacy and Dependability in the context of heterogeneous Embedded Systems as "built-in" rather than as "add-on functionalities", with the ambitious long-term objective of enabling and easing the SPD certification for all the future systems developed in compliance with the pSHIELD reference guidelines.*

In order to realize the pSHIELD overall objective, it is necessary to answer the five key questions hereinafter reported together with the corresponding five answers proposed by the pSHIELD consortium. These answers introduce the six key pSHIELD driving concepts whose actual realization allows reaching the pSHIELD overall objective. These six key concepts are described, following a consistent approach, from Section 4.1 to Section 4.6; the enabling technologies for implementing such six concepts are described in Section 5.

a) <u>How is it possible to achieve modularity?</u>

Modularity will be achieved by means of a **multi-layer approach** (see Section 4.1): Embedded System context can be partitioned into three different horizontal, technology-dependent layers (Node, Network and Middleware) according to its technological foundations (hardware elements, network capabilities and software functionalities). While the horizontal layers operate independently of one another, an additional pSHIELD-specific vertical, technology-independent layer, namely the **Overlay** (see Section 4.2) interoperates with the horizontal layers aiming at inter-layer SPD optimization. By so doing, each horizontal layer is decoupled from the others, so that the specific SPD modules/functionalities can be more easily added, removed, updated and controlled, whilst all inter-layer functionalities are included in the Overlay.

b) <u>How is it possible to address heterogeneity?</u>

Heterogeneity of the three layers' technologies will be addressed by adopting a **seamless approach** (see Section 4.3): the technology-dependent SPD functionalities are abstracted in order to achieve a technological-independent framework; this task will be performed by extensively using semantic modelling techniques. By following this approach, an uniform semantically-enriched ontological representation of carefully selected SPD functionalities is achieved, which is fundamental in order to flexibly handle and control the considered Embedded System (e.g., as detailed later, Composability entails the composition of heterogeneous SPD components and can be relatively easily achieved just thanks to the decoupling of the SPD components from the specific underlying technologies).

c) <u>How is it possible to provide enriched Security, Privacy and Dependability?</u>

Enriched Security, Privacy and Dependability will be provided by the pSHIELD framework according to a two steps approach. At first, **innovative SPD functionalities** (see Section 4.5) will be developed to enhance legacy device functionalities with the most powerful SPD technologies; then, these SPD enhanced SPD functionalities (also referred to as SPD components) will be *consistently* integrated (i.e. composed) by means of the **Composability** capability (see Section 4.4). So, Composability will allow state of the art and/or innovative SPD components to be composed together to provide enhanced, measurable SPD performance: the goal is to reach an overall system SPD levels matching with the target SPD levels deduced on the basis of the SPD requirements of the considered scenario. In other words, Composability aims at guaranteeing a desired SPD levels by selecting, configuring and composing already built-in SPD functionalities, adapting them with add-on further pSHIELD SPD features.
In this respect, it is important remarking that the possibility to univocally measure the achieved SPD levels (thanks to the introduced SPD Metrics), as well as the possibility to guarantee a target SPD level for the whole Embedded System open the way to new or enhanced applications and services, namely the ones requiring a guaranteed SPD level.

d) <u>How is it possible to perform Composability and why Composability is so important?</u>

The **Composability** of heterogeneous SPD technologies is realized by the interoperation between the Overlay vertical layer and the three horizontal layers (Middleware, Network and Node). In particular, the pSHIELD-specific functionalities should (i) dynamically monitor properly selected SPD-relevant

parameters and measurements of the SPD functionalities (thus achieving, by using ad-hoc SPD Metrics, a proper description of the SPD components) lying at the three horizontal layers, (ii) aggregate the monitored information in an SPD context-aware fashion, (iii) elaborate, in the Overlay, the aggregated information according to specific policies, in order to take consistent SPD related decisions concerning which SPD components have to be composed and the related configuration and composition rules, (iv) enforce these decisions back into the selected SPD components of the three horizontal layers.
It is important remarking that all intelligent control tasks related to composability are included in the Overlay which takes its SPD composition decisions on the basis of a very "rich" information, consisting of dynamic, semantically enriched, multi-layer, aggregated SPD-related metadata expressed using a common, formal, technology-independent language (this is possible thanks to the seamless approach mentioned above). Thanks to such rich input information and provided that such information is properly exploited by using appropriate reasoning methodologies, the Overlay can take *consistent* SPD composition decisions. In such a way, it will be possible to go well beyond the present state of the art in which composition of different SPD functionalities is either not possible at all, or performed according to empirical, off-line, open-loop, uncoordinated policies. Conversely, the Overlay can take automatic, closed-loop (i.e. cognitive), consistent decisions aiming at allowing the whole Embedded System to reach a predetermined SPD target level (thus entailing that the SPD requirements of a considered scenario can be satisfied).

e) *How is it possible to quantify SPD so as to favour certification?*

In order to measure the SPD levels, as well as to guarantee a target SPD level, it is necessary to formalize a way to identify, quantify and measure Security, Privacy and Dependability factors. In this respect, specific **SPD Metrics** (see Section 4.6) will be provided, which will be the first step towards the ambitious long-term pSHIELD overall objective of enabling and easing the SPD certification for all the future embedded systems.

Figure 3.1 shows the preliminary high level reference architecture which should support the actual realization of the six key pSHIELD concepts, namely the ones highlighted in bold when answering to the previous questions.

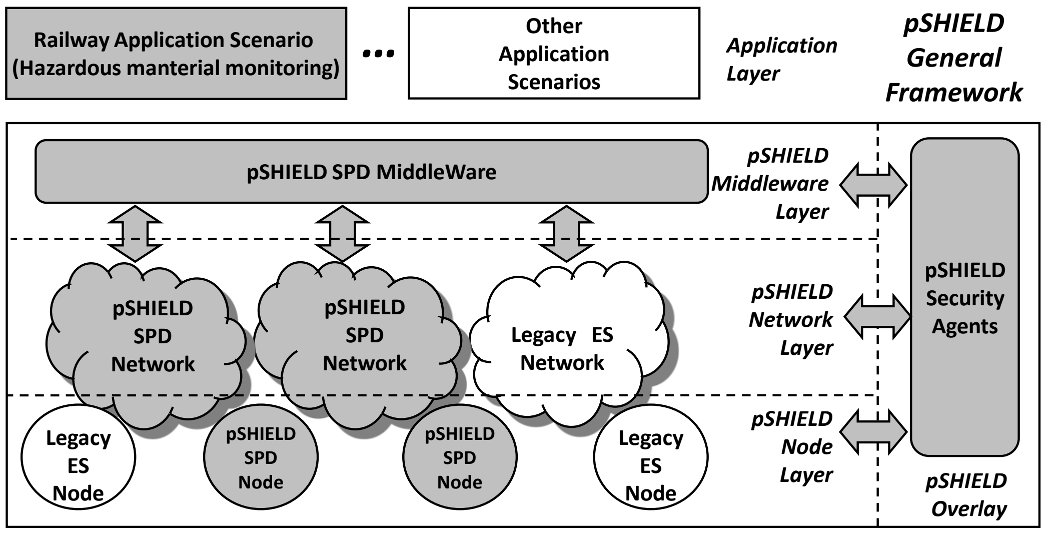


Figure 3.1 pSHIELD preliminary high level reference architecture

In Figure 3.1 the following elements can be easily identified:

- the legacy components at Node, Network and Middleware level;
- the pSHIELD entities at Node, Network and Middleware level which are obtained by enriching with pSHIELD-specific functionalities the legacy entities;
- the pSHIELD Overlay vertical layer;
- the application scenarios that ask the system for target SPD levels.

Since pSHIELD is a pilot project, it will only design and implement a reduced subset of elements and a reduced subset of functionalities embedded in these elements. Nevertheless, these elements and functionalities will be carefully selected in order to provide a preliminary, but exhaustive demonstration of the effectiveness of the above-mentioned six concepts. Design and implementation of the complete set of elements and functionalities will be developed in the scope of nSHIELD[1] project.

In particular, pSHIELD will address the gray elements shown in Figure 3.1, namely:

- a small, but meaningful set of pSHIELD entities at Node, Network and Middleware level;
- the Overlay vertical layer;
- a single application scenario, namely the railway one.

In the next section the preliminary high level reference architecture shown in Fig. 4.1 is further detailed into a logical/functional architecture that constitutes the starting point for the formal definition of a pSHIELD system able to realize the six driving concepts.

3.1.1 pSHIELD functional architecture

During the very first phase of the pSHIELD project, the above mentioned six concepts, as well as the preliminary high level reference architecture of Figure 3.1, have been considered as input for the identification of the functional architecture that supports the pSHIELD framework. In order to formally describe the pSHIELD framework, we decided to use the UML component diagram formalism, where there could be identified the following formal elements:

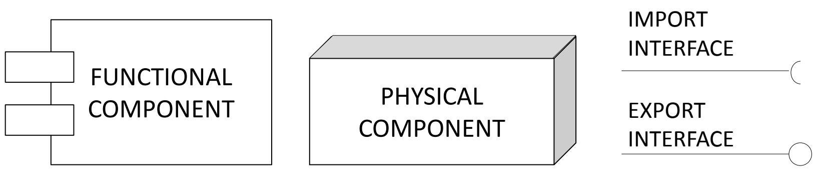


Figure 3.2 pSHIELD functional architecture formalism

A *functional component* describes a functional entity that, in general, does not have necessarily a physical counterpart (e.g. a software functionality, a middleware service, an abstract object, etc.). A *physical component* describes an entity that can be mapped into a physical object (e.g. a hardware component). A functional component as well as a physical component, can require or provide functionalities, properties, connections, services, configuration parameters, energy, etc. To model these aspects, we use the *interface* symbol. An interface can be used to import all the needed elements to make a component to properly work. An interface can also be used to export some elements as outcome of the component operations or physical structure.

From the pSHIELD perspective, there exist three different types of Embedded System Devices (ESDs):

Legacy Embedded System Device (L-ESD): it represents an individual, atomic physical Embedded System device characterized by legacy Node, Network and Middleware functionalities. This device can be modelled as depicted in Figure 3.3. So, the legacy functionalities of an L-ESD can be partitioned into three subsets:

- Node layer functionalities: hardware functionalities such as processors, memory, battery, I/O interfaces, peripherals, etc.
- Network layer functionalities: communication functionalities such as connectivity, protocols stack, etc.
- Middleware layer functionalities: firmware and software functionalities such as services, functionalities, interfaces, protocols, etc.

The L-ESD exposes three interfaces: (i) the legacy, technology-dependent middleware services, (ii) the legacy, technology-dependent network services and (iii) the legacy, technology-dependent node capabilities.

[1] ARTEMIS Call 2010 - Currently under negotiation

pSHIELD Embedded System Device (pS-ESD): it is a L-ESD equipped at least with the minimal set of pSHIELD functionalities at Middleware Layer. This device can be modelled as depicted in Figure 3.4: The pS-ESD exposes the same functionalities as the L-ESD plus an additional interface: the pSHIELD Middleware layer services.

pSHIELD SPD Embedded System Device (pS-SPD-ESD): it is a pS-ESD equipped at least with the minimal set of pSHIELD Overlay functionalities. This device can be modelled as depicted in Figure 3.5.
The pS-SPD-ESD exposes the same functionalities as the pS-ESD plus an additional interface: the pSHIELD Overlay layer SPD services provided by a so-called *Service Agent* operating in that ESD.

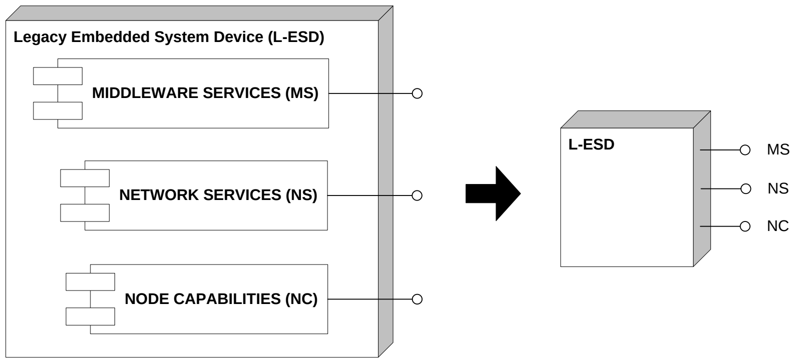


Figure 3.3 Legacy Embedded System Device (L-ESD) with its exposed functionalities

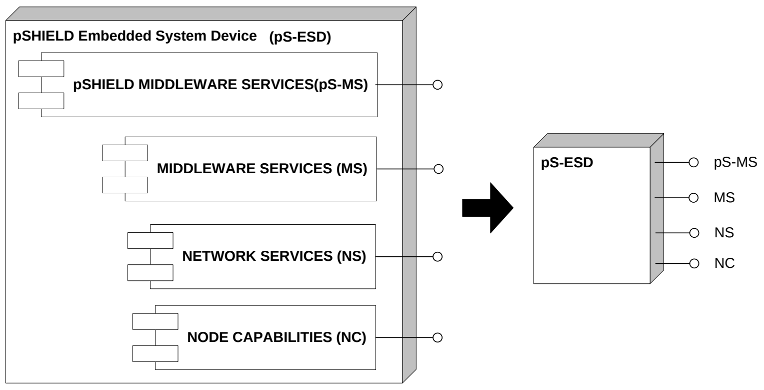


Figure 3.4 pSHIELD Embedded System Device (pS-ESD) with its exposed functionalities

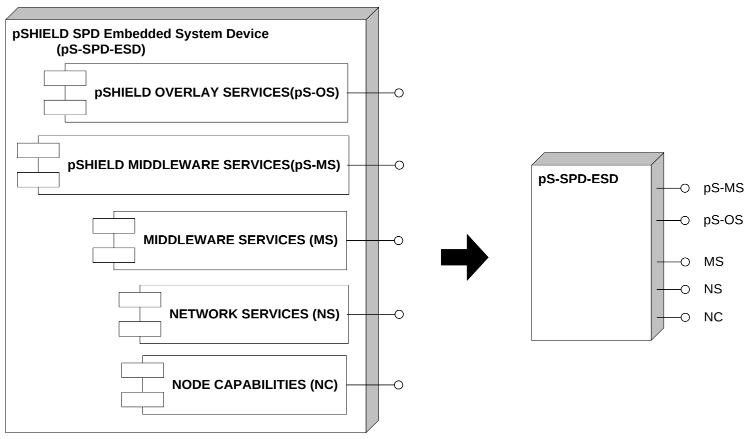

pSHIELD SPD Embedded System Device (pS-SPD-ESD)
pSHIELD OVERLAY SERVICES(pS-OS)
pSHIELD MIDDLEWARE SERVICES(pS-MS)
MIDDLEWARE SERVICES (MS)
NETWORK SERVICES (NS)
NODE CAPABILITIES (NC)
pS-SPD-ESD
pS-MS
pS-OS
MS
NS
NC

Figure 3.5 pSHIELD SPD Embedded System Device (pS-SPD-ESD)

We can define the architecture of a pSHIELD Subsystem (pS-S) as a set of Embedded System Devices including several L-ESD, connected to several pS-ESD and one and only one pS-SPD-ESD. Connections between two generic ESDs (L-ESD, pS-ESD or pS-SPD-ESD) can be performed, by means of legacy functionalities at Node, Network and/or Middleware layer, through the so-called NC, NS and MS functionalities, respectively. This means, for example, that the legacy Node layer capabilities (e.g. legacy physical connectors) of an L-ESD can be used to connect it to a pS-SPD-ESD having a compatible Node layer capability (e.g. the same connector format factor). The connection of a pS-ESD with another pS-ESD or with a pS-SPD-ESD can also be performed by exploiting the additional pSHIELD Middleware layer functionalities exposed by the pS-MS interface. Each pSHIELD Subsystem (pS-S) must have one and only one pS-SPD-ESD, so that a pS-SPD-ESD can be connected to another pS-SPD-ESD by means of so called pS-OS functionalities. A pSHIELD Subsystem is depicted in Figure 3.6.

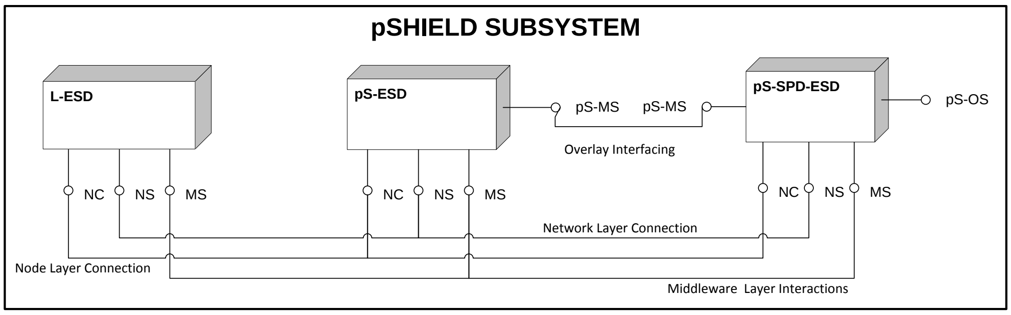


Figure 3.6 pSHIELD Subsystem architecture decomposed into ESD components

Let consider a pSHIELD system including several Embedded System Devices belonging to the three above defined types of ESD: L-ESD, pS-ESD and pS-SPD-ESD. The pSHIELD system architecture can be represented as a set of pSHIELD subsystems each connected to the other by means of the overlay interfaces provide by the Service Agent:

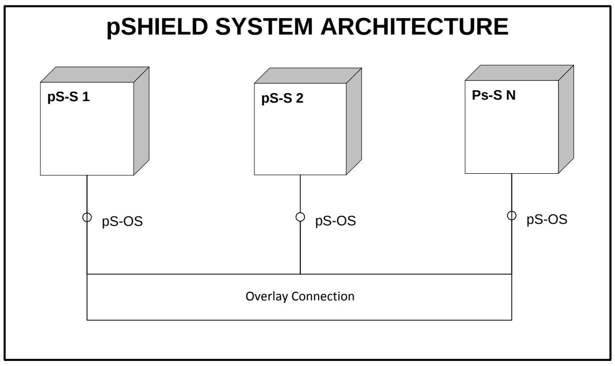


Figure 3.7 pSHIELD System Architecture decomposed into pSHIELD Subsystems

Figure 3.8 shows the pSHIELD System Architecture highlighting the various ESD types.

In particular, for the sake of clarity, only the pSHIELD Subsystem 1 has been exploded to explicitly the various ESD types. In particular, within the pSHIELD Subsystem 1, (i) the connections among the pS-ESD and the L-ESDs have been grouped in the “Legacy Middleware Network Node” cloud, (ii) the connections among the pS-ESDs and the pS-SPD-ESD have been grouped in the “pSHIELD Middleware” cloud, (iii) the connections among different pSHIELD Subsystem (i.e. between different pS-SPD-ESDs) have been grouped in the “Overlay” cloud.

The Overlay consists of a set of SPD Security Agents located each in a different pS-SPD-ESD. Each SPD Security Agent controls a different pSHIELD subsystem. Subsystem identification has to be carefully performed scenario by scenario. Expandability is obtained by enabling communication between SPD Security Agents taking care of different pSHIELD sub-systems.

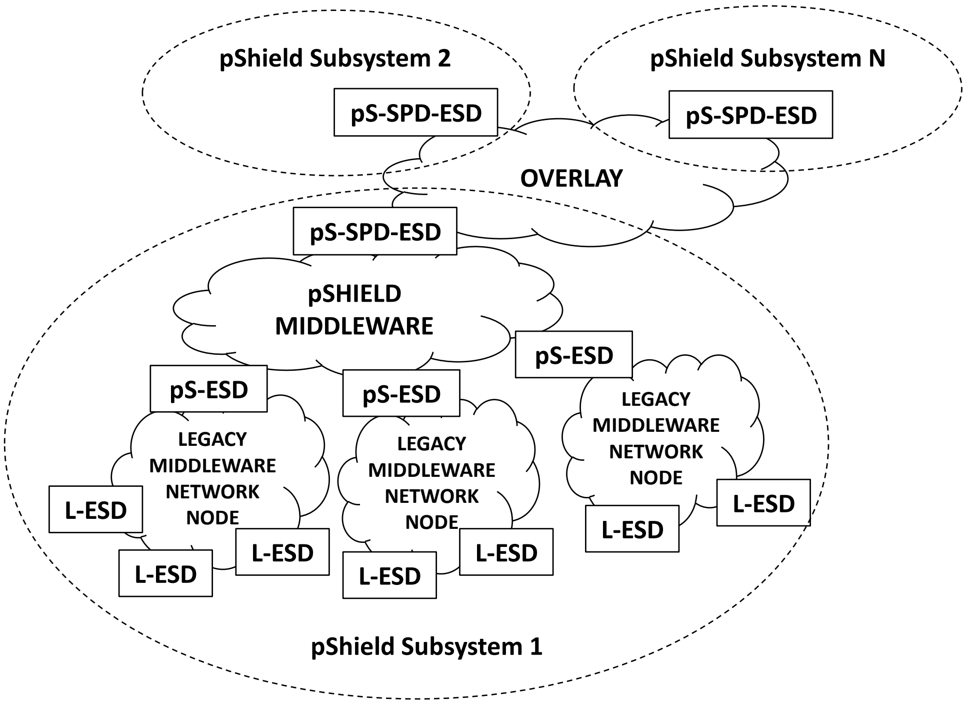


Figure 3.8 pSHIELD System Architecture highlighting the ESD types

In order to introduce the next sections, we should refine the ESD model. Exploding the model of a pS-SPD-ESD (Figure 3.9), it is possible to identify that its external functionalities are provided by the interaction of two components: the L-ESD and the pSHIELD Proxy. While we have already defined the L-ESD component, the pSHIELD Proxy is a new component.

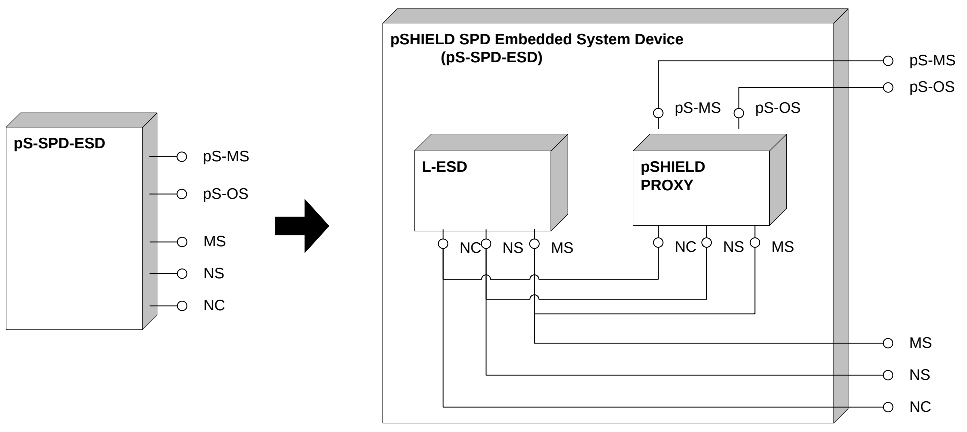


Figure 3.9 pS-SPD-ESD architecture

A ***pSHIELD Proxy*** (pS-P) is a technology dependent component of a pS-SPD-ESD that, interacting with the available legacy Node, Network and Middleware capabilities and functionalities (through the NC, NS and MS interfaces, respectively), provides all the needed pSHIELD enhanced SPD functionalities. From the above figure, it is clear that the pS-SPD-ESD functionalities belongs both to the L-ESD component (MS, NS and NC interfaces), as well as to the pSHIELD Proxy component (pS-MS and pS-OS interfaces).

The pSHIELD Proxy component can be further decomposed in two components (see Figure 3.10), namely the pSHIELD Adapter and the Security Agent. The rationale behind these two components, as well as their architecture, is hereinafter explained.

The ***pSHIELD Adapter*** is a technology dependent component in charge of interfacing with the legacy Node, Network and Middleware functionalities (through the MS, NS and NC interfaces). The legacy functionalities can be enhanced by the pSHIELD Adapter in order to make them *pSHIELD-compliant,* i.e. they become SDP *legacy device components*, which can be composed with other SPD components, according to the SPD Composability approach detailed in 4.4. In addition, the pSHIELD Adapter includes *Innovative SPD functionalities* which are SPD *pSHIELD-specific components,* which can be composed with other SPD components. The pSHIELD Adapter exposes the technology independent pSHIELD Middleware layer functionalities that are used by the Security Agent component.

The ***Security Agent*** is a technology-independent component in charge of aggregating the information coming from the pSHIELD Middleware Services provided by the internal pSHIELD Adapter or by other pSHIELD Proxies located in the same subsystem. The Security Agent is also in charge of gathering the information coming from other Security Agents connected on the same Overlay (through the pS-OS interface). The Security Agent includes proper control algorithms working on the basis of the available information; the decisions taken by these Control Algorithms are enforced through the pS-MS and the pS--OS interfaces.

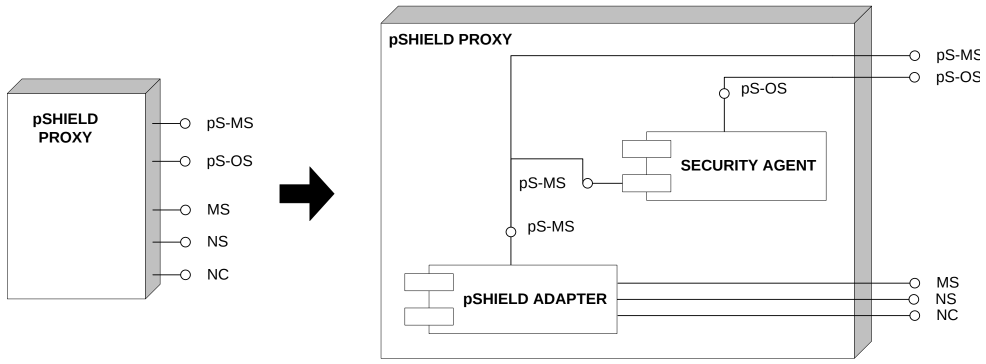


Figure 3.10 pSHIELD Proxy component architecture

The Security Agent component can be further decomposed in two components (see Figure 3.11), namely the Semantic Knowledge Representation and the Control Algorithms. The rationale behind these two components, as well as their architecture, is hereinafter explained.

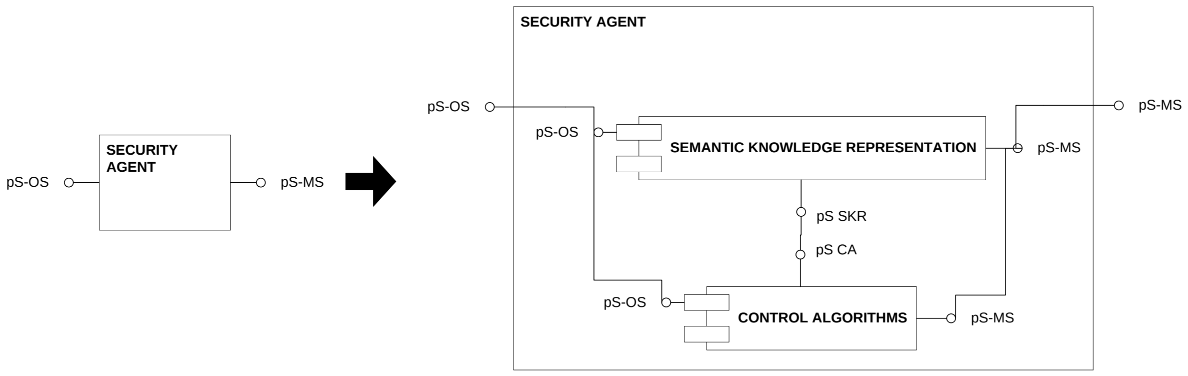


Figure 3.11 pSHIELD Security Agent component architecture

The ***Semantic Knowledge Representation*** is in charge of bi-directionally exchanging technology-independent (and semantic enriched) information from the pS-MS and the pS-OS interfaces. It is also in charge to provide such information via the pS-SKR interface to the Control Algorithms component.

The ***Control Algorithms*** retrieves the aggregated information on the current SPD status of the subsystem, as well as of the other interconnected subsystems, by the pS-CA interface connected to the Semantic Knowledge Representation; such retrieved information is used as input for the Control Algorithms. The outputs of the Control Algorithms consist in decisions to be enforced in the various ESDs included in the pSHIELD subsystem controlled by the Security Agent in question; these decisions are sent back via the pS-MS interface, as well as communicated to the other Security Agents on the Overlay, through the pS-OS interface.

The pSHIELD Adapter can be further decomposed as in three components (see Figure 3.12), namely the pSHIELD Node Adapter, the pSHIELD Network Adapter and the pSHIELD Middleware Adapter. The rationale behind these three components, as well as their architecture, is hereinafter explained.

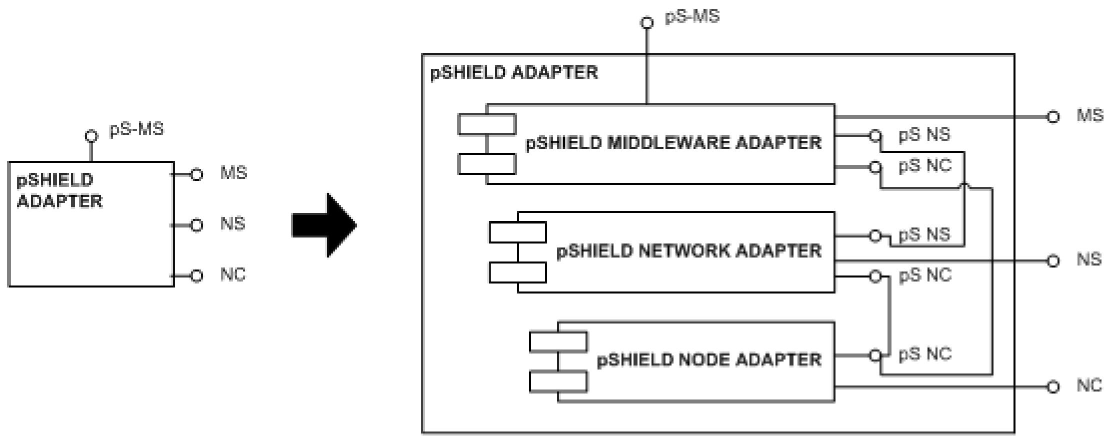


Figure 3.12 pSHIELD Adapter component architecture

- The ***pSHIELD Node Adapter*** includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD node capabilities (using the NC interface) in order to enhance them with the pSHIELD Node layer SPD enabling technologies (such as **FPGA Firmware** and **Lightweight Asymmetric Cryptography**). This adapter is in charge to provide (through the pS-NC interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Node layer legacy and Node pSHIELD-specific functionalities. Moreover, the pSHIELD Node Adapter translates the technology independent commands, configurations and decisions coming from the pS-NC interface into technology dependent ones and enforce them also to the legacy Node functionalities through the NC interface.

- The ***pSHIELD Network Adapter*** includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD network services (through the NS interface) and the pSHIELD Node Adapter (through the pS-NC interface) in order to enhance them with the pSHIELD Network layer SPD enabling technologies (such as **Smart Transmission**). This adapter is also in charge to provide (through the pS-NS interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Network layer legacy and Network pSHIELD-specific functionalities. Moreover, the pSHIELD Network Adapter translates the technology independent commands, configurations and decisions coming from the pS-NS interface into technology dependent ones and enforce them also to the legacy Network functionalities through the NS interface.

- The ***pSHIELD Middleware Adapter*** which can be further partitioned in the **Core SPD services** and in the **Innovative SPD functionalities.** These two components are linked through the pS-MS interface. The Figure 3.13 shows the pSHIELD Middleware Adapter component architecture.
  The *Core SPD Services* are in charge to discover all the available functionalities at Node, Network and Middleware layers and to describe them in a technology-independent fashion. All the information is sent to the Overlay through the pS-MS interface. The pSHIELD Middleware Adapter should also carry into operation the decisions taken by the Overlay and communicated via the pS-MS interface by actually composing the discovered SPD functionalities. The pSHIELD Middleware Adapter includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD middleware services (through the MS interface) in order to make them discoverable and composable SPD functionalities.

Figure 3.14 shows the pS-ESD architecture. As expected, a pS-ESD is an enhanced L-ESD with some additional pSHIELD capabilities: the ones provided by the pSHIELD Adapter component. However, the pS-ESD is not equipped with a Security Agent (otherwise it would have become a pS-SPD-ESD).

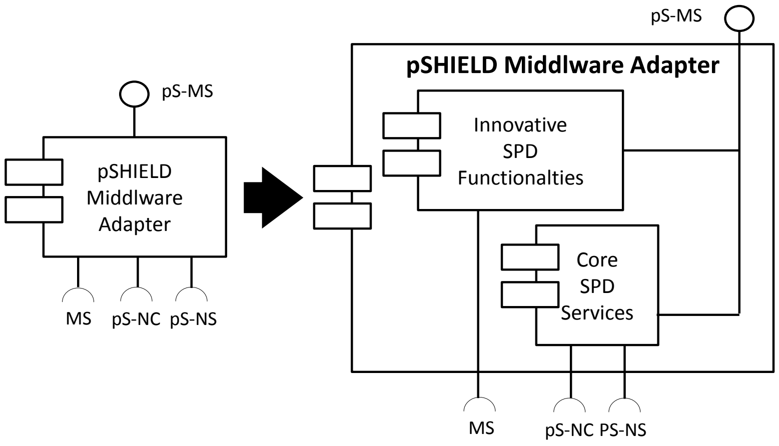

pS-MS
pSHIELD Middlware Adapter
MS
pS-NC
pS-NS
pS-MS
pSHIELD Middlware Adapter
Innovative SPD Functionalties
Core SPD Services
MS
pS-NC
PS-NS

Figure 3.13 pSHIELD Middleware Adapter component architecture

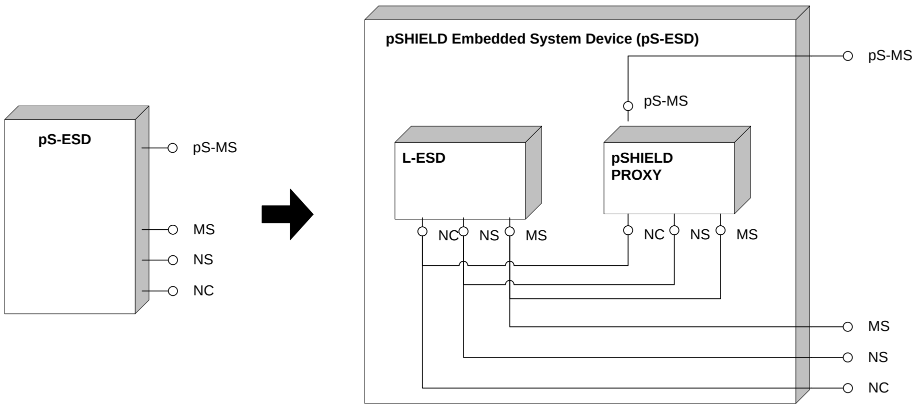


Figure 3.14 pS-ESD component architecture

Finally the Legacy Embedded System Device (L-ESD) can be further exploded into a more detailed component diagram, as shown in Figure 3.15.

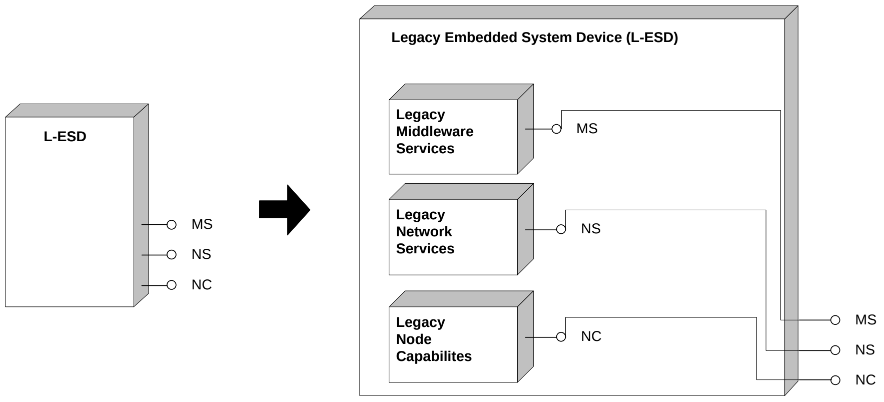


Figure 3.15 L-ESD component architecture

The L-ESD architecture is composed by three components: (i) the Legacy Middleware Services; (ii) the Legacy Network Services and (iii) the Legacy Node Capabilities.

The **Legacy Middleware Services** includes all the legacy middleware services (i.e. messaging, remote procedure calls, objects/content requests, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant. In order to be pSHIELD compliant, these services should be enriched with pSHIELD SPD functionalities. This task is in charge of the pSHIELD Middleware Adapter.

The **Legacy Network Services** includes all the legacy network services (protocol stacks, routing, scheduling, Quality of Service, admission control, traffic shaping, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant. In order to be pSHIELD compliant, these services should be enriched with pSHIELD SPD functionalities. This task is in charge of the pSHIELD Network Adapter;

The **Legacy Node Capabilities** component includes all the legacy node capabilities (i.e. battery, CPU, memory, I/O ports, IRQ, etc.) provided by the Legacy Embedded System Device which are not pSHIELD compliant. In order to be pSHIELD-compliant, these capabilities should be enhanced and enriched with pSHIELD SPD functionalities. This task is in charge of the pSHIELD Node Adapter;

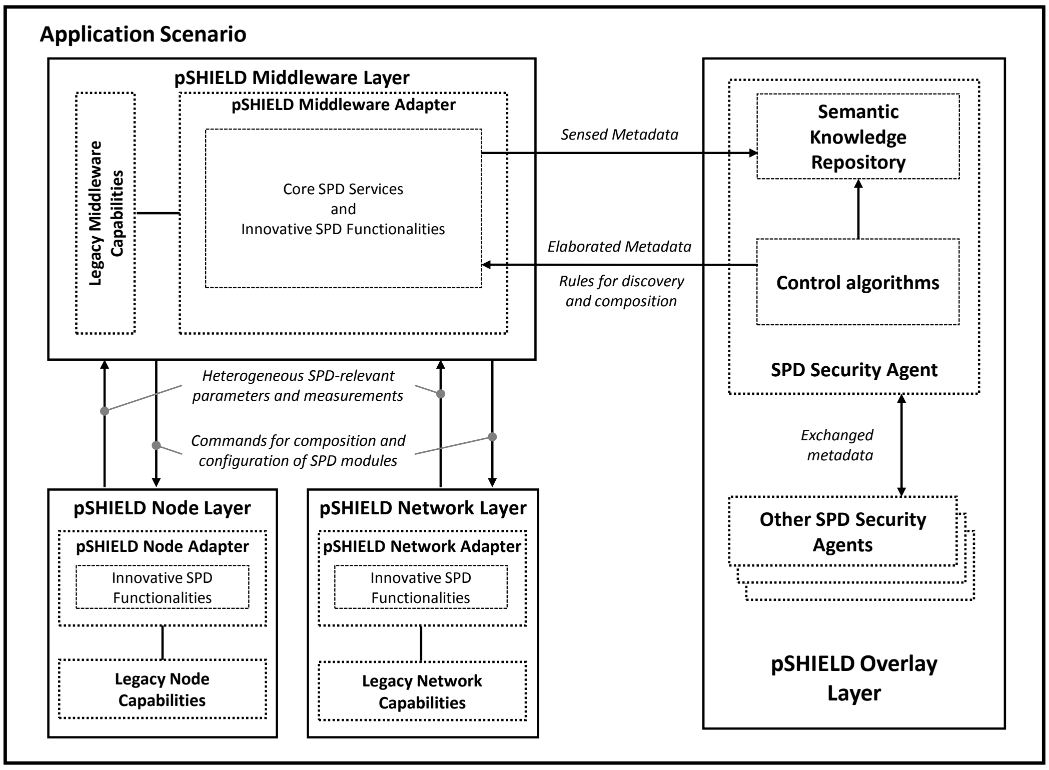

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 3.16 pSHIELD functional component architecture

The result of the above considerations is the pSHIELD functional architecture depicted in Figure 3.16. This figure highlights some of the key pSHIELD enabling technologies mapped into the four functional layers introduced in the previous section. More specifically:

1. The pSHIELD Node that is described in Section 5.1;
2. The Innovative SPD functionalities (e.g. smart SPD driven transmission and Cryptographic algorithms) that are described in Sections 5.3 and 5.2;
3. The semantic knowledge models that are described in Section 5.4;
4. The core SPD services that are described in Section 5.5;

5. The control algorithms based on Hybrid Automata theory which is described in section 5.6;

In the following, the six key concepts introduced in the previous section will be described just with reference to the functional architecture shown in the above figure. In particular, in the next six sections, the rationale behind the functional blocks (and relevant interfaces), appearing in this section, which are relevant to the corresponding concept, will be explained.

4 pSHIELD key concepts

4.1 pSHIELD multi-layered approach

4.1.1 Description

The following figure highlights in light red the partitioning of the Embedded System context into four layers (Node, Network, Middleware and the vertical Overlay).

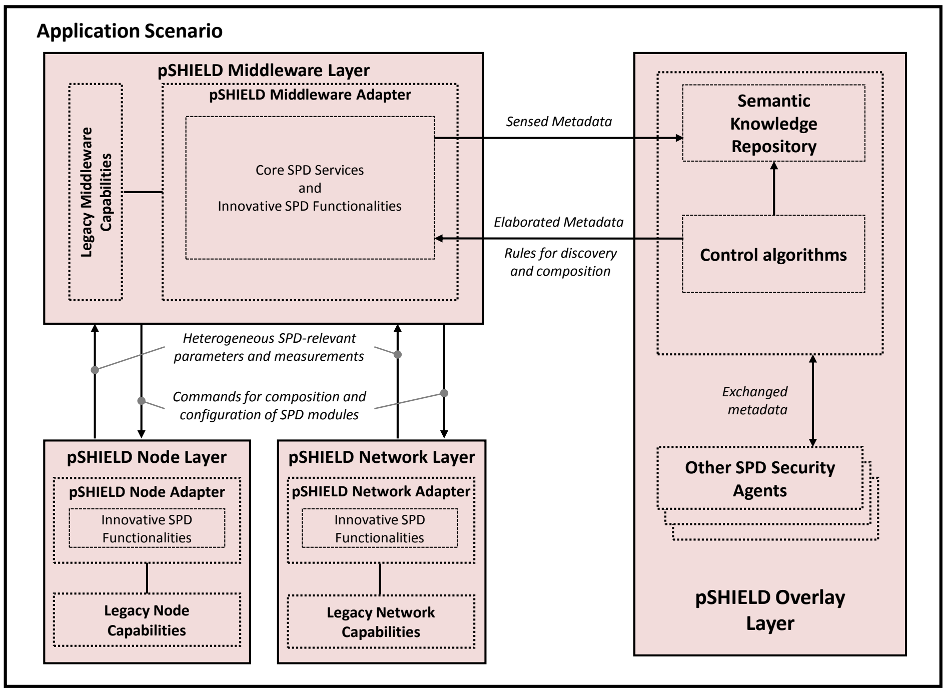

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.1 pSHIELD multi-layered approach: a functional view

The pSHIELD multi-layered approach considers the partition of a given Embedded System into three technology-dependent horizontal layers: the node layer (meaning the hardware functionalities), the network layer (meaning the communication functionalities) and the middleware layer (meaning the software functionalities). This classification is fully in line with the present context of industrial Embedded System suppliers, which are mainly organized in these three major sectors.

In addition, pSHIELD considers a fourth pSHIELD-specific technology-independent vertical layer, namely the so-called Overlay which takes strategic SPD related decisions, as detailed in the next section.

The pSHIELD Middleware layer includes the so-called Core SPD services which are in charge to mediate the information exchange between the Overlay vertical layer and the three horizontal layers.
In particular, the Core SPD Services are in charge of (i) discovering and of keeping updated the available SPD resources, services, functionalities, data and contextual information coming from the Node, Network and Middleware layer, (ii) translating this technology-dependent heterogeneous information into technology-independent metadata, (iii) filtering, semantically enriching and aggregating these metadata which will be eventually stored in the Semantic Knowledge Representation database of the pSHIELD Overlay (see next section). As detailed in the next section, the pSHIELD Overlay relies its decisions just on these last metadata.
Moreover, the Core SPD Services are also in charge of enforcing the Overlay decisions all over the three horizontal layers, i.e. from the Middleware layer down to the Node layer, passing through the Network layer.

Thus, while the intelligent control functionalities of the pSHIELD architecture are located in the Overlay layer, the sensing and actuation functionalities are located in the Core SPD services of the pSHIELD Middleware layer. The Core SPD Services perform such functionalities in a transparent way, i.e. without the need of modifying the technologies underlying the legacy Embedded System Devices. In other words, the the pSHIELD Middleware represents a sort of convergence layer by which the Overlay can transparently access and control all layers (Node, Network and Middleware) of the underlying heterogeneous technologies.

4.1.2 Formalized conceptual model

Figure 4.2 shows the layered architecture of a pSHIELD SPD Embedded System Device highlighting the pSHIELD multi-layered approach.

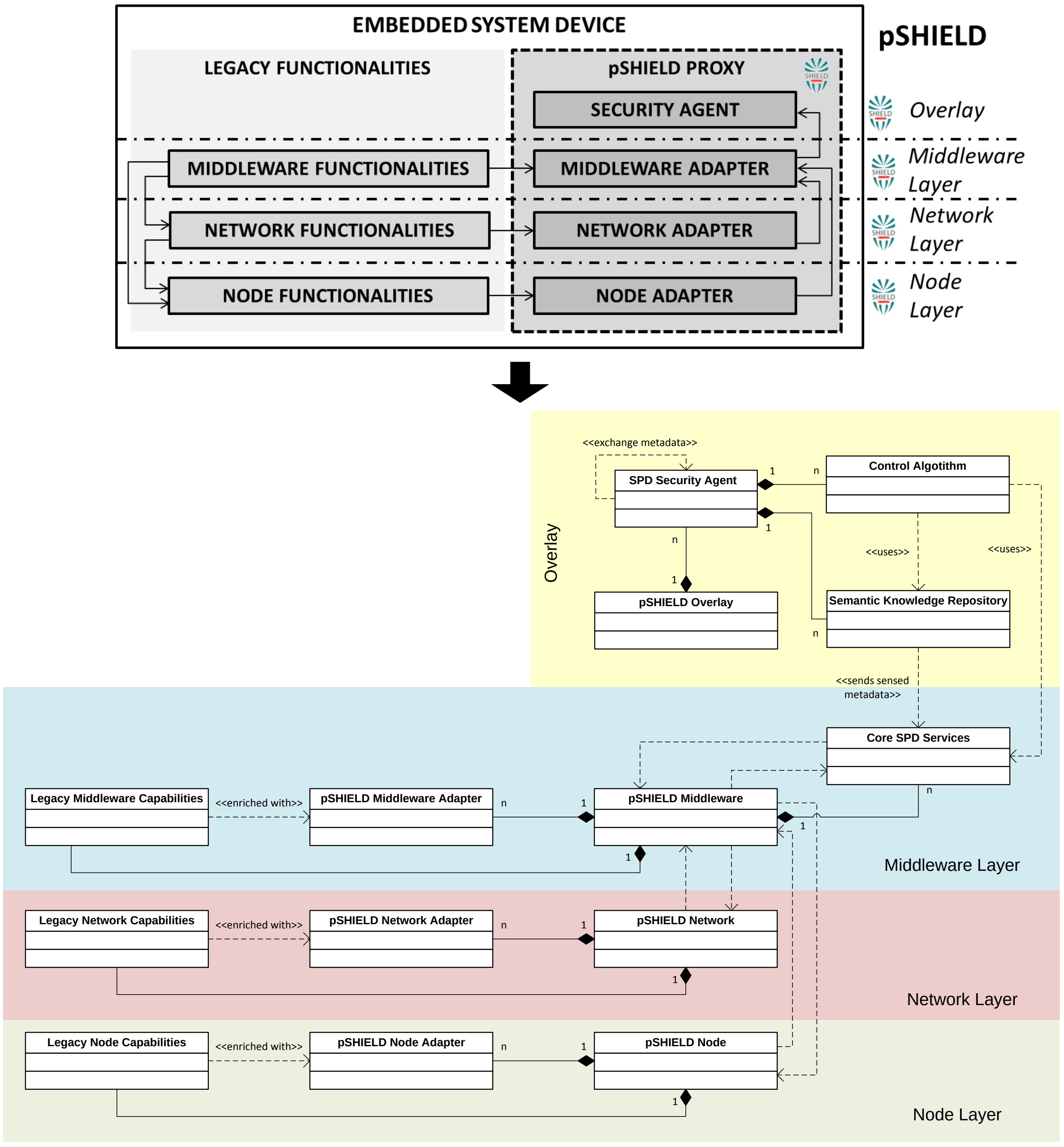


Figure 4.2 ESD plus pS-P layering architecture

The key conceptual element enabling the pSHIELD multi-layered approach is the pSHIELD proxy. It hosts all the pSHIELD functionalities divided into the four pSHIELD layers: (i) the pSHIELD Security Agent acts at Overlay; (ii) the pSHIELD Middleware Adapter acts at Middleware Layer; (iii) the pSHIELD Network Adapter acts at Network Layer and (iv) the pSHIELD Node Adapter acts at Node Layer.

Is it worth noting that it is not mandatory for a pSHIELD Proxy instance to include all the Adapters (Node, Network, Middleware) and the Security Agent (SA); likewise, in a given Adapter or SA not all the possible pSHIELD-specific functionalities have to be present. Conversely, in principle, any combination of the Adapters/SA installation and any subset of functionalities within each installed Adapters/SA is possible; the most appropriate combination should be selected case-by-case depending on the requirements of the considered scenario. Nevertheless, a given pSHIELD Adapter will be referred to as *pSHIELD-compliant* if, in such an ESD, has been installed at least a pSHIELD Middleware Adapter with the Core SPD services. For instance, a resource limited pSHIELD ESD, as a sensor device, can be equipped with an instance of pSHIELD Proxy that only includes the pSHIELD Middleware Adapter with the basic Core SPD Services. The other way around, a powerful embedded device, can be equipped with an instance of pSHIELD proxy that provides all the pSHIELD Adapter/SA functionalities at all layers (Node, Network, Middleware and Overlay).

Obviously, whenever the Node, Network or Middleware Adapter functionalities are not present in an ESD, the associated ESD legacy functionalities (e.g. expressed in terms of resources, information, data, services) cannot be controlled by the pSHIELD Overlay and hence cannot be discovered, configured, composed by the pSHIELD Middleware.

4.2 Overlay

4.2.1 Description

The following figure highlights in light red some key functionalities and interfaces involving the Overlay layer.

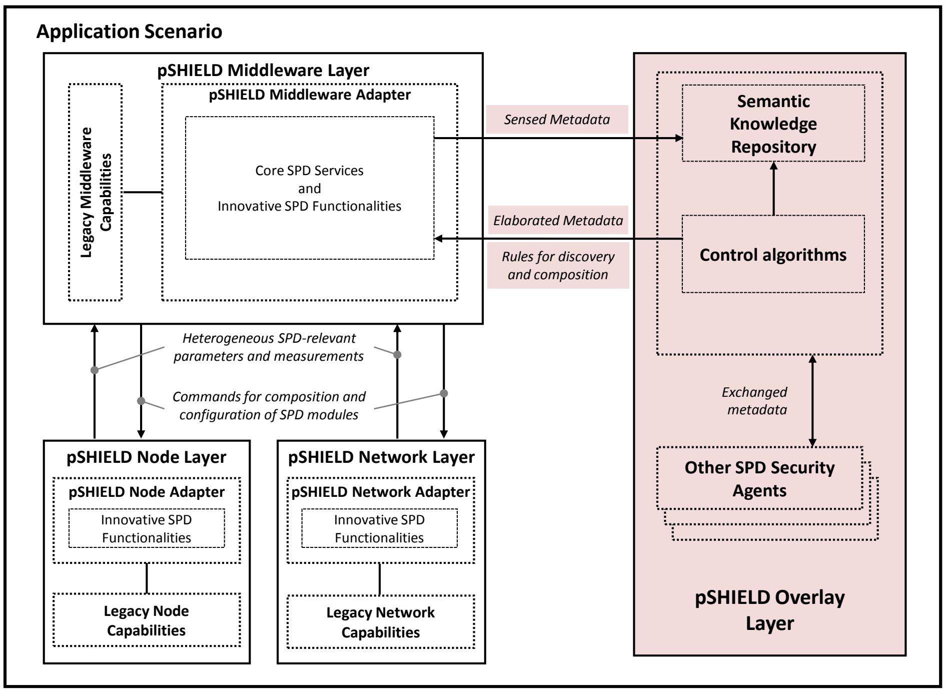

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.3 pSHIELD overlay: a functional view

The Overlay consists of a set of SPD Security Agents, each one controlling a given pSHIELD subsystem. Subsystem identification has to be carefully performed scenario by scenario. Expandability of such framework is obtained by enabling communication between SPD Security Agents controlling different sub-systems. As a matter of fact, the presence of more than one SPD Security Agent is justified by the need of solving scalability issues in the scope of system-of-systems: exponential growth of complexity can be overcome only by adopting the policy of *divide et impera* (divide and rules).

Each SPD Security Agent, in order to perform its work, exchange carefully selected information with the other SPD Security Agents, as well as with the three horizontal layers (node, network and middleware) of the controlled pSHIELD subsystem.

Each SPD Security Agent collects properly selected heterogeneous SPD-relevant measurements and parameters coming from node, network and middleware layers of the controlled pSHIELD subsystem; this information is used as valuable input for the Control Algorithms. Since the SPD Security Agent is mainly a software functionality and not a physical system itself, it needs the mediation of the pSHIELD Middleware Core SPD Services as it has been explained in the previous section.

The heterogeneous data collected from the three horizontal layers are abstracted, translated into technology-independent metadata, semantically enriched, aggregated, as detailed in Section 4.3. The resulting metadata (referred to as *sensed metadata*) are stored in the Semantic Knowledge Representation of the considered SPD Security Agent. By so doing, this database stores a dynamic, semantic representation of the controlled pSHIELD subsystem.

In the considered SPD Security Agent, the representation mentioned above is used as a valuable, rich input for a set of intelligent, technology-independent, closed-loop Control Algorithms. These last, by using (as input) the above-mentioned representation and by adopting appropriate advanced methodologies able to profitably exploit such input, produce (as output) decisions aiming at guaranteeing, whenever it is possible, target SPD levels over the controlled pSHIELD subsystem. In the Composability feature described in the section 4.4, these decisions consist in a set of rules for discovery, configuration and composition of SPD components.

The decisions mentioned above are eventually actuated in the pSHIELD subsystem controlled by the considered SPD Security Agent, by exploiting the mediation of the middleware as it will be detailed in the next section.

Summarizing, each SPD Security Agent consists of two key elements:

(i) the Semantic Knowledge Repository (i.e. a database) storing the dynamic, semantic, enriched, ontological aggregated representation of the SPD functionalities of the pSHIELD subsystem controlled by the SPD Security Agent;
(ii) the Control Algorithms generating, on the grounds of the above representation, key SPD-relevant decisions (consisting, as far as the Composability feature is concerned, in a set of discovery, configuration and composition rules).

4.2.2 Formalized conceptual model

The formalized conceptual model of a pSHIELD Security Agent, described in the previous section, is reported in Figure 4.4.

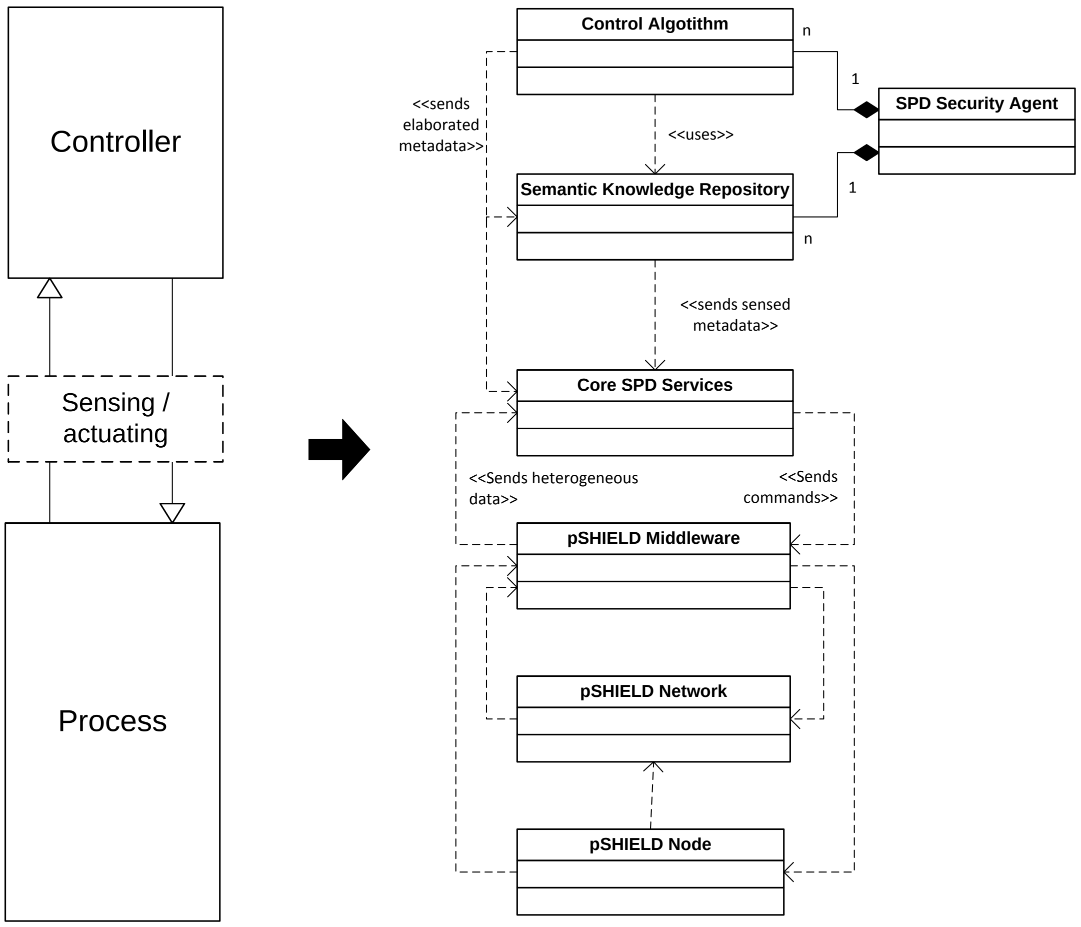


Figure 4.4 Formalized conceptual model for pSHIELD Security Agent

Figure 4.4 also highlights the correspondence between the class diagram and a classical feedback control scheme (including Process, Controller and Sensing/Actuation functionalities) where:

- the Process to be controlled is represented by the three horizontal layers (Node, Network and Middleware);
- the Controller is the Security Agent supported by the Semantic Knowledge Repository;
- Sensors/Actuators are represented by the Core SPD Services lying at the pSHIELD Middleware layer.

The formalized conceptual model of the pSHIELD Overlay is shown in Figure 4.5. Taking into account that the pSHIELD Overlay can include several Security Agents each one controlling a pSHIELD subsystem, the pSHIELD Overlay model is easily obtained by composing in parallel the feedback control scheme relevant to each Security Agent presented in the previous figure. The coordination and information exchange between the different controllers (i.e. SPD Security Agents) is performed by means of metadata (referred to as *exchanged metadata*). The advantages in terms of flexibility entailed by this modular approach are evident.

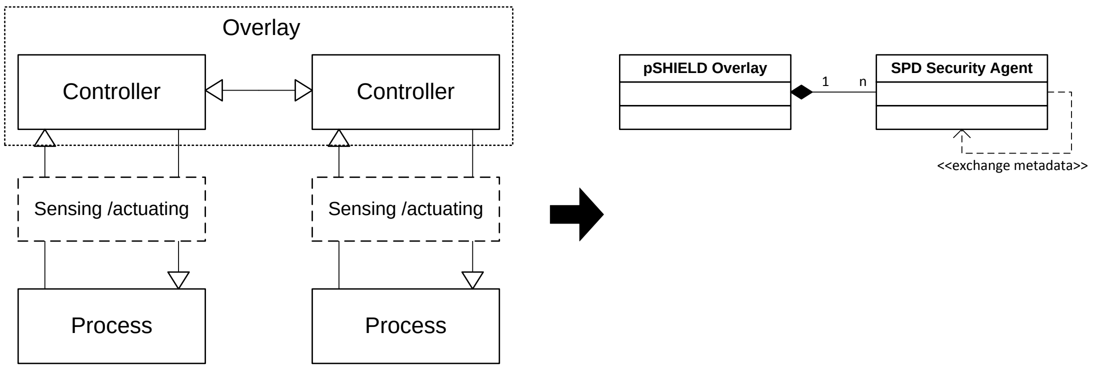


Figure 4.5 Formalized conceptual model for the pSHIELD Overlay

4.3 Seamless approach

4.3.1 Description

The following figure highlights in light red some key functionalities and interfaces involving the Seamless approach.

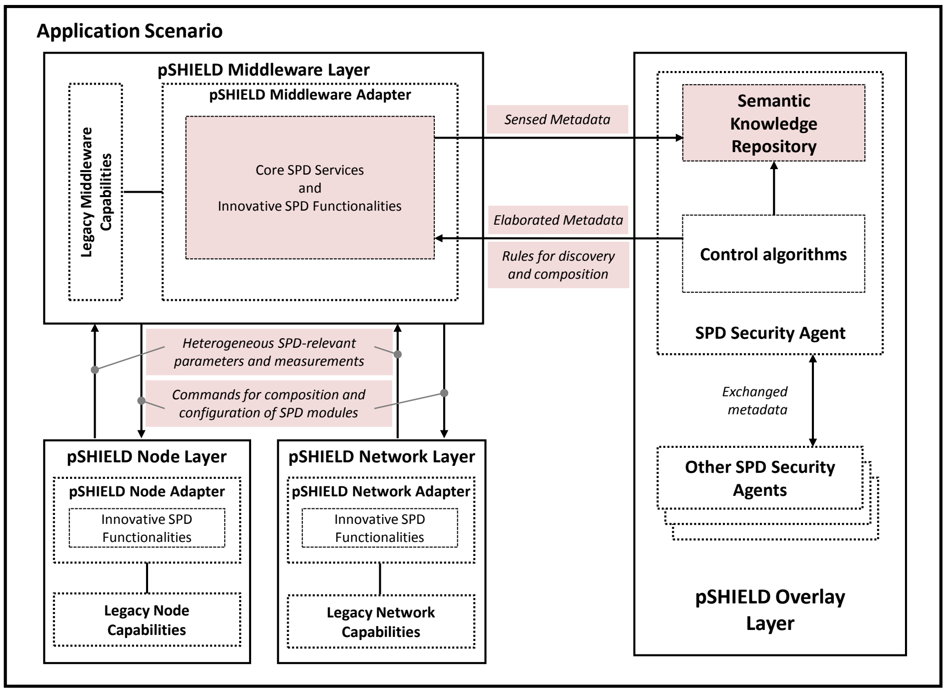

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.6 pSHIELD seamless approach: a functional view

The pSHIELD project aims at providing enriched SPD functionalities to *heterogeneous* Embedded Systems according to a common approach which leaves out of consideration the specificity of the underlying technologies.

The proposed solution for coping with heterogeneity, is to design and implement a proper ***semantic aware framework*** that performs *sensing/actuating functionalities*: the sensing functionalities have to provide for translating selected monitored SPD parameters and measurements from a technology-dependent world into a technology-independent one, whilst the actuating functionalities for translating technology-independent decisions into technology dependent commands to enable the composition of technology dependent SPD functionalities. Such a framework is based on a dynamic, homogeneous, semantically enriched description of heterogeneous SPD functionalities characterized by specific, quantifiable SPD Metrics. The semantic aware framework is realized by a set of Core SPD Services included in the pSHIELD Middleware Adapter. In particular, among the others, the following key Core SPD Services will be used: *Discovery, Composition, Orchestration and **Semantic Database**.*
As far as the sensing functionalities are concerned, they will be coordinated by the Orchestration service (in turn, controlled by the Overlay) which should be able to dynamically (i) gather and aggregate, in a SPD-aware fashion, different metadata coming from pSHIELD Adapters located in different layers of the same ESD (ii) monitor the status of the active SPD functionalities reasoning the *semantically enriched, aggregated, homogeneous metadata*, (iii) further aggregate, in a SPD-aware fashion, metadata even considering the metadata exchanged with other pSHIELD Middleware Adapters belonging to other ESDs. The scope of the Orchestration is to obtain *homogeneous, semantically enriched, aggregated metadata* (hereinafter, for the sake of brevity, simply referred to as ***sensed metadata***) very *rich* in SPD related information to be used by the pSHIELD Overlay. These sensed metadata are homogeneous and

technology-independent, i.e. the Overlay can deal with such semantic aggregated metadata ignoring the fact that they are coming from heterogeneous and technology dependent parameters and measurements.

In particular, the outcome of the aggregation performed by the Orchestration service located in the ESDs under a Security Agent control, are a set of sensed metadata that are stored in the **Semantic Knowledge Representation database** of the Security Agent controlling the considered pSHIELD subsystem. A Security Agent can also acquire sensed metadata from other Security Agent to have a more visibility and complete information on the surrounding environment. These metadata serve as very valuable inputs for the Control Algorithms present in the SPD Security Agent in question. So, such Control Algorithms "see" seamless metadata, even though they have been originally generated by heterogeneous SPD parameters and measurements associated to different ESDs. It is fundamental noting that the Control Algorithms in question, being fed with a very rich feedback information (in principle, multi-layer, multi-component, multi-subsystem) and provided that they adopt proper control methodologies (e.g. hybrid automata) able to actually exploit the valuable input information, can take *consistent* and *long-sight* decisions in configuring a proper composition of the available SPD functionalities in order to achieve the target SPD level over the whole system. In the Composability feature described in the next section these decisions consist in a set of rules for configure discovery and composition of SPD components.

As far as the actuating functionalities are concerned, they will be coordinated by the Discovery and Composition services (in turn, configured by the Overlay) which should be able to dynamically translate the elaborated metadata representing the technology-independent Control Algorithm decisions into actual actions consisting in: (i) dynamically discover, filter and select all the available SPD functionalities that match with the Overlay configuration rules (ii) apply the Overlay decisions composing the discovered SPD functionalities.

4.3.2 Formalized conceptual model

The formalized conceptual model for the seamless capability of the pSHIELD semantic ontological framework is shown in Figure 4.7 and is based on the concepts described in the previous section.

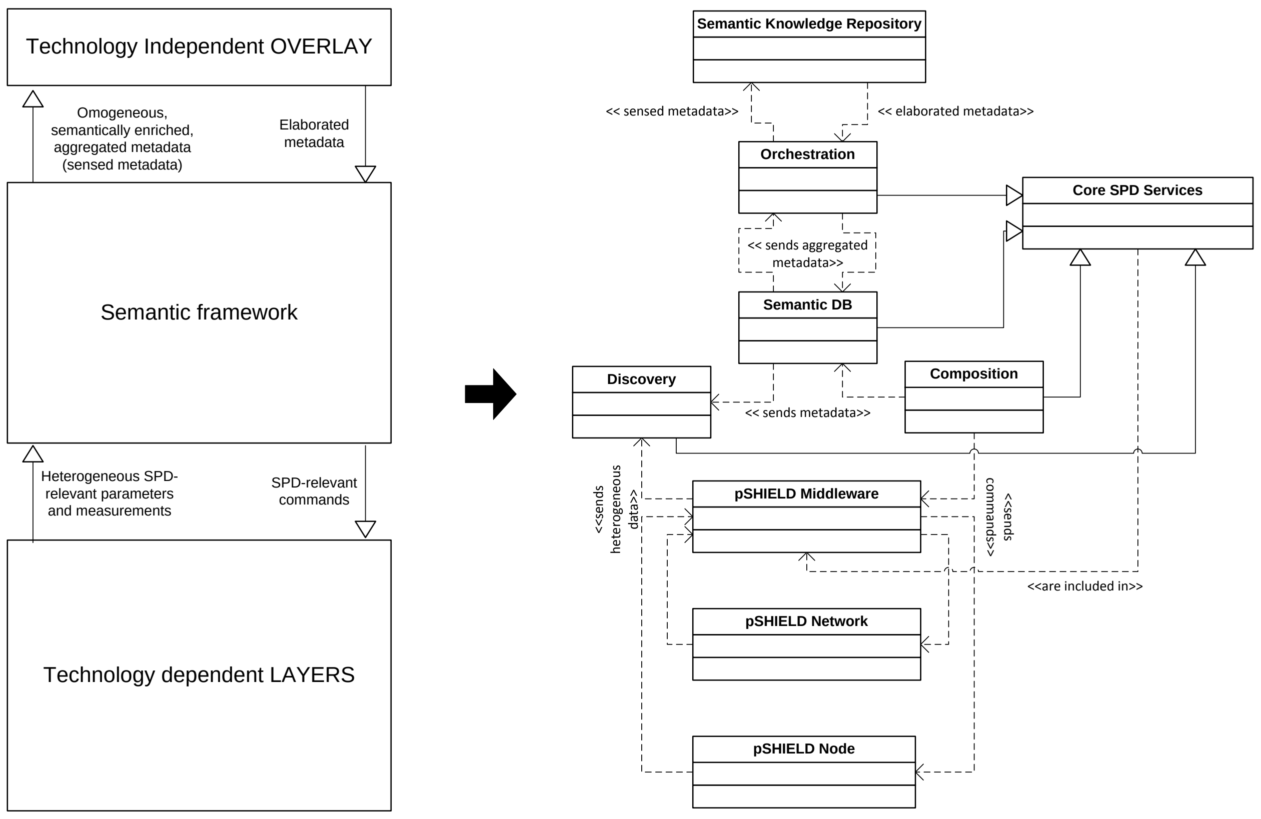

Technology Independent OVERLAY
Omogeneous, semantically enriched, aggregated metadata (sensed metadata)
Elaborated metadata
Semantic framework
Heterogeneous SPD-relevant parameters and measurements
SPD-relevant commands
Technology dependent LAYERS
Semantic Knowledge Repository
<< sensed metadata>>
<< elaborated metadata>>
Orchestration
Core SPD Services
<< sends aggregated metadata>>
Semantic DB
Discovery
Composition
<< sends metadata>>
<<sends heterogeneous data>>
<<sends commands>>
pSHIELD Middleware
<<are included in>>
pSHIELD Network
pSHIELD Node

Figure 4.7 Formalized conceptual model for pSHIELD seamless approach

4.4 Composability

4.4.1 Description

The following figure highlights in light red some key functionalities and interfaces involved in the Composability feature.

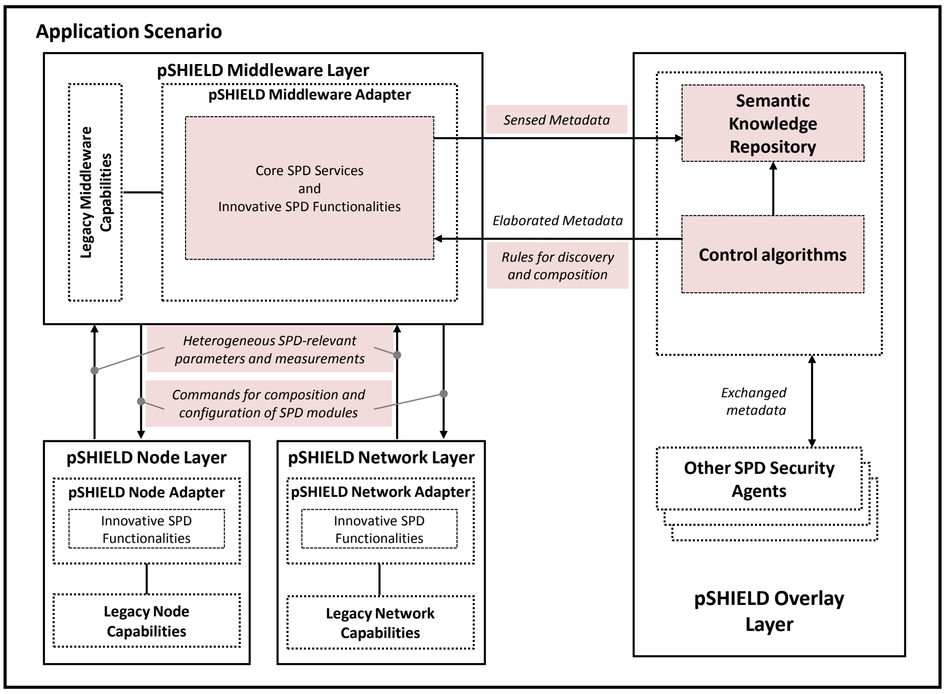

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.8 pSHIELD composability: a functional view

The main added value provided by the pSHIELD framework is the possibility to compose different (possibly heterogeneous) SPD functionalities (also referred to as *SPD components*) aiming at achieving in the considered system of Embedded System Devices a target SPD level which satisfies the requirements of the considered scenario.

A **SPD component** is defined as a functionality which (i) offers a given SPD service through an interface which can be semantically described according to the provided SPD Metrics, (ii) can be accessed through the pSHIELD Middleware Core SPD Services for being configured (if necessary) and activated (or deactivated).

Composition of SPD components entails the following three fundamental issues:

- Design and implementation of the *sensing functionalities* needed to identify the most appropriate sensed metadata to describe the SPD components.
- Design and implementation of the Control Algorithms which, on the basis of the sensed metadata above, i.e. on the basis of the ontological description (possibly semantically enriched) of the SPD components, have to provide the rules for discovery, configuration and composition of the SPD components.
- Design and implementation of the *actuating functionalities* needed to actuate the above rules.

As concerns the Control Algorithms, they include all the intelligence behind the Composability concept. As a matter of fact, these algorithms have to take decisions driving both the sensing and the actuation functionalities. These decisions consist in rules, carried into operation by the Core SPD Services, to configure the discovery, composition and orchestration of SPD components (and consequently of their functionalities and provided services). As an example, the configuration rules allow to properly select (i) which SPD components have to be monitored, (ii) the parameters and measurements related to these components which have to be monitored, (iii) the monitoring frequency, etc.. The rules for configuration of Core SPD Services allow to properly select (i) which SPD components have to be involved in the composition process, (ii) how the selected SPD components have to be configured and composed. In this last respect, it should be clear that the Control Algorithms are in charge for dynamically deciding the most appropriate SPD Composition Topology. Note that the Control Algorithms, in principle, thanks to the provided abstraction, have a complete freedom of interconnecting whichever SPD Component with whichever SPD Component which means that Control Algorithms adopt a networked (m:n) composition topology. Nevertheless the choice for the best composition model and the related supported composition topologies depends on the model implemented in the Control Algorithm. In this respect one of the pSHIELD enabling technologies (described in Section 5.6) investigates on the use of Hybrid Automata control theories.

As already outlined in the previous sections, the Control Algorithms decisions are taken by properly elaborating, according to appropriate advanced methodologies, the semantically enriched aggregated metadata stored in the Semantic Knowledge Representation database, i.e. the Control Algorithms can avail of a *rich* input information. So, a key issue for the effectiveness of the pSHIELD Composability will be the ability of the Control Algorithms in exploiting the above rich input information for producing SPD-relevant decisions aiming at driving the actual SPD level (computed according to proper SPD Metrics, as explained in Section 4.6) towards the target one (this last being determined on the grounds of the SPD requirements of the considered scenario).

As concerns the sensing and actuating functionalities, as already detailed in Section 4.3, they are implemented through the Core SPD Services lying in the pSHIELD Middleware. As it has been explained in Section 4.3, the Core SPD Services provide their sensing and actuation functionalities by adopting a layering composition topology. The Core SPD Services will be described and formalized in Section 5.5. Thus, in the following section, we concentrate on formalizing the pSHIELD SPD components, and their composition in the Overlay.

4.4.2 Formalized conceptual model

4.4.2.1 pSHIELD SPD component

Two kinds of pSHIELD SPD components can be identified and formalized:

(i) the **pSHIELD-specific components**, i.e. the innovative SPD functionalities ad hoc developed for the pSHIELD project which are included in the pSHIELD Adapters. They can be classified in *Node (see* Figure 4.9*), Network (see* Figure 4.10*) and Middleware (see* Figure 4.11*) pSHIELD-specific components* according to whether they are included in the pSHIELD Node, Network or Middleware Adapter. They can be directly accessed by pSHIELD Middleware Core SPD Services through the pS-NC, pS-NS and pS-MS interfaces.

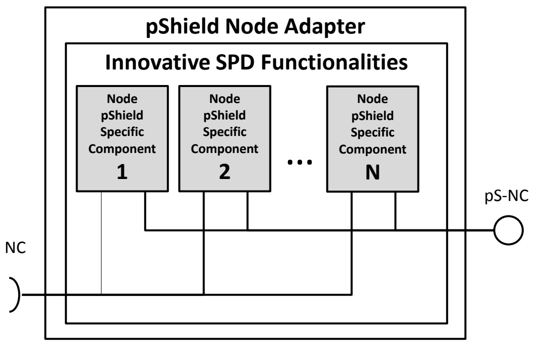

pShield Node Adapter
Innovative SPD Functionalities
Node pShield Specific Component 1
Node pShield Specific Component 2
...
Node pShield Specific Component N
pS-NC
NC

Figure 4.9 pSHIELD Specific Node Components related to the pSHIELD Node Adapter

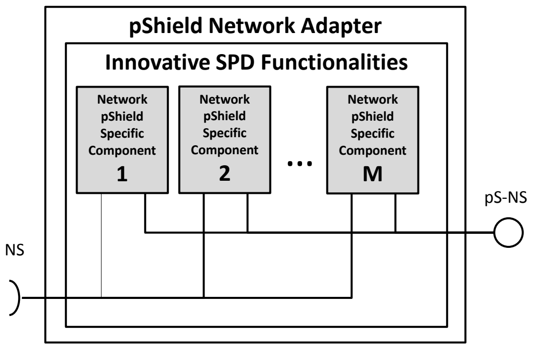

pShield Network Adapter
Innovative SPD Functionalities
Network pShield Specific Component 1
Network pShield Specific Component 2
...
Network pShield Specific Component M
pS-NS
NS

Figure 4.10 pSHIELD Specific Components related to the pSHIELD Network Adapter

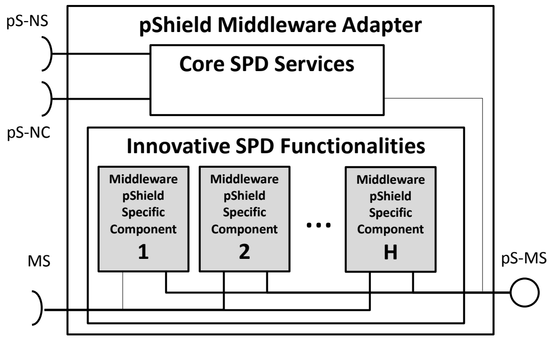

pS-NS
pShield Middleware Adapter
Core SPD Services
pS-NC
Innovative SPD Functionalities
Middleware pShield Specific Component 1
Middleware pShield Specific Component 2
...
Middleware pShield Specific Component H
MS
pS-MS

Figure 4.11 pSHIELD Specific Components related to the pSHIELD Node Adapter

(ii) the **legacy device components,** i.e. the SPD functionalities already present in the legacy devices which can be accessed through the pSHIELD Adapters; they can be classified in *Node (see* Figure 4.12*), Network (see* Figure 4.13*) and Middleware (see* Figure 4.14*) legacy device components* according to whether they are included in a legacy Node/Network/Middleware which can be accessed through the corresponding pSHIELD Adapter.

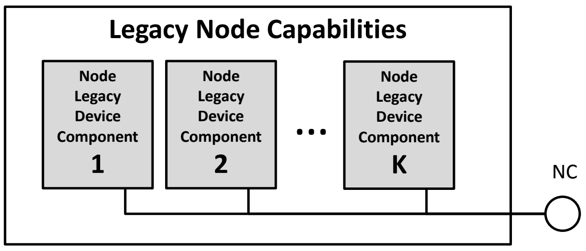

Legacy Node Capabilities
Node Legacy Device Component 1
Node Legacy Device Component 2
...
Node Legacy Device Component K
NC

Figure 4.12 Node Legacy Device Components representing the Legacy Node Capabilities

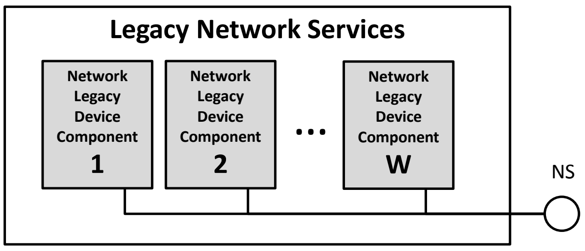

Legacy Network Services
Network Legacy Device Component 1
Network Legacy Device Component 2
...
Network Legacy Device Component W
NS

Figure 4.13 Network Legacy Device Components representing the Legacy Network Services

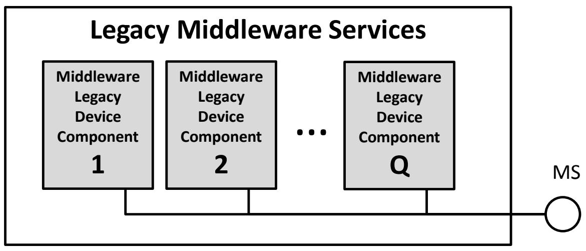

Legacy Middleware Services
Middleware Legacy Device Component 1
Middleware Legacy Device Component 2
Middleware Legacy Device Component Q
MS

Figure 4.14 Middleware Legacy Device Components representing the Legacy Middleware Services

The Legacy Device Components can be accessed by pSHIELD Middleware Core SPD Services through the interface provided by the pSHIELD Node, Network and Middleware Adapters. As seen in Figure 4.15 the Core SPD Services can directly control the pSHIELD specific SPD components (i.e. the Innovative SPD functionalities) as well as indirectly control the Middleware, Network and Node Legacy Device components through, respectively, the pSHIELD Middleware, Network and Node Adapters.)

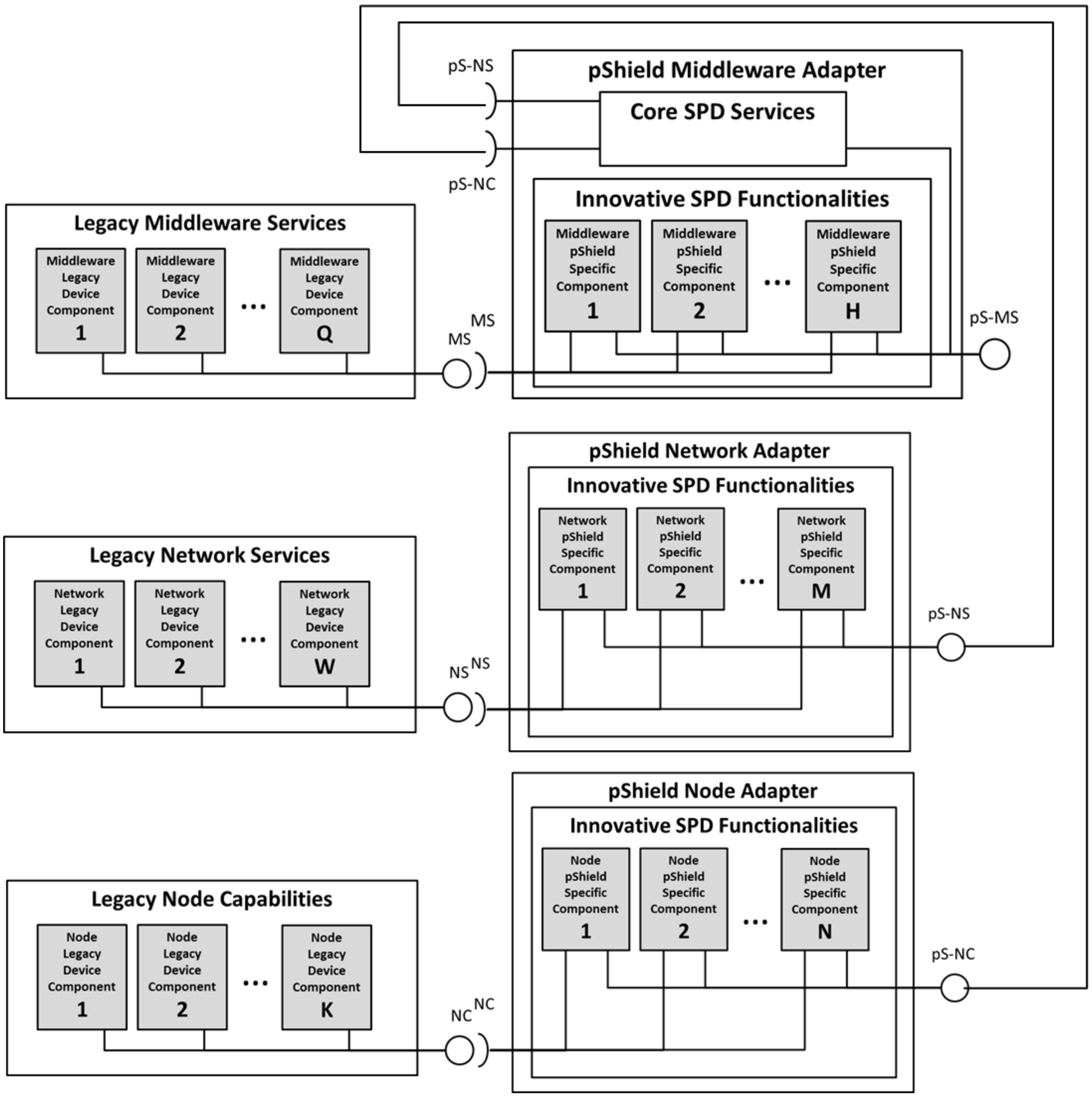

pShield Middleware Adapter
Core SPD Services
pS-NS
pS-NC
Innovative SPD Functionalities
Middleware pShield Specific Component 1
Middleware pShield Specific Component 2
Middleware pShield Specific Component H
pS-MS
Legacy Middleware Services
Middleware Legacy Device Component 1
Middleware Legacy Device Component 2
Middleware Legacy Device Component Q
MS
MS
pShield Network Adapter
Innovative SPD Functionalities
Network pShield Specific Component 1
Network pShield Specific Component 2
Network pShield Specific Component M
pS-NS
Legacy Network Services
Network Legacy Device Component 1
Network Legacy Device Component 2
Network Legacy Device Component W
NS
NS
pShield Node Adapter
Innovative SPD Functionalities
Node pShield Specific Component 1
Node pShield Specific Component 2
Node pShield Specific Component N
pS-NC
Legacy Node Capabilities
Node Legacy Device Component 1
Node Legacy Device Component 2
Node Legacy Device Component K
NC
NC

Figure 4.15 pSHIELD components' interactions

On the basis of the pSHIELD SPD component description provided above, a Middleware, Network or Node pSHIELD-specific component can be modelled as components requiring, respectively, the MS, NS and NC functionalities, while providing, respectively, the pS-MS, pS-NS and pS-NC functionalities (see Figure 4.16):

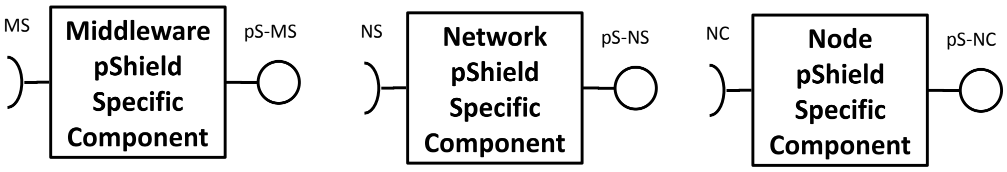

MS
Middleware pShield Specific Component
pS-MS
NS
Network pShield Specific Component
pS-NS
NC
Node pShield Specific Component
pS-NC

Figure 4.16 Middleware, Network and Node pSHIELD Specific Components

It is worth to note that the MS, NS and NC interfaces are used by the Middleware, Network or Node pSHIELD-specific Component to exploit those technology dependent functionalities provided by the Middleware, Network and Node Legacy Device Components. In turn, the Middleware, Network and Node pSHIELD-specific Components provide their technology independent, pSHIELD-compliant functionalities via the pS-MS, pS-NS and pS-NC interfaces. In the light of the above consideration, for the purpose of the pSHIELD project, only the pS-MS, pS-NS and pS-NC interfaces can be formally specified, because are technology independent.

As explained in the previous section, pSHIELD considers the Composability of SPD functionalities (i.e. Middleware, Network and Node pSHIELD Specific Components) both in the Overlay and in the pSHIELD Middleware Layer: (i) in the Overlay proper Control Algorithms (residing in the pSHIELD Security Agent) guarantee a reference, desired SPD level of the whole system identifying the best configuration rules to compose the available SPD functionalities, (ii) in the pSHIELD Middleware the Core SPD services continuously monitor the SPD functionalities status and actuate the decisions taken by the Control Algorithms applying the configuration rules composing the available SPD functionalities provided by the pSHIELD Node, Network and Middleware Adapters.

A key role in the whole Composition process described above is played by the Middleware, Network and Node pSHIELD-specific Components interfaces: pS-MS, pS-NS and pS-NC. The formalized conceptual model of these interfaces is the same, thus let focus on the pS-XX interface, where XX means MS or NS or NC, without loss of generality.

A pS-XX interface is constituted by the following elements:

1. **function**: it represents the functionality provided by the corresponding pSHIELD-specific Component. It is defined by:
    a. **name**: it represents the name of the functionality used to identify it.
    b. **input parameters**: it represents the list of (optional) input parameters needed to make the function properly work.
    c. **output parameters**: it represents the list of (optional) output parameters provided by the function.
2. **contract**: it is the contract associated to the function. It can be used by the Core SPD services to properly compose the available services. It is optional and can be used to define the function pre-conditions, post-conditions, side effects, exceptions, acceptable ranges, etc.;
3. **description**: it is a semantic description of the interface, containing the necessary information to make this function discoverable;
4. **SPD status**: it represents the current SPD level associated to the function.
5. **connector**: it represents the means to compose together two functionalities

The specification of function, contract and description syntax and semantic has been provided in pSHIELD, applying a semantic design of the pSHIELD domain. The aim is to model a proper ontology from which the specification (in terms of formal syntax and semantic) of each pS-MS, pS-NS and pS-NC interface can be derived. This approach is described in Section 5.4. The use of a semantic to derive a formal model has been done to ease the automation of SPD functionalities Composition. Indeed, once a semantic model is defined, an ontology can be derived (e.g. using the Ontology Web Language) and an instance of such ontology can be used to map a specific scenario. The ontology instance can be used to obtain the pS-XX interfaces of all the involved SPD functionalities.

The SPD status is expressed using the SPD metrics, thus its specification, in terms of syntax is provided in Section 4.6, where the SPD metrics are formalized. Instead its semantic is specified in the Semantic Model defined for the the whole pSHIELD in section 5.4. The actual value of the SPD status can be derived on-line, applying the Common Criteria (described and formalized in Section 5.7) to the specific functionalities.

The semantically enriched information contained in the description field of a pS-XX interface is used by the Core SPD Services to discover it. The function and contract information of a pS-XX interface are used by the Core SPD Services to compose it. The SPD status is continuously monitored by the Core SPD Services and is used by the Overlay to determine the best configuration rules.

4.4.2.2 pSHIELD SPD components composition

In order to enable the composability by the Overlay control algorithms, a formal model is needed to represent the atomic elements that we are willing to compose, in terms of *components*, *connectors*, *contracts* and *topology*. Even though these terms belongs mostly to the world of Software Engineering, we believe that they could be very useful to provide a more focused formal description tailored to the heterogeneous context of Embedded Systems (in general) and pSHIELD (in particular).

The pSHIELD SPD component, as seen by the Overlay, is an **atomic pSHIELD SPD component** that is a generic atomic SPD Functionality (innovative or legacy) provided by a pSHIELD device at node, network or middleware level (see the description at the beginning of this section); for example, an atomic pSHIELD SPD component could be a particular fault-tolerant routing protocol, or a specific cryptographic algorithm, or an authentication software module and so on. It is extremely important that each functionality is atomic because this assumption is at the basis of current methodology for security assessment and, consequently, at the basis of the pSHIELD mechanism for SPD quantification (see Common Criteria approach for further details).

In Figure 4.17 the formalized conceptual model of a pSHIELD SPD atomic component is represented: it is a simple data structure containing:

- the **name** of the SPD functionality,
- the semantic **description** of the interfaces for discovery purposes,
- the **SPD status** associated to it,
- the **Input / Output** parameters needed to properly work (Optional),
- the **contract** describing how the functionality can be composed by Core SPD services and, if applicable, the constraints to which is subject while composed with other SPD functionalities,
- one or more **connectors** with a connection ID (destination) and a type (see below).

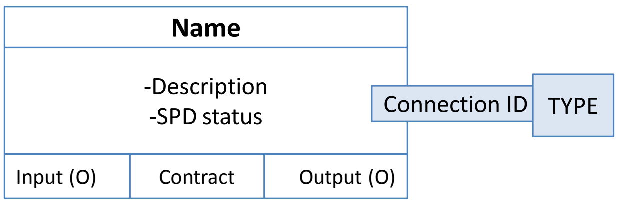


Figure 4.17 atomic pSHIELD component and connector

In the literature the model of an atomic component considers the input/output variables as “compulsory” fields, but this is not the case of pSHIELD, since, usually, the SPD Functionalities don’t share information or energy flows and can be considered as non-interacting entities, so this fields are only optional. A clear example could be the composition of SPD functionalities to build a trusted monitoring system that cipher the output of its sensors and register the access of people: these two functionalities can be composed together to enrich the SPD capability of the whole scenario, but they act independently. This is quite common in SPD context.

The **pSHIELD connector** is a formal representation of the composition of two SPD functionalities to satisfy a common objective, as depicted in the logic of the Medieval Castle of document M0.2. In particular, an SPD functionality is usually seen as a mean to protect a specific good or *asset*. In Figure 4.18 the good/asset is represented by a treasure stored in a room and protected by different SPD functionalities (doors). The aim of a malicious attacker is to reach the treasure by defeating the SPD Functionalities.

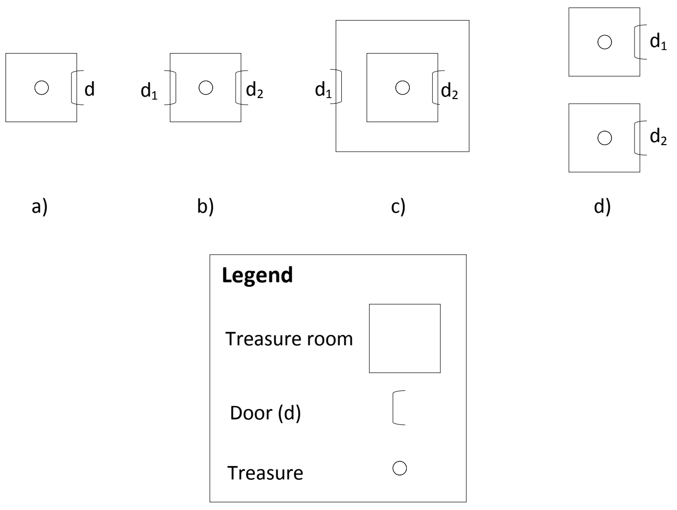


Figure 4.18 Medieval castle composition approach

So, in order to reduce the faults of the overall system, two (or more) SPD functionalities can be composed and applied together to the same asset. In particular three **types** of connections are foreseen in SPD literature (see the Common Criteria):

1) Parallel composition (PP): the SPD functionalities protect the same room, so it is sufficient to break one of them to reach the treasure (case b))

2) Concentric composition (CC): the SPD functionalities are composed serially (concentric) so that the attacker has to break all the protections to reach the treasure (case c))

3) Concurrent composition (CO): the treasure is distributed and the SPD functionalities are composed separately on the different assets and, moreover, one of these assets could be a “trap” (honey pot) to distract the attacker; in spite of this, the attacker can reach the treasure only with a certain probability p of choosing the right door to break (case d)).

It must be said that these three cases are the most significant, but other types of connection are possible and can be defined when needed, without affecting the overall approach.

Furthermore, when the result of the composition of two or more functionalities (named enriched functionality) is composed as a whole with another functionality, the enriched functionality can be modelled as a pSHIELD component as well (i.e. the model is still valid) with the exception that the SPD status needs to be quantified according to a specific procedure (see section 5.7). A set of examples is provided below.

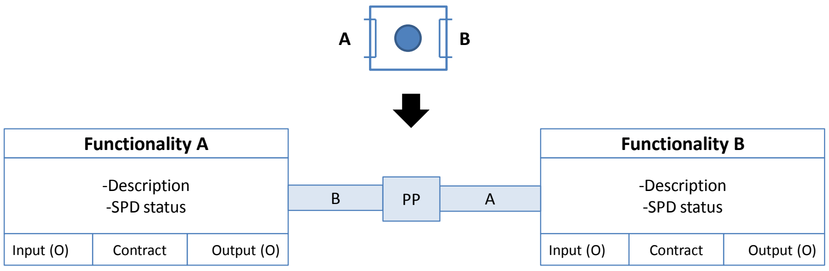


Figure 4.19 Parallel composition of two SPD functionalities

In Figure 4.19 an example of parallel composition is provided, where two functionalities A and B are composed together to protect the same asset at the same hierarchical level: breaking indifferently one of them will result in compromising the asset. Since they are independent functionalities composed together, no I/O relations are identified. The result could be indicated as A[PP]B that means A *Parallel Composed* with B. Of course it is possible also to compose more than two pSHIELD SPD functionalities with the same approach, and the result is provided in Figure 4.20 and could be indicated as A[PP]B[PP]C, that means A *Parallel Composed* with B and C.

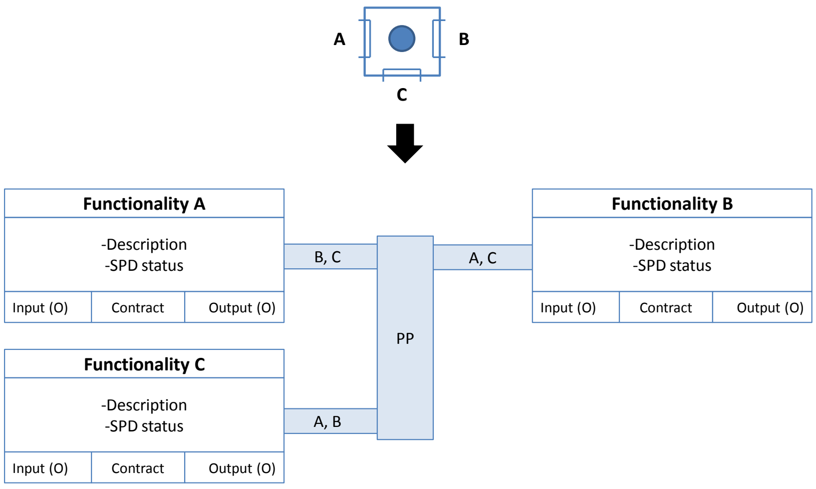


Figure 4.20 Parallel composition of three SPD functionalities

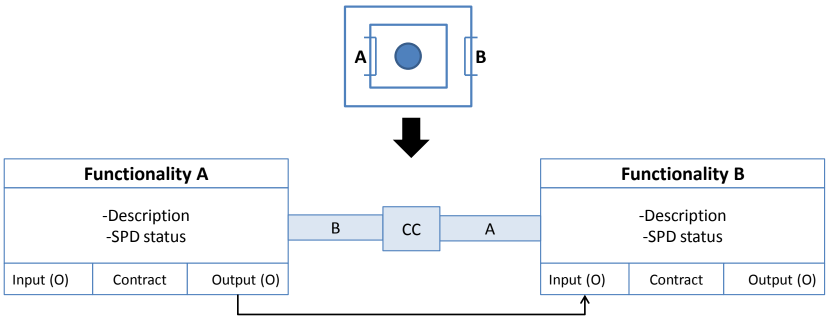


Figure 4.21 Formal model for concentric (hierarchical) composition of two SPD functionalities

Figure 4.21 represents an example of concentric composition of two functionalities A and B: the attacker has to break both of them to reach the treasure. Since they are concentric, in this example we have represented also a link between input and output parameters of the two functionalities. The result could be indicated as A[CC]B, that means A *Concentric Composed* with B. Note that the direction of the relation is not significant, because in the scope of the treasure protection it is not important the order of breaking functionalities, but the fact that both of them are compromised.

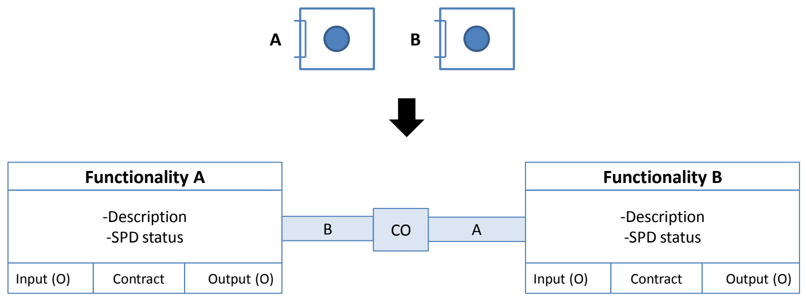


Figure 4.22 Concurrent composition of two SPD functionalities

In Figure 4.22 an example of concurrent composition is provided, where two functionalities A and B protect two parts of the treasure. The result is immediate and is indicated as A[CO]B, that means A *Concurrently Composed* with B.

The last example, reported in Figure 4.23, represent a complex scenario where four SPD functionalities are composed together in all the three ways: A and B protect in parallel one part of the treasure and C protect another part of the treasure, then both of them are protected by a concentric D functionality. In this case the attacker has, at first, to break functionality D to go further and to decide whether to attack functionality C or functionality A[PP]B.

Please note that the same functionality can participate to more than one composition relation.

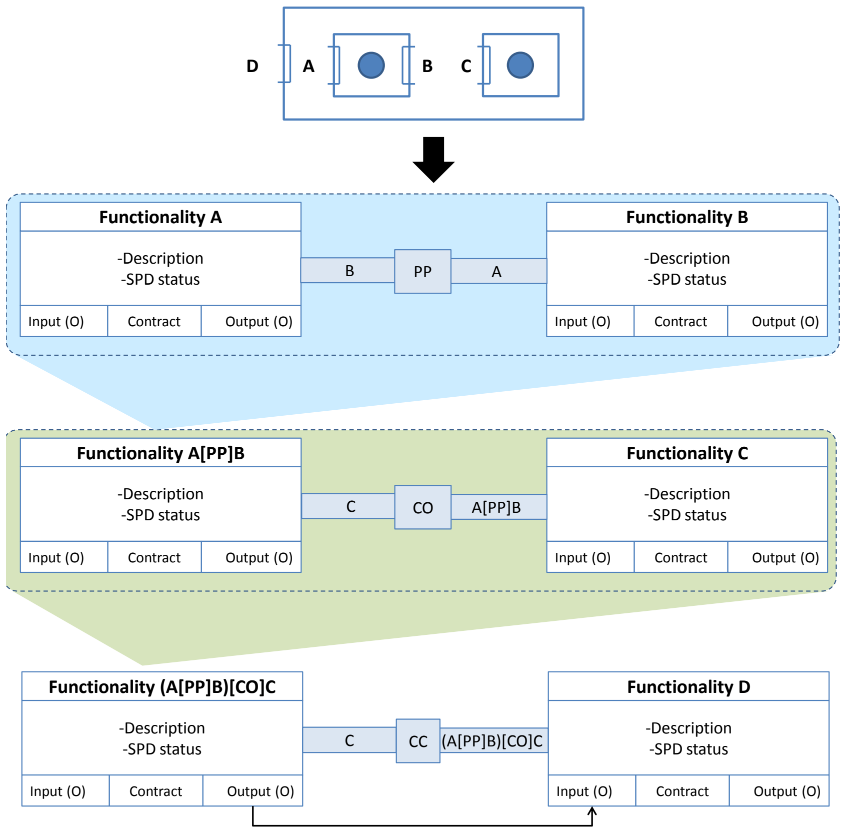


Figure 4.23 Parallel plus concentric plus concurrent composition of four SPD functionalities

These example should clarify the adopted approach concerning composability **topology**. Each component can be simultaneously composed with other components, so that a networked (n:m) topology is the most suitable to describe, from the Overlay point of view, composability topology (see Figure 4.24).

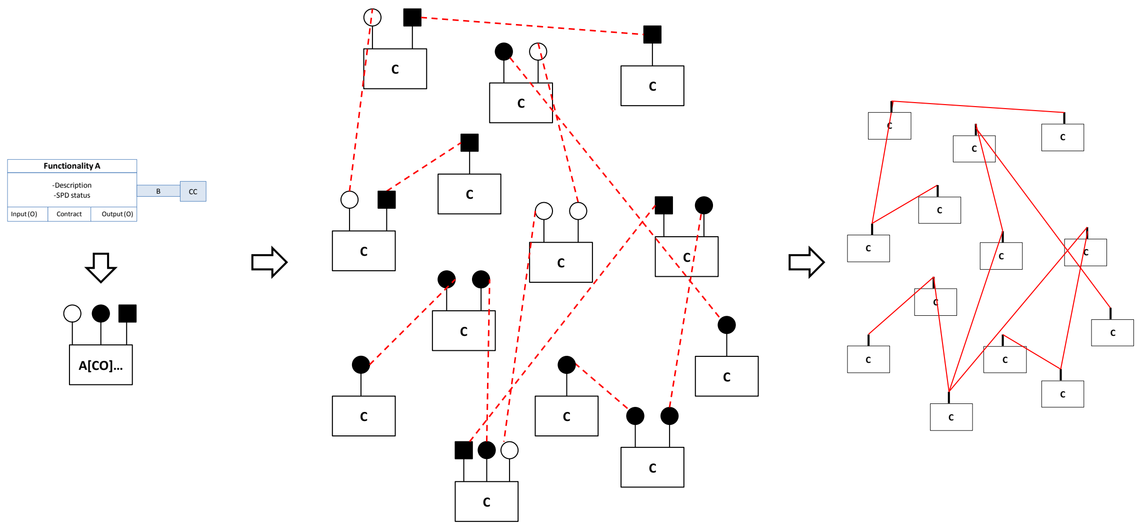


Figure 4.24 Networked (n:m) topology for composition

4.4.2.3 Control algorithms

The composability is performed by a set of Control Algorithms which, on the basis of a reference value which represents the target SPD level (defined according to the selected SPD Metrics) and of the representation of the available pSHIELD SPD atomic components, decide which configuration is the most suitable to reach the target SPD level. The pSHIELD architecture is flexible enough, so that different control algorithm methodologies can be adopted; in the pSHIELD project, as detailed in section 5.6, Hybrid Automata are the selected methodology.

4.4.2.4 SPD status quantification

The control algorithms are based on the assumption that SPD is measurable and quantifiable. As already said, a specific procedure to quantify the SPD level for each component is outlined in section 4.6, while the procedure to propagate the SPD levels through the composition, is reported in document M0.2.

4.5 Innovative SPD functionalities

4.5.1 Description

As highlighted in the previous section, the Innovative SPD functionalities reside in the pSHIELD Middleware, Network and Node Adapters.

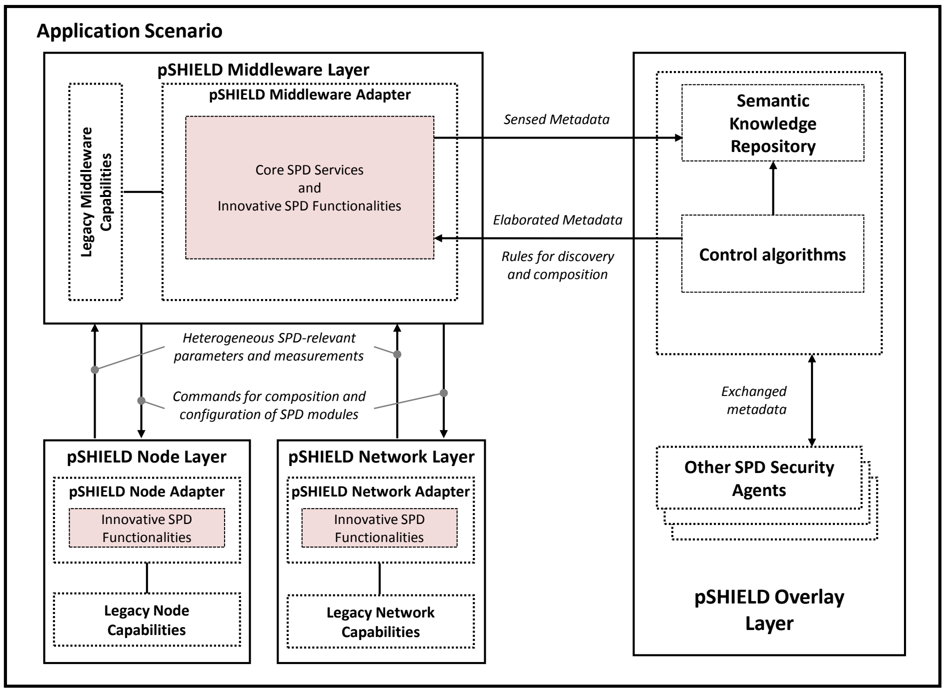

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.25 pSHIELD innovative SPD functionalities

They are constituted by a variety of pSHIELD-specific components. Each pSHIELD-specific component (as described in section 4.4.2.1) represents an innovative SPD functionality ad hoc developed for the pSHIELD project which is included in the pSHIELD Node, Network or Middleware Adapter.

The pSHIELD innovative SPD functionalities can be developed ex-novo or reusing the available legacy SPD functionalities provided by the already existing Legacy Embedded System Devices. Thus a pSHIELD innovative SPD functionality can be:

1. a completely novel SPD technology, not relying on any existent legacy SPD functionality that is pSHIELD discoverable and composable;
2. a legacy functionality enriched and enhanced in some SPD aspects and pSHIELD discoverable and composable;
3. a legacy functionality with no technology enhancement, but pSHIELD discoverable and composable.

The pSHIELD innovative SPD functionalities can be directly monitored and used by the pSHIELD Middleware Core SPD Services to apply the configuration rules elaborated by the Overlay.

More in detail the following enabling technologies have been investigated in the pSHIELD project to develop Node, Network and Middleware pSHIELD-specific SPD components:

- Nano, micro, personal and power node SPD functionalities described and formalized in section 5.1;
- Cryptographic algorithms described and formalized in section 5.2;
- Smart SPD driven transmission described and formalized in section 5.3;
- Core SPD services described and formalized in section 5.4;

4.6 SPD Metrics

4.6.1 Description

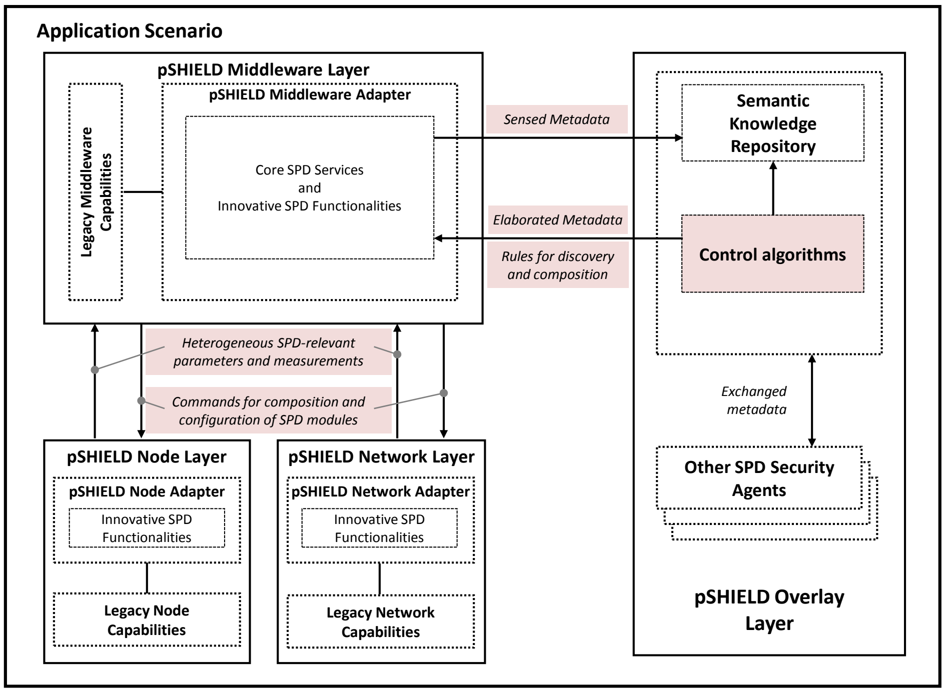

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 4.26 pSHIELD SPD metrics: a functional view

At the basis of the pSHIELD project (and, in particular, of the composability mechanism) there is the possibility to identify and quantify the SPD properties of each component, as well as the SPD properties of the overall system, i.e to define appropriate SPD Metrics.
SPD Metrics allow (i) users to define in an univocal way the requirements for the specific application, (ii) to describe the SPD level provided by the components, and (iii) to compute the SPD level achieved by the system through the Composability mechanism.

Figure 4.26 highlights in light red some of the information flows based on the selected SPD Metrics. In Figure 4.26 all the information flows that require the exchange of SPD metrics values are indicated (measurements from the physical system, configuration parameters, aggregated metadata and so on) together with the architectural component that widely uses these metrics (the control algorithms in the Overlay).

Since no common, standardized and shared metrics exist for Security, Privacy and Dependability tot-court, their identification will be the first innovative key concept provided by the pSHIELD project; the second innovation will be provided by the definition of a procedure to quantify them. In particular the project aims to model common SPD metrics capable of improving the overall SPD measure in any specific application domain, being technology independent.

4.6.2 Formalized conceptual model

4.6.2.1 Fault classifications

The concepts behind SPD metrics definition are extensively reported in deliverable D2.3.1. The underlying approach is very shortly outlined in the following.

Common SPD metrics can be defined introducing first the Threats and Faults concepts.

Threats (defined in D2.1.1, section 4) are expressed according to the following taxonomy:

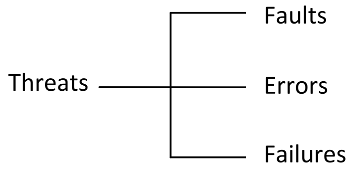


Figure 4.27 Threats taxonomy

where a *fault* is defined as a cause of an *error* and a *failure* is linked to the error that is outside of the error tolerance boundary.

Faults can be classified according to various approaches.

A first fault classification distinguishes between *human-made faults* (HMF) (i.e. the ones resulting from human actions) and *nonhuman-made faults* (NHMF) (i.e. the ones caused by natural phenomena without human participation). In turn, the former can be classified in faults with unauthorized access (FUA) (i.e. the ones caused by malicious attempts) and other faults (NFUA) (see Figure 4.28).

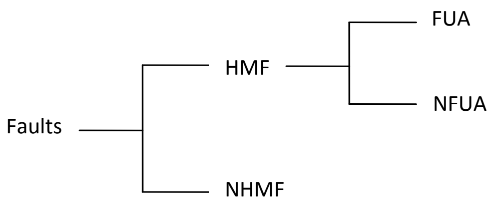


Figure 4.28 Faults taxonomy

A second fault classification (see D2.1.1, section 4, for further details) is related to the SPD attribute which is affected by the fault; so we distinguish among confidentiality faults, integrity faults, availability faults, reliability faults, safety faults and maintainability faults.

Confidentiality refers to the property that information or data are not available to unauthorized persons or processes, or that unauthorized access to a system's output will be blocked by the system's filter. Confidentiality faults are mainly caused by access control problems originating in cryptographic faults, security policy faults, hardware faults, and software faults. Cryptographic faults can originate from encryption algorithm faults, decryption algorithm faults, and key distribution methods. Security policy faults are normally management problems and can appear in different forms (e.g., as contradicting security policy statements).

Integrity refers to the absence of improper alteration of information. An integrity problem can arise if, for instance, internal data are tampered with and the produced output relies on the correctness of the data.

Availability refers to a system's readiness to provide correct service, whilst *reliability* refers to continuity of correct service, but these two attributes can considered as one because both guarantee the correct service with an error $e(t) < \varepsilon$. A typical cause of such faults is some sort of denial of service (DoS) attack that can, for example, use some type of flooding (SYN, ICMP, UDP) to prevent a system from producing correct output. The perpetrator in this case has gained access to a system, albeit a very limited one, and this access is sufficient to introduce a fault.

Safety refers to absence of catastrophic consequence on System users end environment. A safety fault can arise if, for instance, an unauthorized system access can cause the possibility of human lives being endangered.

Maintainability refers to the ability to undergo modifications and repairs

A third fault classification identifies eight elementary fault classes related to the nature of the fault as shown in Figure 4.29.

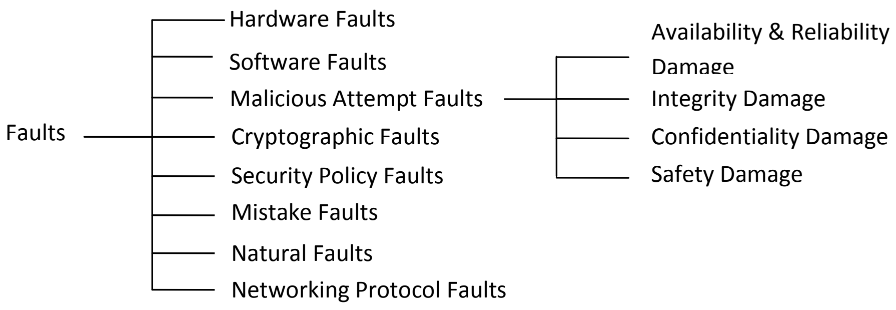


Figure 4.29 Elementary Faults classes

Figure 4.30 shows the existing relationships among the above-mentioned three fault classifications..

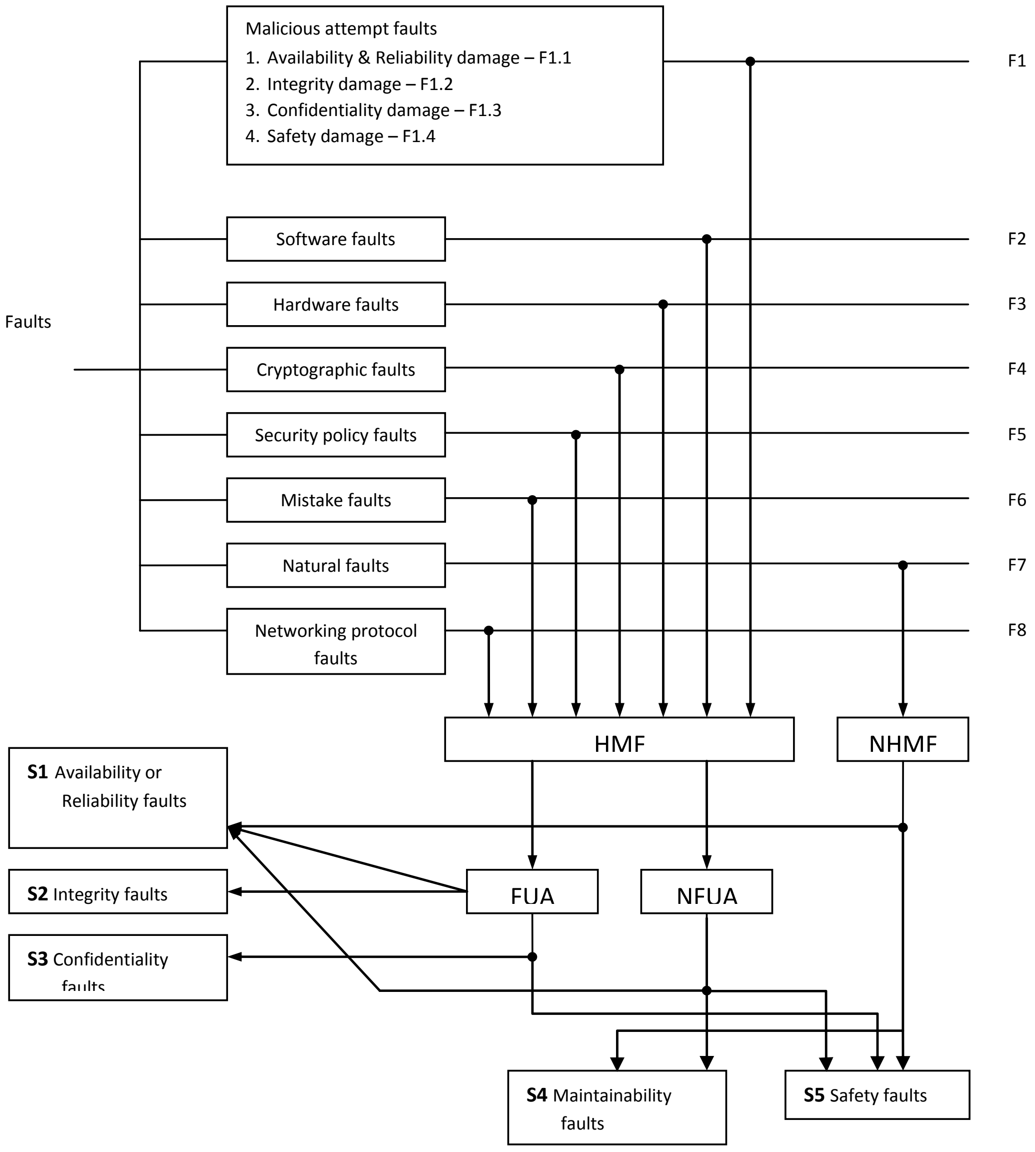


Figure 4.30 Tree representation of faults

4.6.2.2 SPD metrics identification and integration

The selected pSHIELD SPD Metrics refers to **fault tolerance**. In particular, each identified SPD functionality (in particular, each SPD component) will be characterized by a numeric value (reported in the SPD status of the SPD component) indicating how much is valid the implementation of that particular functionality to increment the tolerance to faults belonging to a given fault class. Note that fault classes result from the fault classifications performed in the previous section.

Fault tolerance levels are evaluated following a standardized approach based on **Common Criteria Standards (ISO15408)** as defined in Section 5.7: for this purpose, a vulnerability assessment activity with the purpose to estimate the robustness is performed. This evaluation will be based upon a vulnerability analysis and supported by testing (see D2.2.1 for procedure’s details).

The result will be a numeric value indicating the fault tolerance level (i.e. the SPD status) of the considered SPD functionality with respect to faults belonging to a given fault class.

A key issue related to Composability, will be the design of a procedure allowing to compute the overall SPD level of a considered pSHIELD subsystem on the basis of the SPD level of the atomic SPD components. In this respect, a methodology based on AND/OR/MEAN composition has been adopted and is described in M0.2 (it resembles the one adopted for the Medieval Castle approach described in Section 4.4).

5 pSHIELD enabling technologies

The key concepts described in the previous section and that contribute towards the realization of the pSHIELD overall objective, can be achieved and demonstrated by studying, designing and developing a set of innovative methodologies, technologies and functionalities that constitute the very first tangible results provided by the pSHIELD project.

Since these innovative solutions are considered as the most suitable to reach the pSHIELD goals, we will refer in general to them as the "pSHIELD enabling technologies", being them hardware components, software components, logical procedures or algorithms.

A one-to-many relation can be defined between the enabling technologies and the six key concepts, given the fact that a single technology can concur to the realization of more than one concept and vice versa (for instance the semantic models are the mean to achieve both composability and the seamless approach, or the composability needs both semantic models and SPD core services to be realized).

In Figure 5.1 the main enabling technologies are listed as well as their relation with the above mentioned key concepts. In this phase of the project a top-down approach has been followed to populate this list: starting from the descriptions provided in Section 3 and extrapolating the more suitable solutions to face them.

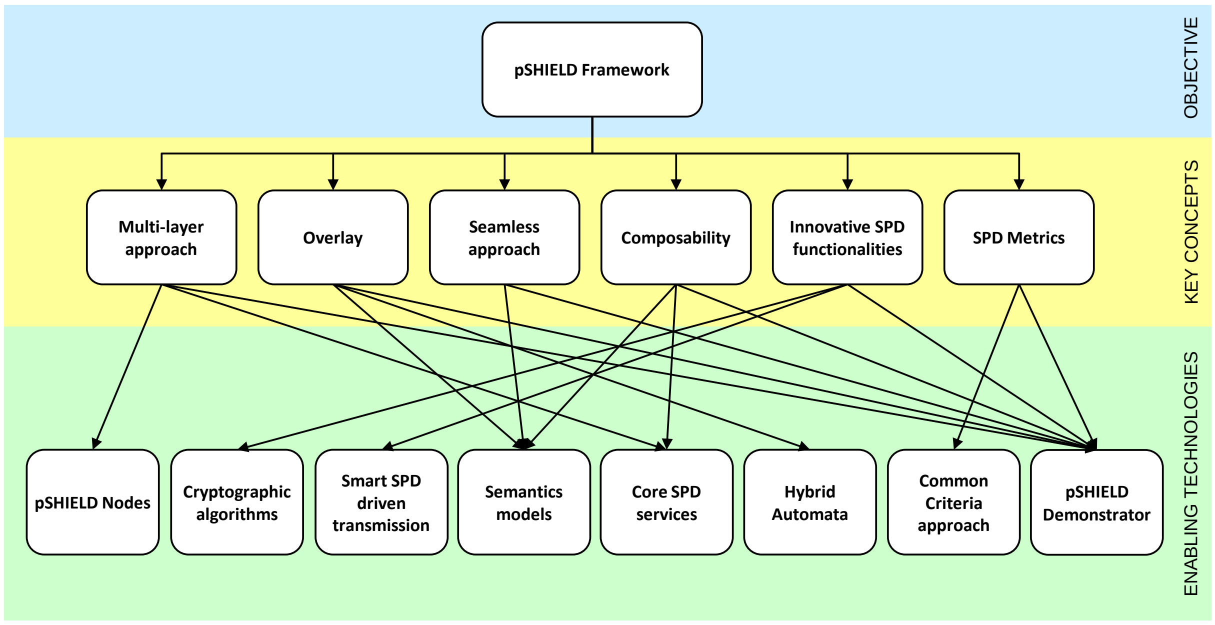


Figure 5.1 pSHIELD enabling technologies

The pSHIELD demonstrator is considered to be an enabling technology as well, since it constitutes a tangible proof of concept at the basis of the project and is the translation of all the formalized conceptual models into an actual working system.

5.1 pSHIELD nodes

5.1.1 A general node description

A **pSHIELD Node** is an Embedded System Device (ESD) equipped with several legacy Node Capabilities and with a pSHIELD Node Adapter. A pSHIELD Node is deployed as a hardware/software platform, encompassing intrinsic, Innovative SPD functionalities, providing proper services to the other pSHIELD Network and Middleware Adapters to enable the pSHIELD Composability and consequently the desired system SPD[2].
There are three kinds of **pSHIELD Node** deploying each different configuration of Node Layer SPD functionalities of the pSHIELD framework, and comprising different types of complexity: **Nano nodes**, **Micro/Personal nodes** and **Power nodes**. Nano nodes are typically small ESD with limited hardware and software resources, such as wireless sensors. Micro/Personal nodes are richer in terms of hardware and software resources, network access capabilities, mobility, interfaces, sensing capabilities, etc. Power nodes offer high performance computing in one self-contained board offering data storage, networking, memory and (multi-)processing.
While the three pSHIELD Node types cover a variety of different ESDs, offering different functionalities and SPD capabilities, they share the same conceptual model, enabling the pSHIELD seamless Composability.

5.1.1.1 Formal conceptual model

Figure 5.2 provides a conceptual model of a pSHIELD Node. This is a generic model for all the pSHIELD Node types, which can be implemented in different architectures, providing different functionalities, different SPD compliance levels and different services, depending on the type of node and application field (related work is EVITA project [42] and A. Ludovic et al. article[3]).

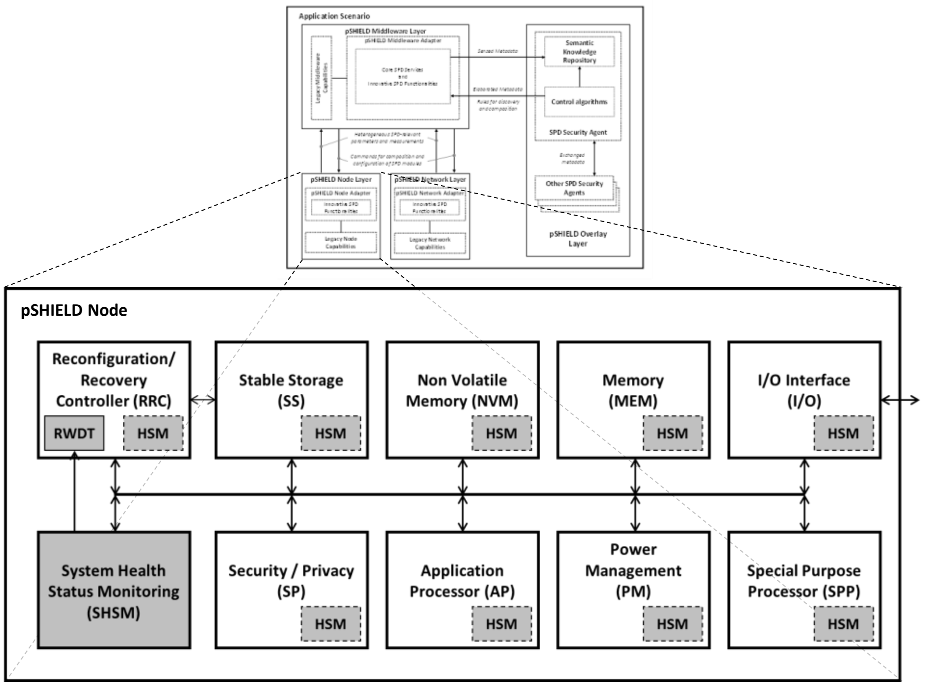

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer
pSHIELD Node
Reconfiguration/ Recovery Controller (RRC)
RWDT
HSM
Stable Storage (SS)
HSM
Non Volatile Memory (NVM)
HSM
Memory (MEM)
HSM
I/O Interface (I/O)
HSM
System Health Status Monitoring (SHSM)
Security / Privacy (SP)
HSM
Application Processor (AP)
HSM
Power Management (PM)
HSM
Special Purpose Processor (SPP)
HSM

Figure 5.2 Formal conceptual model of a pSHIELD Node

[2] “Security and Dependability of Embedded Systems: A Computer Architects’ Perspective” Jörg Henkel, University of Karlsruhe, Karlsruhe, Germany; Vijaykrishnan Narayanan, Pennsylvania State University, USA; Sri Parameswaran, University of New South Wales, Australia; Roshan Ragel, University of Peradeniya, Sri Lanka VLSID '09 Proceedings of the 2009 22nd International Conference on VLSI Design EEE Computer Society Washington, DC, USA ©2009

[3] Apvrille Ludovic et el., SECURE AUTOMOTIVE ON-BOARD ELECTRONICS NETWORK ARCHITECTURE,” http://www.eurecom.fr/util/publidownload.fr.htm?id=3132

The formal conceptual model of a generic pSHIELD Node can be derived from the pSHIELD functional component architecture shown in Figure 3.16. In order to show a generic model valid both for the pSHIELD Nano, Micro, Personal and Power Nodes, the different Node Layer Innovative SPD Functionalities (i.e. SPD components) are grouped into proper **modules** containing a functional subset of the Innovative SPD capabilities provided by the pSHIELD Node.

In brief, the main modules of a generic pSHIELD Node are:

- The **Application Processor** (AP), which is the main processing unit.
- A **Stable Storage** (SS), for storing the status of the system, a bit stream to program an FPGA, and/or the software for system start-up, operating system and application.
- ROM, EEPROM, FLASH, Hard Disk or other forms of **Non-Volatile Memory** (NVM).
- RAM, SRAM, DRAM, or other forms of volatile **Memory** (MEM).
- **I/O Interface** (I/O) to connect to any peripheral and to the rest of the pSHIELD embedded functionalities.
- **Special Purpose Processor** (SPP) module for any pre- or post-processing, such as compression/decompression, conversion, etc.
- **Power Management** (PM) module for managing power sources, monitoring power consumption, etc.
- **Secure/Privacy** (SP) module to perform security and privacy actions, such as encryption, decryption, key generation, etc.
- **Reconfiguration/Recovery Controller** (RRC), for recovering the system in case of error, and for reconfiguring it on demand.
- **Health Status Monitoring** (HSM) for checking the status of each individual component.
- **System Health Status Monitoring** (SHSM) for checking the status of the whole system.
- **Recovery Watchdog Timer** (RWDT) for restarting recovery if no activity is detected from the SHSM.

Depending of the type of node, application, technology, etc. each of these modules may be implemented with different pSHIELD SPD functionalities or even be not implemented.
More information on each of these modules is provided in section 5.1.1.2.
The pSHIELD Node may be supported by generic hard boards with CPUs or PICs and FLASH memory, special designed boards, boards with FPGA (partially dynamically reconfigurable or not), etc.
Section 5.1.1.5 shall describe some different configuration possibilities for these boards, depending on node types.

5.1.1.2 Detailed module description

5.1.1.2.1 Health Status Monitoring

Each module is monitored by a special Health Status Monitoring (HSM) module. Each of these modules is specially designed to monitor the module it is attached to, depending on its specific characteristics. This may be a solution by hardware, software or both. These modules are responsible for collecting status information from the monitored modules and send it through a periodic heartbeat to the System Health Status Monitoring (SHSM).
If an error is detected in the monitored module, depending on its severity, the HSM may either send the error data inside the status information, or stop sending the heartbeat. If possible, the HSM inhibits further actions from the erroneous module, preventing error propagation.
The SHSM module supervises all HSM modules, and sends a heartbeat to the Recovery Watchdog Timer (RWDT), embedded in the Reconfiguration/Recovery Controller (RRC). If one of the HSM fails the periodic heartbeat, the SHSM also stops its own heartbeat, and the RWDT triggers RRC for recovery (e.g. by resetting the system).
The SHSM may also perform its own monitoring activities, such as performing a Power-On Self-Test.
The SHSM should connect the Stable Storage, if present, to store all status information for a possible post-mortem analysis.

The connection between the different HSMs and the SHSM may be through the system bus, a dedicated bus, a point-to-point connection, or any combination of these. The connection between the SHSM and the RWDT must be through a dedicated point-to-point connection.
The HSM may be implemented as a hardware solution, periodically monitoring the activity in some connections, registers or pins of the processor, such as data or address buses, and stored status information in internal registers. Then it sends the information to the SHSM, either through a client or a master interface. It must possess an internal timer for detecting inactivity.
The HSM may also be implemented as a firmware solution (as an alternative or complementing the hardware solution). This software can be a thread or a process inside the monitored module which, periodically, collects health status information (internally or in the stable storage, if available) and sends it to the SHSM.

5.1.1.2.2 Security/Privacy

The Security/Privacy controller consists of one or more modules able to perform different security-related functionalities, such as Data Encryption, Data Decryption, Generation of Cryptographic Keys, etc.[4]
These modules may be simple memory-mapped devices, or secure cryptoprocessors. They are connected to the bus by a master or slave interface. They may also run as processes inside the application processor. Section 5.2 provides a description of the cryptographic algorithms that shall be used.
Each of these controllers should have a HSM, periodically collecting status information, and sending it to the SHSM. These HSM may be hardware, e.g. collecting activity of the devices, or software, if they have any software implementation or device driver.

5.1.1.2.3 Application Processor

The application processor consists on a processing unit that can range from a single microcontroller to multiple processor cores, based on FPGA (soft-core)[5], hard-core, or both.
This module includes an interrupt controller, and might possess cache memory, a memory management unit, a floating point unit, and a debug interface.
This processing unit interfaces with a bus (such as AMBA High-Speed Bus – AHS) to access every module that may be available, such as RAM, ROM, FLASH, other memories, high-speed I/O controllers (such as Ethernet and USB), low-speed I/O controllers (such as serial), and the status and configuration registers from Security/Privacy modules, Special Purpose Processing modules and Power Management modules.
The application processor must have access to the stable storage, if available, for checkpointing.
It may also interface with the Reconfiguration/Recovery Controller for system reconfiguration, and the Health Status Monitoring module for gathering status information.
It is advisable this processor to contain an embedded HSM module, which is responsible for checking the health status of the application processor. This may be done at hardware or firmware level.

5.1.1.2.4 Power Management

The Power Management module is responsible for managing power supply, such as switching to redundant power sources, or checking the level of power consumption. This may be hardware or firmware, running on a dedicated processing unit or in the application processor.
A HSM module should also monitor the activity of this module.

5.1.1.2.5 Special Purpose Processor

The Special Purpose Processor is a pre- or post-processing unit, realizing specific dedicated activities, such as decompressing input data, or compressing output data and digital signal processing. This is a

[4] “Experimental Security Analysis of a Modern Automobile” Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno Department of Computer Science and Engineering University of Washington Seattle, Washington 98195–2350 Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage Department of Computer Science and Engineering University of California San Diego La Jolla, California The IEEE Symposium on Security and Privacy, Oakland, CA, May 16–19, 2010.

[5] Fault-Tolerant Resynthesis for Dual-Output LUTs Ju-Yueh Lee, Electrical Engineering Department, University of California, Los Angeles Yu Hu, Electrical Engineering Department, University of California, Los Angeles Rupak Majumdar, Computer Science Department, University of California, Los Angeles Lei He, Electrical Engineering Department, University of California, Los Angeles Minming Li, Computer Science Department, City University of Hong Kong Proceedings of the 2010 Asia and South Pacific Design Automation Conference IEEE Press Piscataway, NJ, USA ©2010

hardware unit, implemented with a dedicated microprocessor or microcontroller, or with a logic circuit. This module should have its own HSM, running at hardware or firmware level, collecting status information and sending it to the SHSM.

5.1.1.2.6 I/O Interface

The I/O interface consists of a series of High-Speed I/O controllers, such as USB hosts or Ethernet MACs (Media Access Controller) or other controllers, and Low-Speed I/O controllers, such as UART hosts or other peripheral ports. They access the bus through a master or slave interfaces.
Each of these controllers should have a HSM, periodically collecting status information, and sending it to the HSM. These HSM may be hardware, e.g. collecting activity of the devices, or software, collecting the status at device driver’s level.

5.1.1.2.7 Memory

The Memory module includes both the memory controller and the RAM memory (SRAM, or DRAM).
The HSM may consist of any error correction and detection module, which preferably would also store information about every detected and corrected error.

5.1.1.2.8 Non-volatile Memory

The Memory module includes both the controllers and the non-volatile memory, such as FLASH, ROM, or even Hard-Disk (solid state).
The HSM may consist of hardware or software, detecting and correcting any error, and preferably storing status information.

5.1.1.2.9 Stable Storage

Stable Storage[6] is a memory whose contents survive system malfunctions. It must resist to external and internal failures, and guarantee atomic reads and writes. It is usually implemented using two memory banks, and when data is send to this memory, it is first written in bank1, then internally into bank2 and, only when both operations are completed, the stable write is considered done. Some error detection codes are also added, i.e. ECC , EOS, Parity algorithms.
A stable read consists of checking parity bits and comparing both banks. Only when both banks are corrupted, or are different from each other, but both with valid codes, the read fails.
A simpler version could be simply the addition of error detection and correction codes.
The control of stable writes and stable reads may be performed by hardware or software.
Depending of the stable storage control, the HSM may consist of hardware, software, or both. Every detected error (tolerated or not) should be stored for informing the SHSM.
This features are necessary to implement the rollback recovery strategy.

5.1.1.2.10 Reconfiguration/Recovery Controller

This is a hard processor or microcontroller, responsible for either reconfiguring the node or recovering in case of an error. It should be a hard device, independent from the rest of the system (e.g., it should not be part of the same FPGA that contains some parts of the system, such as the AP)[7].
The reconfiguration of the system is made by request, usually by an application running on the AP which, in turn, has received the request from the network, through any local input device, or after a local decision based on the system status (e.g., a certain threshold of system health or power level has been reached).
Reconfiguration may consist on connecting/disconnecting some devices, or reprogramming the FPGA, either totally or partially.[8]

[6] “Software-Implemented Stable Storage in Main Memory”, João Carlos Cunha, Dep. Engenharia Informática e de Sistemas, Instituto Superior de Engenharia de Coimbra; Centro de Informática e de Sistemas da Universidade de Coimbra Coimbra, Portugal; João Gabriel Silva, Dep. Engenharia Informática, Universidade de Coimbra, Centro de Informática e de Sistemas da Universidade de Coimbra Coimbra, Portugal IX Brazilian Symposium on Fault-Tolerant Computing (SCTF), Florianópolis/SC, Brazil, March 2001

[7] “Rewiring For Robustness” Manu Jose, Computer Science Department, University of California, Los Angeles YuHu, Electrical and Computer Engineering Department, University of Alberta RupakMajumdar, Computer Science Department, University of California, Los Angeles Lei He, Electrical Engineering Department, University of California, Los Angeles Proceedings of the 47th Design Automation Conference ACM New York, NY, USA ©2010

[8] “IPR: In-Place Reconfiguration for FPGA Fault Tolerance” Zhe Feng, Electrical Engineering Department Yu Hu, Electrical Engineering Department Lei He, Electrical Engineering Department Rupak Majumdar2, Computer Science Department University of California, Los Angeles ICCAD ’09, November 1-4, 2009, San Jose, California, USA.

Recovery is always triggered on error, and by action of the Health Status Monitoring. At least, a RWDT (recovery watchdog timer) must be periodically set by the SHSM and, on timeout, starts a basic system recovery.
Recovery may consist on AP reset (only software recovery), on system reset, on FPGA reprogramming (partially or totally), or on any other more complex activity, such as switching to a spare module (on a stand-by spare), disconnecting a redundant module (on an N-Modular Redundancy), etc.[9]
The basic recovery action consists on reprogramming the FPGA or, if it doesn't exist, on resetting the system.
The RRC, by request of the SHSM or by the action of two consecutive timeouts from the RWDT (if the SHSM is failing) may also halt the whole system, due to a permanent failure. However, if the RRC itself fails (its own HSM module should detect it), the SHSM may also be able to halt the system.

5.1.1.3 pSHIELD Node SPD components

A pSHIELD Node must provide to the other layers of the pSHIELD framework a set of Node Layer Innovative SPD Functionalities that comply with the pSHIELD conceptual model.
This section describes the pSHIELD SPD components provided by the pSHIELD Node Layer.

5.1.1.3.1 Security and Privacy SPD functionalities

- **Encrypt/Decrypt data** – allows the encryption and decryption of data for local storage, transmission over the network or even communication with other peripherals.
- **Secure Firmware Upgrade** – allows secure firmware upgraded either locally or remotely, for system configuration
- **Login/Logout** – allows a user to login or logout either locally or remotely
- **Secure Connect/Disconnect** – establishes a secure connection to a remote node or other peripheral
- **Secure Send/Receive** – exchanges data with a remote site in a secure way

5.1.1.3.2 Dependability SPD functionalities

- **Stable read/stable write** – reads and write data, e.g. a checkpoint, in stable storage
- **Get health status** – gets the health status information of the whole system
- **Reconfigure** – requests reconfiguration of the system. This reconfiguration may be the connection or disconnection of a device, the reconfiguration of an FPGA, etc.
- **Recover** – Requests recovery of the system from failure. This recovery may be partial (a module, a block from the FPGA, only software, etc.) or total (e.g. write full bit-stream in the FPGA and restart system)
- **Fail safe** – requests system to go to a safe state and stop
- **Self-test** – requests for a partial or full self-test of the system
- **Degrade functionality** – requests a system reconfiguration to function in a degraded mode, e.g. for power saving
- **Degrade dependability** – requests a system reconfiguration to decrease dependability, e.g. after failure of a redundant module.
- **Change power** – requests a switch to another power source

[9] "A New Placement Algorithm for the Mitigation of Multiple Cell Upsets in SRAM-based FPGAs" L. Sterpone, Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy N. Battezzati, Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy Design, Automation & Test in Europe Conference & Exhibition, 2010

5.1.1.3.3 Performance/Metrics SPD functionalities

- **Get performance/metrics** – gets performance and metrics information from the whole system

  Following, the list of the metrics provided by the node:

  1. System and components health status
  2. System and components configuration;
  3. Power consumption;
  4. Power supply status;
  5. Number of detected errors per type and component;
  6. Number of recoveries per types and component
  7. Failed components;
  8. Number of intrusion attacks;

5.1.1.3.4 Discovery/Composability SPD functionalities

- **Discovery** – provide to the pSHIELD Middleware Adapter the information, raw data, description of available hardware resources and services in order to allow the system composability
- **Connect/Disconnect** – connects or disconnects specific SPD functionalities for system composability;

5.1.1.3.5 Miscellaneous SPD functionalities

Depending on the application field, other services are provided, mainly related to the Special Purpose Processor modules:

- **Compress/decompress** – requests data compression or decompression for local storage or exchange over the network or with peripherals
- **Configure/calibrate** – requests the configuration or calibration of a device attached to the node
- **Digital Signal Processing** – digital signal acquisition and conversion (ADC/DAC)

5.1.1.4 Intrinsic SPD capabilities

To comply with SPD requirements, the pSHIELD Node has some Node Layer intrinsic, architectural characteristics specially designed to provide dependability, security and privacy exclusively at Node layer. This intrinsic SPD capabilities can be configured by the pSHIELD Overlay and composed by the pSHIELD Middleware Core SPD Services, however they apply autonomously and continuously in the pSHIELD Node. Let see in detail how intrinsic SPD capabilities can be provided by the pSHIELD Node.

5.1.1.4.1 Dependability

Dependability is mainly assured by the Health Status Monitoring (HSM) modules, attached to each of the other modules of the pSHIELD node. If any error is detected, a centralized System HSM module triggers system recovery, performed by the Reconfiguration/Recovery Module. If the SHSM itself fails, the recovery watchdog timer (RWDT) starts system recovery.
Other modules also provide other aspects of dependability, such as the Power Management (power failures) and the Stable Storage (for recovery).
There are thus several levels of dependability:

- Each module has a HSM module that monitors its health and periodically sends health status information to the SHSM
- On error, the HSM may inhibit the monitored module, performing a fail-fast operation

- If the SHSM stops receiving status information from one of the HSM, or receives error information, or even the information itself is erroneous, it starts a recovery procedure, instructing the RRC.
- The SHSM may also perform other health status monitoring operations, such as checking activity on the bus or performing a POST
- If the SHSM fails, the RWDT starts system recovery
- If the RRC fails, the SHSM halts the system
- On permanent failure of one of the modules, the RRC may halt the system
- The PM assures system availability by managing redundant power sources or triggering a low-power mode if power level is low.
- The Stable Storage assures data survivability for rollback-recovery.

The pSHIELD power node may exhibit advanced recovery and reconfigurability capabilities through partial FPGA reconfiguration[10]. Recent advances in FPGA technology offer the possibility of repairing a failed module by reloading the bit stream in the FPGA frames that contained this module[11]. Furthermore, this FPGA reconfiguration may be used for changing the device functionality during runtime.
Also depending on application criticality, other forms of fault-tolerance may be used, such as static redundancy (e.g. Triple Modular Redundancy - TMR)[12] or dynamic redundancy, such as stand-by spare. This redundancy may be applied for each one of the modules that constitute the pSHIELD Node, even Nano Node.

5.1.1.4.2 Security and Privacy

Security and privacy are assured by the Security/Privacy module. The level of security and privacy depends on the modules that are implemented, which may assure, for example, Data Encryption, Data Decryption, Generation of Cryptographic Keys, etc.
A detailed description of the cryptographic algorithms enabled by the pSHIELD nodes is provided in section 5.2.

[10] "In-Circuit Partial Reconfiguration of RocketIO™ Attributes", http://www.xilinx.com/support/documentation/application_notes/xapp662.pdf
"Two flows for Partial Reconfiguration: Module Based or Difference Based", http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf
"Dynamic Reconfiguration of RocketIO MGT Attributes", http://www.xilinx.com/support/documentation/application_notes/xapp660.pdf
[11] Cheatham (portal.acm.org/citation.cfm?id=1142167)
[12] "On the Reliability of Cascaded TMR Systems", Masashi Hamamatsu, Nomura Research Institute, Ltd., Yokohama-City, Japan, Tatsuhiro Tsuchiya Tohru Kikuno, Osaka University, Suita-City, Japan, 2010 Pacific Rim International Symposium on Dependable Computing

5.1.1.5 Summary of SPD components required for pSHIELD Power Node, Micro/Personal Node and Nano node

As previously stated, different pSHIELD node types are enabled by different technologies and provide different functionalities.

pSHIELD node module	Power node	Micro/Personal node	Nano node
Application Processor (AP)	Multi-core processor (hw and/or sw core)	Microcontroller (hw and/or sw core)	Microcontroller
Stable Storage (SS)	Flash-based, 2 memory banks, w/ hw or sw control	Flash-based, single memory bank	N.A.
Non-volatile memory (NVM)	ROM, EEPROM, FLASH, Hard Disk or other forms of non-volatile memory	ROM, EEPROM, FLASH	ROM, EEPROM, FLASH
Volatile memory (MEM)	RAM, SRAM, DRAM	RAM, SRAM, DRAM	RAM
I/O Interface (I/O)	USB, ETHERNET, UART, CAN, RS232, RS485, GPIO	GPIO,ETHERNET, RS232, CAN	SERIAL; Wi-fi; RF-ID;BT; Zigbee;
Special Purpose Processor (SPP)	Hardware digital signal processing (DSP) and/or glue logic blocks	ADC	ADC
Power Management (PM)	UPS, Power Monitoring Device	Power Monitoring Device	N.A.
Secure/Privacy (SP)	AES Encryption TPM Module	TPM module OTP (one time) Password	N.A.
Reconfiguration/Recovery Controller (RRC)	IP Core	IP Core, ASIC or VLSI	N.A.
Health Status Monitoring (HSM)	IP Core	IP Core, ASIC or VLSI	N.A.
System Health Status Monitoring (SHSM)	IP Core	IP Core, ASIC or VLSI	N.A.
Recovery Watchdog Timer (RWDT)	IP Core	IP Core, ASIC or VLSI	N.A.

Table 1 pSHIELD enabling technologies by node types

To formalize in another way the available (A), optional (O) or not available (N.A.) SPD components for each pSHIELD Node Type (reported in the table columns as Power Node, Micro/Personal Node and Nano Node) and for each pSHIELD Node module (reported in the table as rows) are detailed which Node Layer SPD components are mandatory.

	pSHIELD Node Type		
pSHIELD Node SPD functionalities	**Power node**	**Micro/Personal node**	**Nano node**
Security/Privacy functionalities			
Encrypt/Decrypt data	**[A]**	**[A]**	**[N.A.]**
Secure Firmware Upgrade	**[A]**	**[A]**	**[O]**
Login/Logout	**[A]**	**[A]**	**[A]**
Secure Connect/Disconnect	**[A]**	**[A]**	**[N.A.]**
Secure Send/Receive	**[A]**	**[A]**	**[A]**
Dependability functionalities			
Stable read/stable write	**[A]**	**[O]**	**[N.A.]**
Get health status	**[A]**	**[O]**	**[N.A.]**
Reconfigure	**[A]**	**[N.A.]**	**[N.A.]**
Recover	**[A]**	**[A]**	**[N.A.]**
Fail safe	**[A]**	**[O]**	**[N.A.]**
Self-test	**[A]**	**[O]**	**[O]**
Degrade functionality	**[A]**	**[O]**	**[N.A.]**
Degrade dependability	**[A]**	**[O]**	**[N.A.]**
Change power	**[A]**	**[O]**	**[N.A.]**
Performance/Metrics services	**[A]**	**[A]**	**[N.A.]**
Discovery & Composability functionalities			
Discovery	**[A]**	**[A]**	**[A]**
Connect/Disconnect	**[A]**	**[A]**	**[A]**
Miscellaneous functionalities			
Compress/decompress	**[A]**	**[O]**	**[N.A.]**
Configure/calibrate	**[A]**	**[O]**	**[A]**
Digital Signal Processing	**[A]**	**[N.A.]**	**[N.A.]**

Table 2 pSHIELD services by node types

Legend:
[A]: available
[O]: optional
[N.A]: not available

Let see more in detail the hardware characteristics of the different types of pSHIELD Nodes:

- pSHIELD Power Node in section 5.1.2;
- pSHIELD Micro/Personal Node and Nano Node in section 5.1.3.

5.1.2 Power nodes

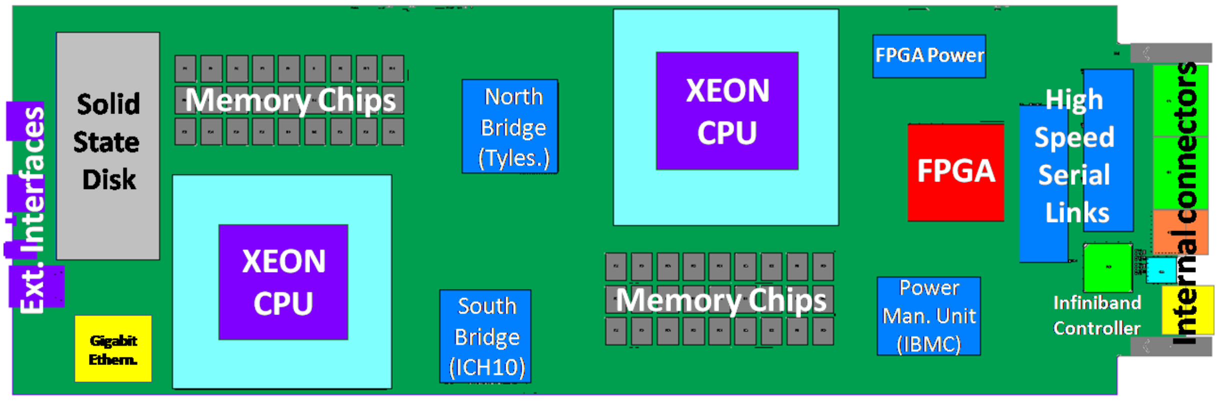
The Power Node has been conceived as a rugged and mobile high performance embedded system, optimally designed in terms of dimensions, weight, power consumption and capable to work in harsh environmental conditions. The reference application context is defence/aerospace ground mobile and airborne environments, addressing manned and unmanned applications where reliable high performance computing is required.

The following description illustrates a possible hardware solution that implements a Power Node, accordingly with the general model detailed in section 1.1.1. We intend to adopt this approach for the development of the Power Node prototype.

5.1.2.1 Power Node: the adopted solution.

The proposed implementation of the Power Node is based on a powerful computing architecture: a dual Intel Xeon 5500/5680 series (Quad core CPU) motherboard, with at least 6GB of on-board soldered DDR3 memory and a high data retention 80GB SSD drive. A high speed, high density FPGA device will also be present, providing easy adaptability and implementation of dedicated functions and special algorithms related to SPD functionalities. It will offer a maximum processing power of around 80GFlops.

In the following images the concept of a possible Power Node device is described. The first image illustrates the form factor of the Power Node board and the positioning of the components on the board itself. The second image represents the board covered by a cold-plate that can be air or liquid cooled. The shape of this cold-plate is intended, at this stage of the project, only for descriptive purposes. The final version could be different.



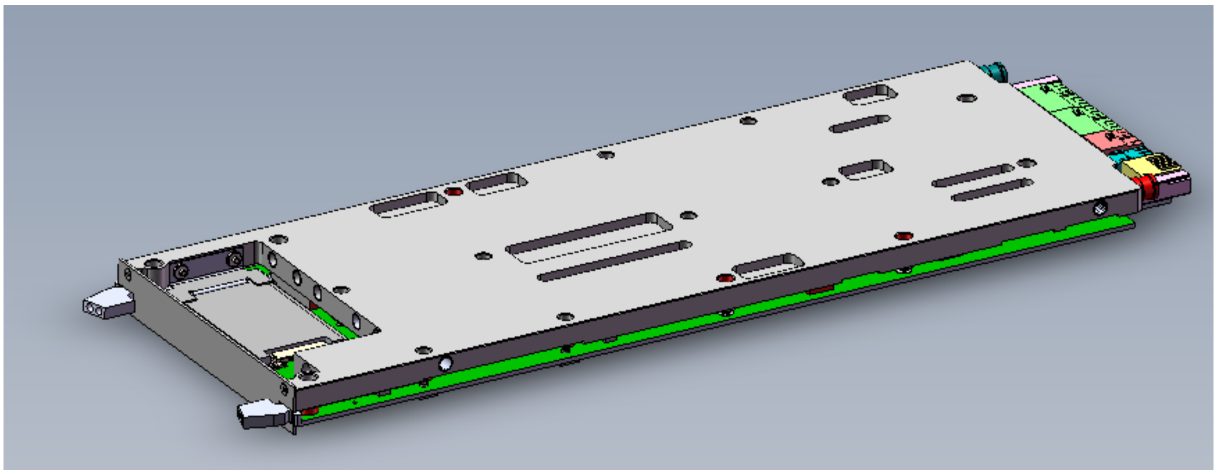


Figure 5.3 Power Node board concept, with and without cooling heat sinks

5.1.2.2 Power Node: hardware related aspects.

A possible hardware solution of Power Node consists of a High Performance Platform based on Nehalem/Wesmare Xeon Intel dual-processor board with Tylesburg chipset; it is equipped with a high density FPGA and a high speed Infiniband controller, moreover there is an Ethernet Gigabit interface. Every component is supervised by a Power Management Controller Unit (IBMC).

The Power Node core architecture will consists of two Intel Xeon X5680 or X5570 CPUs connected via Quick Path Interconnect (QPI), a dedicated low latency and high bandwidth bus capable of up to 6.4GT/s. Three channels of DDR3 memory are connected to each CPU, which integrates a high performance memory controller. The system hub (I/O Bridge) will be an Intel 5520 (Tylersburg) chipset and provides connectivity between the CPUs and the rest of the system; each CPU is connected to its Tylersburg with a QPI link. A Mellanox QDR ConnectX2 adapter is connected to the Tylersburg via one x8 PCIe 2.0 link: it provides a high Infiniband compliant connection. The hardware programmable part of the Power Node Is represented by an Altera Stratix IV FPGA, that is connected to the Tylersburg with 2 x8 PCIe 2.0 links.

Finally, the peripheral hub (Intel ICH10) is connected to the Tylersburg and provides the following additional peripherals:

- one optional SATA SSD, used to provide local fast and permanent storage;
- one Zoar Gigabit Ethernet adapter;
- 2x external accessible USB ports;
- one Output Video Port;
- one UART for low level debug.

The independent, embedded controller for the Power Management (IBMC) allows the monitoring of each performance parameters, such as temperatures, voltages, etc. Access to these parameters can be done by the Power Node applications, locally and remotely over the network. The IBMC provides an SNMP interface to the Power Node and allows setting traps for specific events. It can also trigger and monitor the Power-On-Self-Test. In terms or remote control, the embedded IBMC permits the remote configuration of the Power Node through the network and additional remote configurability can be done through the FPGA. The overall Power Node architecture of the solution that could be used is depicted in Figure 5.4.

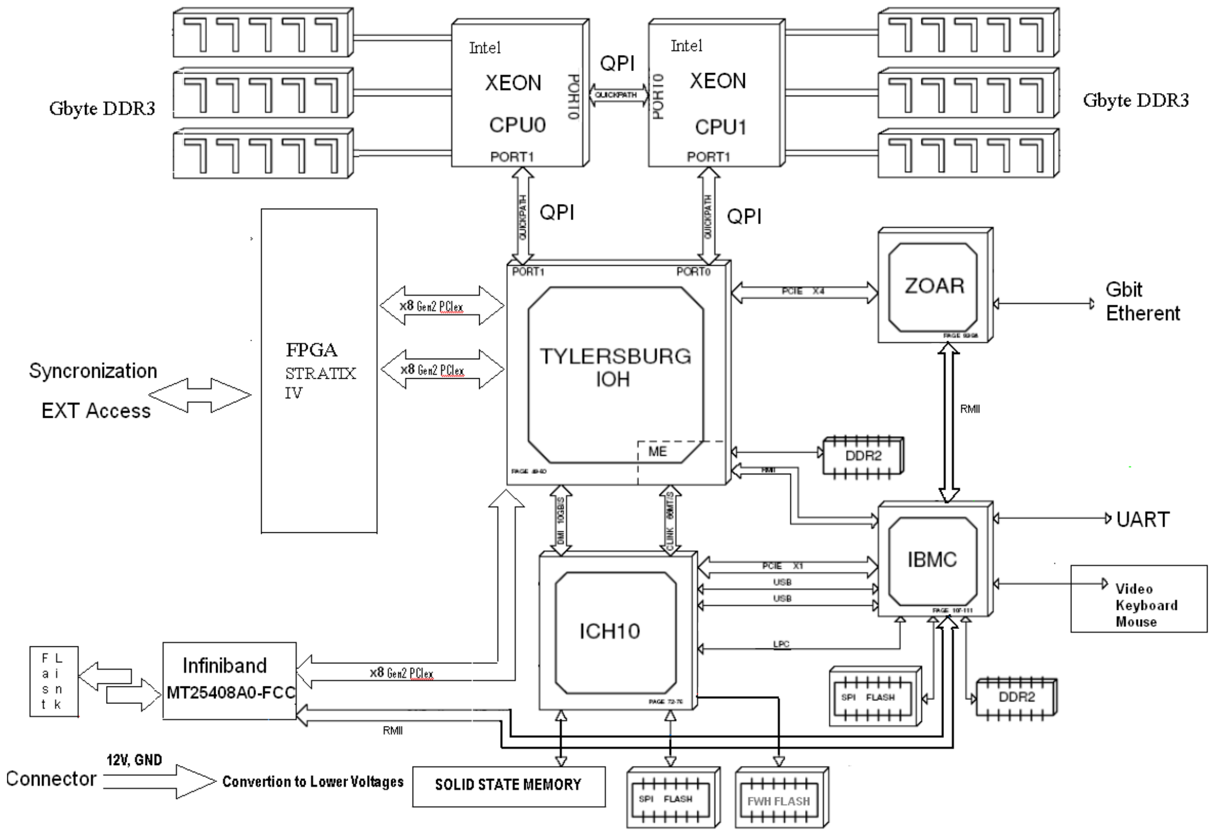


Figure 5.4 Power Node architecture: high level description.

The FPGA Processor is responsible for some security aspects. It includes a core logic that monitors the security of the Power Node. Tampering with the node triggers a protection mechanism in the security node that:

- Physically disconnects any I/O and network,
- Deletes any data resident on the node,
- Initiates the physical destruction of the device itself by driving the power supply.
- It provides security features such as cryptographic capabilities through a dedicated core embedded in the FPGA.
- More in general, the hardware supports the Intel AES-NI technology.

The Power Node architecture has been conceived thinking also to composability, in order to provide the possibility to build network of Power Nodes depending on the specific requirements of the specific application context. The Infiniband interface allows to create virtual 3D torus networks of Power Nodes, which are very efficient in terms of bandwidth and latency, and are capable of scaling up with no performance penalty. The torus network is managed by a network processor implemented in the FPGA of each Power Node, which interfaces to the system hub through two x8 PCI Express Gen 2 connections, for a total internal bandwidth of 80Gbs. Thanks to the FPGA implementation, the torus network processor permits standard, ad-hoc and application-dependent collective communications. Finally, the I-O and network interfaces are programmable, in order to permit interfacing the system to multiple network and bus technologies and protocols, increasing in this way the potential scalability of the network.

The possibility to aggregate multiple identical units has an impact also on dependability, providing redundancy. The execution segregation through hardware virtualization allows for protection, monitoring, disabling and replacement of malfunctioning or compromised nodes. Moreover, in case of a fault, redundant hardware provides dependable operations. This is accomplished at the hardware level through duplication of the resource and at a functional level through aggregation of resources (spare Power Nodes).

5.1.2.3 Power Node: software related aspects.

The software development for the Power Node could be mainly devoted to the adaptation of a commonly available Linux Distribution, in order to benefit from the richness of the features of a widely adopted operating system.

Regarding the OS the first choice could be “RedHat Enterprise Linux OS Verison 5.5 x86_64” which needs a license but is very well supported. Alternatively, if an open-source linux distribution is required, the Power Node can support linux distribution derived from RedHat, which are available for free but don’t have usually an excellent support. In this case, The operating system could one of the following:

- CentOS,
- Scientific Linux.

The development of specific drivers for the operating system will be mandatory to use all the peripheral available on-board.

Finally, a specific Software Development Kit (SDK) will be provided in order to allow the developer to fully exploit the potentialities and the functionalities offered by the hardware.

5.1.3 Nano, Micro and Personal nodes

The technology advancements in computing hardware and software enables a new generation of small Embedded System Devices (ESDs) to perform complex computing tasks. Extremely small sensor devices provide advanced sensing and networking capabilities. In parallel, many operating systems targeting these types of devices have been developed to increase their performance. The way for designing pSHIELD Nano, Micro/Personal Nodes is two folds:

1. To design completely new nano, micro/personal nodes that are complaint with the pSHIELD system design as described in Section 5.1.1.

2. To keep legacy technologies as they are, developed for many applications including those that are targeted in pSHIELD, which means to assume a heterogeneous infrastructure of networked ESDs like IEEE 802.15.4, IEEE 802.11, etc. An ordinary sensor technology (not all, since we need those that are designed for ES) permits to consider an augmentation of SPD functionalities at different levels of the hardware and firmware modules. This means an enhanced nano, micro/personal node with physical layer and protocol stack composed of existing and new SPD technologies. As result of this integration a new types of networked SPD ESDs will be created. This new SPD ESDs will compose a heterogeneous SPD network infrastructure too.

Developing a nano, micro/personal node equipped with some Legacy functionalities and with the pSHIELD Node Adapter, we obtain a composable pSHIELD Node. It means that it has all desired SPD functionalities and services for the pSHIELD application scenario selected. Additionally to that, the pSHIELD Node keeps almost all desired functionalities of a standardised sensor technology with additional SPD features that make it composable into the pSHIELD framework. The architectural design of the pSHIELD Nodes will relay on the ISO/IEC 9126 standard that has 6 top level characteristics: functionality, reliability, usability, efficiency, maintainability and portability.
The architectural design of the pSHIELD Nodes is not an easy architectural task since it requires to face many different constrains in the same times. Some of these constraints can converge in the same direction but some of them will be divergent and in the opposite directions. To cope with this challenge architectural design, as shown in section 4, the pSHIELD ESD use two approaches: (i) a Network approach and a (ii) Functional approach. The network approach constrains the architectural system design from network point of view. This approach should guarantee that all pSHIELD Nodes are part of a SPD Network that can be easily integrated with standard IP-based network like GSM, UMTS, etc. In other words it means that an SPD network is implementable and interoperable with standard networks to comply the main business cases of the application scenarios. The Functional approach constrains architectural design from the SPD requirements related to the node, network and middleware layers. The real innovation of pSHIELD is the introduction of the Overlay that makes the two approach converge.

5.1.3.1 Nano, Micro and Personal node description

Figure 5.5 provides a general view of the pSHIELD Nano, Micro/personal Node architecture. This is a generic model, which can be implemented in different architectures, providing different functionalities and different services, depending on the tasks to be accomplished by the node and the application field.

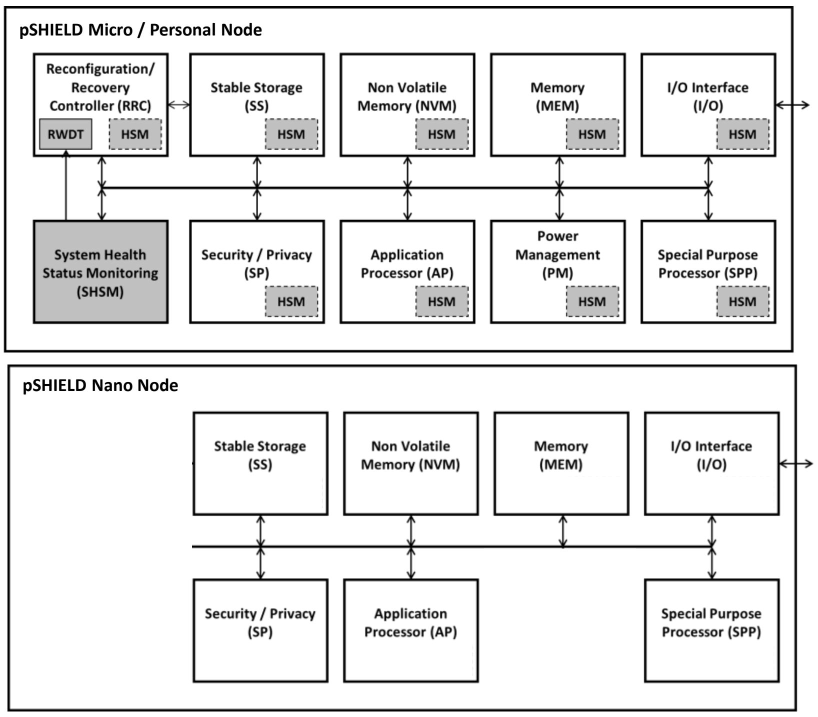

pSHIELD Micro / Personal Node
Reconfiguration/ Recovery Controller (RRC)
RWDT
HSM
Stable Storage (SS)
Non Volatile Memory (NVM)
Memory (MEM)
I/O Interface (I/O)
System Health Status Monitoring (SHSM)
Security / Privacy (SP)
Application Processor (AP)
Power Management (PM)
Special Purpose Processor (SPP)
pSHIELD Nano Node

Figure 5.5: Schematic view of SPD modules of a generic pSHIELD Nano, Micro and Personal Node

Health Status Monitoring and Controllers take care for different control functionalities. The hardware interface will cover the specification of the cryptographic hardware security blocks, high demanding level security performance, for example secure boot, secure time-stamping, and all necessary security management functionality such us device administration, key creation, and key import-export. Additionally, it defines the hardware interpreted data structures and direct interdependencies.

	High Security Level	Medium Security Level	Low Security Level
MEM	64 kByte	64 kByte	Optional
NVM	512 kByte	512 kByte	Optional
SPP Cryptography	AES-128 CCM, GCM, ECC	AES-128 CCM, GCM , ECC	AES-128 CCM, GCM, ECC
SP	AES-PRNG with TRNG seed	AES-PRNG with TRNG seed	Optional
AP	ARM Cortex-M3 32 bit, 50– 250 MHz	ARM Cortex-M3 32 bit, 50– 250 MHz	No
I/O Interface	Yes	Yes	Yes

Table 3: An example of the personal node SPD components.

From Figure 5.5 we can see that sensor, memory, radio and interface units are more or less standard modules for any standardized sensor technology. The multi-core processor will play a key role of the ES design with SPD features. For example, the memory can be realized by ROM, RAM, and FLASH, the radio can be 802.15.4, 802.11, RFID, UWB, etc., and the interfaces can be ADC and DAC, Timer, UART, I2C, etc. Finally, the multi-core processor can be realized as single microcontroller for nano and micro nodes, or more than one core microcontroller for personal nodes, and multi-core processor for power node (SPD & HSM, Application and Specific processors).
Today 3D integration nano-technology offers a new perspective for extremely complex heterogeneous System On Chip (SoC) design. Figure 5.6 shows hardware architecture of SoC design.

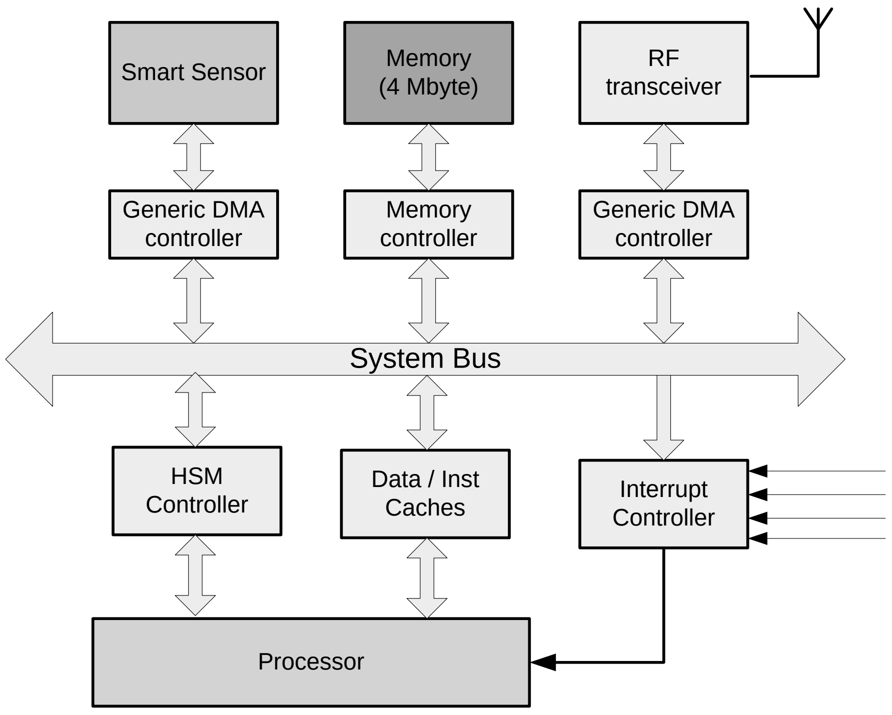


Figure 5.6: Hardware architecture and nano node chip partitioning.

To address the innovative issues and challenges in pSHIELD following solutions and **long-term objectives** are proposed for the development of pSHIELD Nano Nodes:

- A new hardware architecture described in Figure 5.6 is proposed based on two Innovative SPD components:

  − an intelligent low power smart sensor with on chip detection capability,

  − a digital image processing and communication chip based on optimized signal processor.

- Low power nano node with a target of less than 1mW by defining and implementing a multi-level power management strategy.
- Miniaturization through 3D Integration with special care for thermal study and electrical interactions between analog, digital and RF.
- Autonomous system working on a battery and communicating an optimized dataflow through wireless RF link.

For a typical multi-tasks software application running on a mono-processors architecture, (MIPS32 processor core with separated data and instruction caches). The ultra-low power processor chip can be designed by using VHDL (Very high speed integrated circuits Hardware Description Language). It contains also many digital peripherals like timers, watchdogs, interrupt controllers, HSM controller, UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), DMA (Direct Memory Access) controllers and interfaces/controllers to memories and cache memories. The design will also be performed taking into account the constraints of 3D integration.

5.1.3.2 Nano, Micro and Personal Node operating systems

Selection of the operating system for the demonstrator is an important design constrain, since we need to decide in which sensor platform will be realized SPD functionalities. The only requirement that we posed for this operating system is related to it possibility to be designed for embedded devices. There are two candidates for that: TinyOS and Contiki. Additionally, Hydra platform is a new concept that is realized in such a way that between physical and application layer is only a middleware.

5.1.3.2.1 TinyOS

This operating system (OS) is a free and open source operating system and platform that is designed for WSNs. It is an embedded operating system, written in the nesC Programing language as a set of cooperating tasks and processes. NesC is actually a dialect of the C programming language that is optimized for the memory limitation of the sensor networks. TinyOS features summary:

- No Kernel: Direct hardware manipulation.
- No Process Management: Only one process on the fly.
- No Virtual Memory: Single linear physical address space.
- No S/w Signal or Exception: Function call instead.
- No User Interface, power constrained.
- Unusually application specific H/w and S/w.
- Multiple flows, concurrency intensive bursts.
- Extremely passive vigilance (power saving).
- Tightly coupled with the application.
- Simulator: TOSSIM, PowerTOSSIM
- Written in “nesC” Language, a dialect of the ‘C’ language.

5.1.3.2.2 Contiki Operating System

Contiki is also an open source, highly portable, multi-tasking operating system for memory-efficient networked ESDs and WSNs. It is mainly designed for a microcontroller with small amount of memory. The key advantage of Contiki OS is its IP communications (both IPv4 and IPv6). It is flexible for a choice between full IP networking and low-power radio communication mechanisms. Contiki is written in the C programming language and consists of an event-driven kernel, on top of which application programs can be dynamically loaded and unloaded at run time. Contiki has been ported to different hardware platforms, such as MSP430, AVR, HC 12, and Z80. Contiki features summary:

- Event-driven Kernel: reduce the size of the system.
- Preemptive multi-threading support: an application library that runs on top of the event-driven kernel is optionally linked with applications that explicitly require a multithreaded model of computation.
- Simulator: COOJA
- Written in 'C' Language.

5.1.3.2.3 HYDRA middleware

The European Hydra project developed a "Middleware for Heterogeneous Physical Devices" with the aim to help manufacturers and systems integrators to build devices that can be networked easily and flexibly to create cost-effective high performance solutions. For the heterogeneous devices, sensors and actuators envisioned in the pSHIELD project, the large number of manufacturers and Universities are involved and the differences in their speed of innovation become an obstacle for the overall system design. Therefore, there is an urgent need for technologies and tools that make it easier to reap the benefits of networked systems. The complexity to build new technologies and tools grows exponentially with the number of devices, manufacturers and protocols involved. The Hydra middleware is a core technology that has a transparent communication layer, equally supporting centralized and distributed architectures. The Hydra middleware takes security and trust into account and allows to build model-guided web services. It runs on wired or wireless networks of distributed devices with limited resources. The embedded and mobile service-oriented architecture will provide fully compatible data access across heterogeneous platforms, allowing to create true ambient intelligence for networked ESDs. Adding extended security, privacy, trust and new dependability modules may satisfy requirements for having a middleware that will be SPD composable with the rest of the pSHIELD system architecture and network. The Hydra middleware consists of large number of software components – or managers – that handle various tasks needed to support cost-effective development of intelligent applications for networked embedded devices.

The biggest advantage of the Hydra middleware relies on the fact that allows developers to incorporate heterogeneous ESDs into their applications. This middleware can be incorporated in new and existing networks of distributed ESDs, which operate with limited resources: computing power, energy and memory. Additionally, Hydra-middleware provides easy-to-use web service interfaces for controlling any type of physical device irrespective of its network interface technology. Additionally, this middleware is based on a semantic Model Driven Architecture for easy programming and incorporate service discovery, P2P communications and diagnostic. In Hydra framework any physical devices, sensor, actuators or subsystem can be considered as a unique web service.

What we will need from HYDRA middleware for the pSHIELD SPD nodes? A lightweight version of this middleware, to be the Legacy Middleware Layer on top of which the pSHIELD middleware Adapter can host a set of Innovative SPD Functionalities: proper software modules must be added. This solution is in line with the recent IP stacks that are lightweight enough to run on tiny, battery operated ESDs. This is also in line with emerging application space of smart objects that require scalable and interoperable communication mechanisms that support future innovations as the application space grows. This strategy is also aligned with the future application scenarios "the Internet of Human and Things" (ITH). Smart objects are small computers with a sensor and actuator and a communication device, embedded in objects. To support the large number of emerging applications for smart objects, the underlying networking technology must be inherently scalable, interoperable, and have solid standardization base to support future innovation.

5.1.3.3 Specific SPD Considerations for Wireless Sensor Networks

The Wireless Sensor Networks (WSN) applications are used in many critical tasks, like aerospace, automation, monitoring environment, etc. Nowadays, these applications include new properties, such us security, dependability, privacy and trust. For WSNs applications to make security and dependability satisfaction is more and more important. In general WSNs are layered in 5 layers, but there are also other cases like Hydra network where we have three layers. In the pSHIELD project we follow the concept of four functional layers and based on that we are constructing a new type of network that we simple call it

SPD network. Heterogeneity of this SPD network is an extremely important feature of the pSHIELD network, since it allows existence of different type of Embedded System Devices on the Node Layer.
In general, the software part of WSNs can be layered into three levels: sensor software, node software and sensor network software. Sensor software has full access to sensor hardware. The output of a function of sensor software is used by sensor node software. This level includes system software for network maintenance and for some specific applications. For example, middleware resides over the operating system. Application programs use this middleware according to their own specific requirements. So, bottom layer consists of sensor, CPU and radio, on top we have operating system and on top of them Services and applications.
There are two approaches for sensor applications: (i) Service-oriented architecture (SOA) and (ii) Agent-oriented architecture (AOA). SOA is a design approach that defines the interaction among architectural elements in terms of services that can be accessed without knowledge of the underlying platform implementation. AOA proposes an infrastructure that applies active agent technology to WSNs, because the network must be dynamically configurable and adaptive in order to response actively to events where security and dependability must be built into WSNs at the early design stage. The pSHIELD solution leverage this two approaches investigating a hybrid solution where the SOA is applied by the pSHIELD Middleware Adapter and AOA is applied by the Security Agents operating in the pSHIELD Overlay.

5.1.3.4 SPD models for Nano, Micro and Personal nodes

Triverdi et al. present a classification of dependability and security model types: combinatorial models, state-space models, hierarchical models, fixed point iterative model, simulation, analytic and simulation, and hybrid model, that can be applied for the presentation of dependability and security models [29] . For extremely difficult models analytic and simulation can be used in combination with hybrid models. Development of new SPD Embedded System Devices requires careful approach and consideration of a variety aspects that are influencing our design methodology. Dependability and security models are developed almost independently in the area of small networked sensors. Based on the recent paper publish by Triverdi et all, called Dependability and Security Models[13] we have a solid background for modelling security and dependability for SPD ESDs.

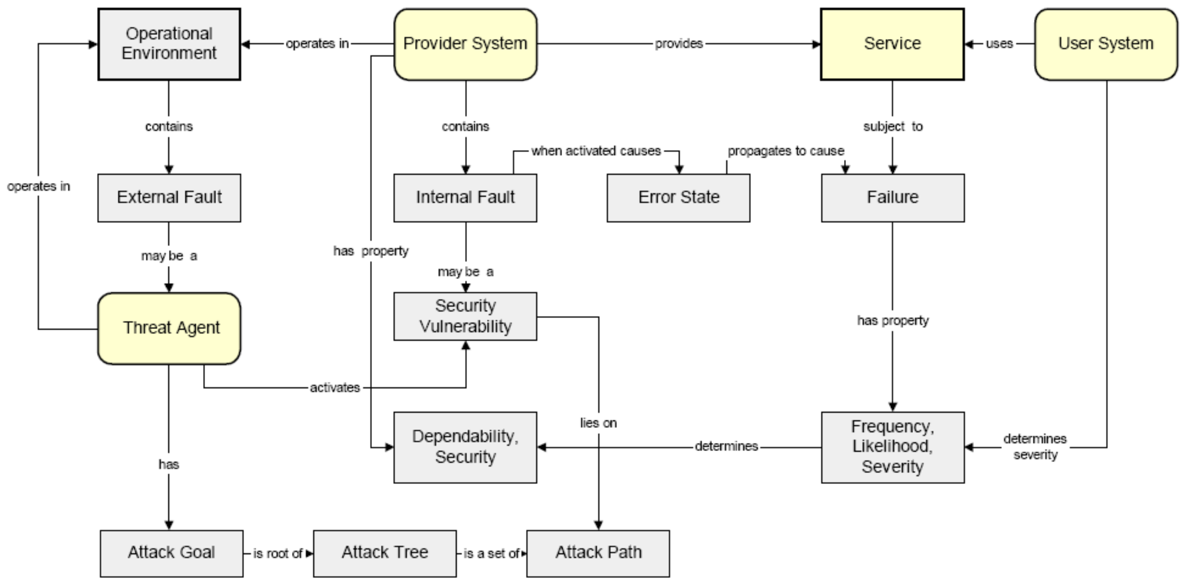


Figure 5.7: Concept model for security and dependability[14].

Security is a property of a system or service. Software-intensive systems are complex, meaning that they are composed of many components of different types which interact with each other to create properties not exhibited by the individual components. The purpose of the system is implemented as the *service* the system, acting as a provider, delivers to another system, the user system. A particular service can fail in a

[13] Kishor S. Triverdi et all, “Dependability and Security Models,” 7th International Workshop on the design of Reliable Communication Networks, DRCN 2009, Washington Dc, October 2009.
[14] John Murdoch, “Security measurements,” White paper, 2006.

variety of ways, resulting in dependability being a composite property, covering the following more specific properties (more of the property is indicative of fewer or absence of the corresponding failures). Dependability and security overlap in the sense that some types of failure fall under both properties.

The definition of dependability and security as the ability to avoid failures raises the question of how a system or service can be measured with regard to such ability. Before addressing this question, we need to define a model of how a service failure is caused.

Many types of fault of concern to security system solution are similar to safety faults. For example, events in the natural environment, accidental, non-malicious actions during development etc. However, security has an additional type of fault arising from the presence of malicious threat agents in the operational and development environment. Such agents can learn and adapt, resulting in evolving external faults. Attack trees can be used to map the objectives of a threat agent onto vulnerabilities of the system. Alternative attack sequences represent the possible ways the agent might achieve his/her goal. Development and operational policies can be adjusted to prioritize defensive actions. Measurement can support the decision making involved, for example in the estimation of the cost to a threat agent of different attack sequences. Under certain assumptions, an increase in attack cost would imply a lowering of the likelihood of the attack sequence occurring and an increase in security with regard to the associated service failure.

5.2 Cryptographic algorithms

5.2.1 Description

Railways have been an important means to freight transportation given the economic and efficient nature of moving bulks of goods by trains directly to their destinations. The scenario proposed by pSHIELD (described in section 5.8) mainly aims at monitoring hazardous materials transported by trains in carriages equipped with a wireless sensor network, devices for intrusion detection and access control, and a location-aware communication transceiver.

Having a wireless monitoring system installed per train carriage increases the need for security. It is essential, given the hazardous nature of the transported materials, to eliminate any risk of harming human lives or the environment through malicious attacks. To cope with such a security requirements some Node pSHIELD specific SPD Components (i.e. Functionalities) will be investigated to develop advanced cryptographic algorithms.

Cryptography has been an established means for many years to provide security and information protection against different forms of attacks. It is seen as the basis for the provision of different systems security, fundamentally by seeking to achieve a number of goals, that are; confidentiality, authenticity, data integrity and non-repudiation. However, with the inclusion of Embedded System Devices such as sensors in the system, different concerns arise regarding the suitability of conventional PC or Internet cryptographic schemes. This is the case given the resource constrained nature of those devices, overall system architecture, possible heterogeneity, etc. Where cryptography can be typically seen as either ciphering through an asymmetric or a symmetric algorithm, different aspects such as key management and authentication are involved.

As previously mentioned, the constrained resources of some of the system's components put serious limitations on the range of the available cryptographic primitives that can be used to secure it. Choosing the right cryptographic technology or technologies to address security will be made by a combination of research and testing different approaches and technologies in order to choose the best for the specific problem, taking into account power consumption, performance and security assurance regarding specific cryptographic tasks, such as signing, verification, encryption and decryption. Once the research and test phases are over, the chosen approach will be prototyped, thus allowing integration into the respective demonstration environment.

The main target in pSHIELD for security improvements on the small ESDs is related to development of efficient cryptographic techniques that are well suited with the Nano, Micro and Personal nodes constrains (limited power, memory, processing capabilities, etc,) and in the same time to offer strong security solutions for the most known threats, for example in sensor technologies and Wireless Sensor Networks (WSNs). In literature we can find many research activities in this field and there are many symmetric key algorithms which are tested to be very secure, but the biggest problem with this kind of system is that the key has to be shared over a public network and in most cases it is pre-deployed on each sensor node. Asymmetric key cryptography solves problem which cannot be resolved by use of symmetric-key cryptography. For this kind of cryptography two keys are generated: one private and public key. The private key is kept secret while the public key is made public. The message is encrypted using the public key and the private key is used to decrypt the message. Another wide usage of such technique is utilization of private key to sign the message, while the public key is used to verify the signature at other end. The biggest problem associated with use of asymmetric cryptographic system is that it is slow and expensive. The major challenge for the researchers in the field of WSNs is to reduce the computational complexity, and minimize memory usage of the traditional asymmetric cryptographic algorithms like Al-Gamal, RSA, Digital Signature Algorithm (DSA) and Elliptic Curve Cryptography (ECC). Today, the ECC is considered to be most suitable small ESDs like sensors, because its memory and resource consumption is least as compared to all others. The current state of the art (SoA) of the important security solution along with their relevant information concerning pSHIELD security is shown in the following table:

	Year	Language sed	Security Property Implemented	Algorithms
SPINS	2002	N/A	Data Confidentiality, Integrity, Authentication and freshness, authentication	
TinySec	2004	NesC	Access Control, Integrity, Confidentiality, Replay Protection	Skipjack CBC-CS mode
SenSec	2005	NesC	Access Control, Integrity, Confidentiality, Key Management	Skipjack-X CBC-CS mode
MinSec	2007	NesC	Pre-Deployed symmetric key, Confidentiality, Replay Protection, authentication	Skipjack OCB mode
TinyECC	2007	NesC	Key Exchange, Public Key Encryption, Digital Signature	ECC SECG-160
ContikiSec	2009	C	Confidentiality, Integrity, Authentication (CIA concept)	AES CBC-CS mode

Table 4 Development of security solutions for WSNs

The security level between different cryptography technologies is proposed for comparison how efficient security protection is. For that the concept of work factor is defined and is measured in bits of security. It describes the amount of work the fastest currently available attack would require on the algorithm with the specific key. The fastest attack on an algorithm with N bits of security would require 2^N calculated steps. The following table shows a comparison between RSA and ECC cryptography. This table clearly shows that ECC gives much faster performance and delivers higher security bits.

Security Level	Key size (bits)	Decryption time (sec)	Verification time (sec)
80	RSA-1024	2.694	0.191
	ECC-160	0.765	1.042
112	RSA-2048	14.734	0.665
	ECC-224	1.187	1.626
128	RSA-3096	44.274	1.378
	ECC-256	1.375	1.905

Table 5 RSA and ECC security level comparison

ECC is a simple but time consuming process. The researchers have worked on many mathematical methods to reduce the operational time taken to perform such calculations. There are many ECC optimizations that can lead to better performance and resource utilization.

For the pSHIELD project is important to investigate how the reconfiguration possibilities of the system can be used to adapt ECC parameters in order to increase or reduce the security level depending on the application scenario or the energy budget. Additionally, the security is not isolated and will be considered jointly with dependability and privacy issue for further optimisation of the SPD node performance. In particular specific asymmetric cryptography algorithms, such as elliptic curve cryptography (ECC), have been designed and will be implemented thus pushing strong asymmetric cryptography to low cost nodes in pSHIELD Embedded System Devices, which has for a long time been doubted to be feasible at all.
Regarding key management schemes, specific activities will be addressed to implement and deploy a new cryptographic key exchange algorithm called **Controlled Randomness** that limits the need of frequent key exchange between the communicating parties, thus boosting the robustness of the underlying cryptographic operations against cryptanalysis, while keeping the performance cost in acceptable, and in some cases favourable, levels.
We focus on the problem of key replacement and exchange. Periodic changes of cryptographic keys is a necessary operation as to ensure (i) minimal exposure of plain data in case of key compromise and (ii) minimal collection of encrypted data under the same key, as to harden cryptanalytic attacks. Designers of secure systems rely on the fact that keys are periodically changed, in order to maintain the high level of

security offered by the cryptographic primitives. Key replacement operations define the rekeying period for an algorithm (how often the key should be changed) and the behaviour of the system during transition periods where both an old and a new key may be valid. This can happen for example in a loosely synchronized environment, where some clients can receive out-of-order packets from the servers.

An embedded system can incur an interesting trade-off on security level and resource consumption. From a security point of view, the keys must be often refreshed, as explained earlier, in order to maintain the required security level. From a system resource consumption point of view, the keys must be rarely changed, in order to minimize the consumption of precious resources (processor, power, and bandwidth). Further, in some usage scenarios, advanced care must be taken in order to ensure that the new keys will be available by the time they must be used, especially when only intermittent connectivity exists.

The approach we propose has three desirable properties for protecting against cryptanalytic attacks: (i) the attacker must find which packets are encrypted under the same key, in order to then mount a specific cryptanalytic attack, (ii) consecutive packets that may carry redundant information are encrypted under a different key, and (iii) the disclosure of one key does not reveal the contents of a whole session; even if an attacker can know the time moments that the compromised key is used, he cannot resemble a whole session but only sketchy details of it.

These properties allow to extend a key's lifetime, both in terms of time of use and volume of data encrypted under the same key. This extension of lifetime is desirable in many environments, since it achieves a higher level of security utilizing the same cryptographic primitives, it requires less frequent control messages to be exchanged, and it requires fewer management operations.

The concept of controlled randomness i.e., having multiple active keys at any given time moment, offers superior security characteristics compared to conventional protocols. The system designer can reuse well-known cryptographic blocks in an novel way to achieve increased security with minimal hassle

It has been already shown that controlled randomness leads to efficient implementations in a variety of consumer embedded systems. The goal is to apply this knowledge to the pSHIELD concept and refine the model through its practical implementation within the scope of the demonstrator.

Another possible approach towards better security and trust computing is to equip each pSHIELD Embedded System Device with a Trusted Platform Module (TPM). However, this is an impractical solution due to the cost, size and energy constraints of a battery-powered Nano, Micro and Personal node. These constrains suggest to explore a tiered network architecture for ESDs, wherein only a small number of ESDs are equipped with TPMs – for example, these could be the those ESDs that form the backbone or infrastructure network, or platforms with higher computation and communication capabilities: our goal is to achieve this at least for Micro and Personal ones. For such architecture it is possible to analyze how the system can leverage the presence of some ESDs equipped with TPMs to perform the critical steps of cryptographic key establishment, distribution, and management. It is important also to explore comprehensive framework on the recent Elliptic Curve Cryptography (ECC) research results and to compare this with symmetric cryptographic keys between pSHIELD SPD ESDs by utilizing the hardware-based key generation facilities of the few TPM-enabled pSHIELD SPD ESDs in the pSHIELD network. Additionally, TPM architecture should be included in such a way that overall SPD level of the system is optimized with respect to security metrics. Since TPM implements RSA, it is extremely important to investigate possibilities of using ECC security solution due to scared resource of Micro and Personal nodes. There is also another possibility to reduce the key calculation time to meet the potential applications, in particular for WSNs. It is well known that scalar multiplication is the operation in elliptical curve cryptography takes most of the time for the key calculation time on small sensor nodes. Looking for faster algorithms is also a key direction in the pSHIELD project.

5.3 Smart SPD driven transmission

5.3.1 Description

In the past few years, the impact of mobile devices in everyday life is continuously increasing as well as the development of advanced and accessible wireless communication services supporting users in their common activities. Together with new wireless communication technologies and standards, designed to meet the users' demand of high data rate Network Layer services, new mobile and smart Embedded System Devices (ESD) have been proposed exploiting the advances in signal processing techniques and in hardware platforms.

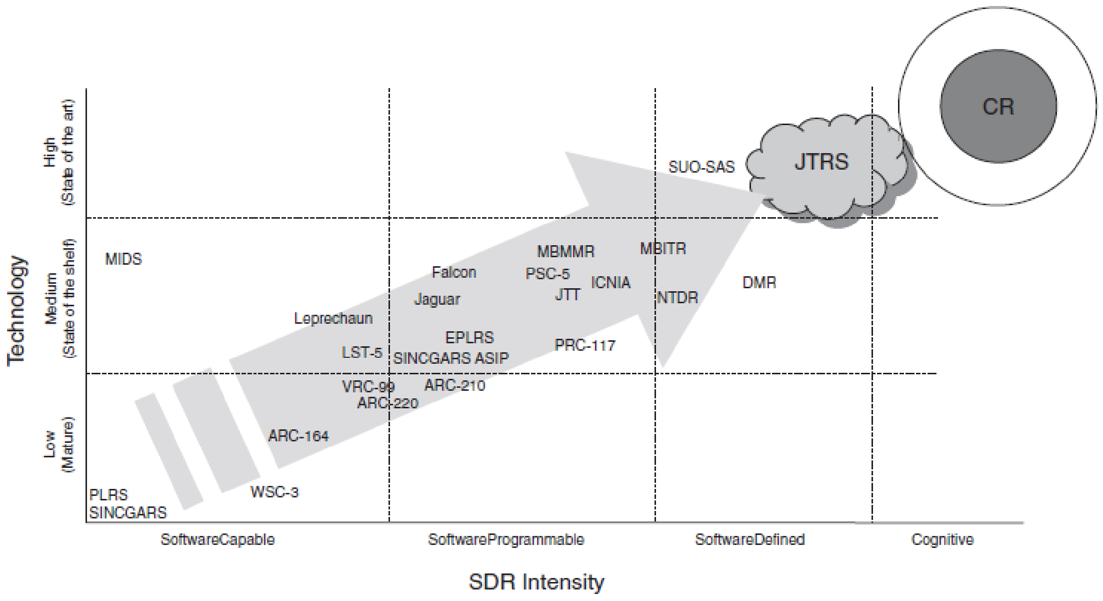


Figure 5.8: Evolution of the radio network technologies for Embedded System Devices: from traditional systems to Cognitive Radios.

Historically, radio Embedded System Devices were designed to perform a single, given task. On the contrary, nowadays, it is required that many wireless services (e.g., Wi-Fi connectivity, global positioning system) can be accomplished on a single Embedded System Device, reducing life cycle costs. To this end, software was added to the system designs to increase capability and flexibility, as shown in Figure 5.8. This innovative Network Layer service, derived by the introduction of software programmable components, is usually known as Software Defined Radio (SDR). Therefore, as an example, it allows to accommodate new standards and new Network Layer services as they emerge upgrading the terminal software without requiring to develop a new dedicated Embedded System Device.

Recently, the research activities on Network Layer functionalities of Embedded System Devices are focusing on the development of intelligent, Cognitive Radio systems capable to understand and to be aware of the surrounding environment allowing to exploit all the available wireless network services by using a single Embedded System Device. As an example, a Cognitive Radio (CR) could learn services available in a locally accessible wireless computer networks, and could interact with those networks by using its preferred protocols, so the users would not have confusion in finding the most suitable connection for, as an example, a video download or a printout. Additionally, a Cognitive Radio could select the carrier frequency and choose the transmitted waveforms according to the perceived environment and to reach a given goal, e.g., to avoid interference with existing wireless networks or to maximize the throughput while guaranteeing an acceptable Quality of Service (QoS).

5.3.2 Formal conceptual model

In the context of pSHIELD project a set of innovative enabling technologies will be investigated in order to apply a Smart SPD driven transmission service in the pSHIELD Network Adapater. The main focus will be given to the study of enabling technologies for the Cognitive Radio paradigm.

According to Haykin's definition[15], a Cognitive Radio is an intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), and uses the methodology of understanding-by-building to learn from the environment and to adapt its internal states to statistical variations in the incoming Radio-Frequency (RF) stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time, with two primary objectives in mind: (i) highly Reliable and Dependable communications whenever and wherever needed and (ii) efficient utilization of the radio spectrum. As it is clear from this definition, the common keywords for an efficient Cognitive Radio are **Awareness** and **Reconfigurability**.

In a radio environment, Awareness means the capability of the Cognitive Radio to understand, learn, and predict what is happening in the radio spectrum, e.g., to identify the transmitted waveform, to localize the radio sources, etc. Reconfigurability is necessary to provide self-configuration of some internal parameters according to the observed radio spectrum. It is enormously important for both civilian and military applications especially when unforeseen situations happen and some Network Layers services are not available, guaranteeing trusted connectivity. It is now abundantly clear that the cognitive radios and their capabilities of dynamically maintaining a reliable and efficient communication can be significantly relevant in Security, Privacy and Dependability (SPD) driven applications where it is necessary to dynamically guarantee a high level of trustworthiness.

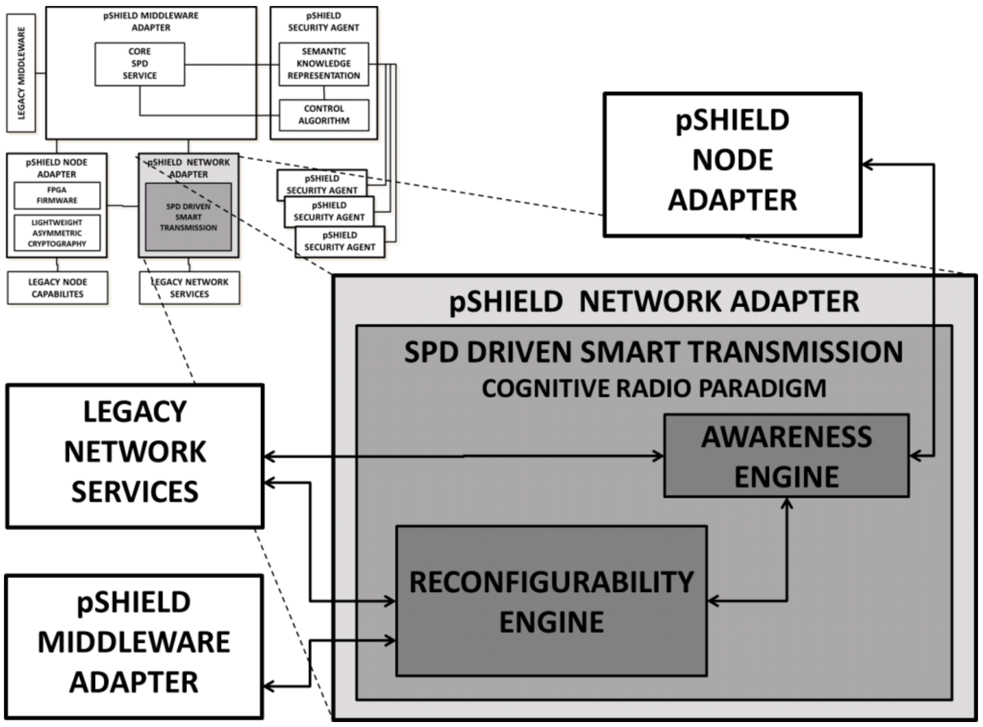


Figure 5.9 pSHIELD NETWORK ADAPTER conceptual model

As depicted in the above figure, according with the pSHIELD functional component architecture defined in Section 3.1.1, the Smart SPD driven transmission is obtained applying a Cognitive Radio paradigm based on two main functional components: the Awareness Engine and the Reconfigurability Engine.

In particular, some Node Layer capabilities (e.g., antennas, cameras) and Network Layer services (e.g. available network resources, radio spectrum, number of active users, transmission protocols and

[15] Simon Haykin, "Cognitive Radio: Brain-Empowered Wireless Communications". IEEE Journal on Selected Areas in Communications, vol. 23, no. 2, February 2005

standards, localization, etc.) can be used by the Awareness Engine to acquire a context awareness of the current radio environment. Then, some reasoning capabilities of the Cognitive Radio provided by the Reconfigurability Engine can be used to select the most appropriate configuration parameters useful to guarantee the needed SPD transmission.

It is important to note that, the Cognitive Radio capabilities are enormously attractive in a wide set of applications for both civilian (e.g., reliable communications, increased data-rate) and military (e.g., detect and decode enemy transmissions) scenarios. Although some encouraging preliminary results have been obtained in some practical environments, some open issues still remain open and to obtain a general and multi-purposes cognitive radio is an open research problem. In general the main research challenges in this domain can be reduced to the following:

- Obtain a precise and concise representation of the radio environment (e.g., available resources, number of active users, transmission standard used, source localization) by using some Node Layer and Network Layer information;
- Define the optimal configuration of the Network Layer according to a given goal (e.g., SPD metrics) and the perceived radio environment;
- Develop algorithms and techniques for providing the capability of learning from the experience in order to face unforeseen and unexpected situations (e.g., a malicious user), that means the Embedded System Device's Network Layer is equipped with cognitive capabilities.

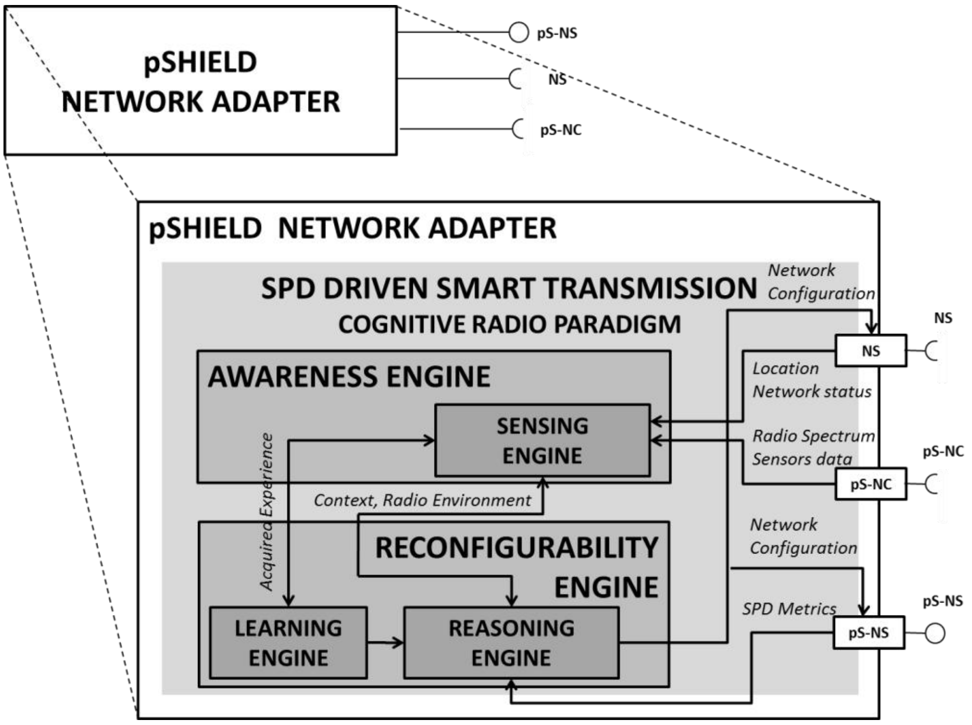


Figure 5.10 pSHIELD Network Adapter component architecture with some examples of exchanged information

In order to clarify how the main research challenges are mapped in the overall pSHIELD component architecture described in Section 3.1.1a more detailed component architecture of the pSHIELD Network Adapter has been depicted in Figure 5.10. One of the most important aims that a cognitive radio has to perform is to obtain a SPD communication whenever and wherever (e.g., in an unknown environment). This task is also known in the open literature as *Spectrum Sharing* and its main objective is to enhance the utilization of the radio spectrum, exploiting the unused resources, without causing harmful interference to existing active users in the monitored environment, while guaranteeing an acceptable level of Network Layer services.

To this end, the Awareness Engine has to gather information from the Legacy Network Services (through the NS interface) and from the pSHIELD Node Adapter (through the pS-NC interface), to acquire radio awareness (i.e. a precise and concise representation of the radio environment). For this purpose the Representation Engine collects all the needed information (radio spectrum, raw data from sensors, available network nodes and their status, etc.) and evaluate a precise and concise representation of the radio environment (i.e. the radio context). These algorithms and techniques which allow to obtain the radio awareness are known as *Spectrum Sensing*. For example, the Representation Engine can detect the active users in a monitored area, identify the used transmission standards, localize the radio sources, etc. This information is then provided to the Reconfigurability Engine to select a proper system Network Layer configuration according to a predefined goal of the Cognitive Radio and the surrounding environment.

To be compliant with the Cognitive Radio paradigm, the Awareness Engine should face even unknown radio contexts and react properly. In others word the Awareness Engine should be able to learn from the experience to face unforeseen and unexpected situations. This task is performed by the Learning Engine, that apply algorithms and techniques for providing cognitive capabilities (i.e. learning from the experience). The outcomes of Learning Engine is a set of parameters that represent the acquired experience useful to configure optimally the Representation Engine and the Reconfigurability Engine.

The Reconfigurability Engine is in charge to identify the optimal configuration of the pSHIELD Network Layer according to a given goal (e.g., the SPD metrics provided by the pSHIELD Middleware), the perceived radio environment (provided by the Representation Engine) and the acquired experience (provided by the Learning Engine). As an example, given a specific radio context, the Reconfigurability Engine can be able to derive an new, optimal Network Layer configuration to establish a communication with an already active Embedded System Device by using its preferred transmission standard, carrier frequency and transmission power. This task is also known as *Opportunistic Communication* and allows to identify and exploit an “opportunity” to establish a communications with the other players in a given context according to the surrounding environment conditions and the SPD goals.

5.4 Semantics models

5.4.1 Description

The semantic models in pSHIELD shall provide an effective way to exploit the benefits of the explicit knowledge of the application domain by means of a coherent and formal representation of SPD technologies in Embedded Systems.

As a matter of fact, pSHIELD shall leverage the formal and explicit representation of the knowledge in order to make automatic processing (reasoning) on it possible.

Semantic models in pSHIELD lean on the concept of *ontology*, and ultimately shall enable interoperability at different levels in the conceptual framework of pSHIELD.

Given that the SPD domain in Embedded Systems shall be captured by a number of ontology, automatic reasoning is enabled in order to support several features of the pSHIELD framework.

Broadly speaking, a semantic engine (reasoner) shall enable interoperability within Middleware Layer and rule based discovery and composition within Overlay Agents, in such a way to provide the following essential enabling mechanisms:

- Semantic reasoning based on ontology models may carry out a reconciliation of heterogeneous formats of parameters exchanged between different layers (also suitable for interaction with legacy agents).
- The semantic characterization of the behavioural aspect of components makes it suitable for an agent to determine "what the service does".
- The semantic characterization of the composition of functionalities and of the relations among them makes it suitable for an agent to reason about SPD metrics of the current configuration and – if needed - to carry out reconfigurations of the system at run-time, by means of rule-based combination / composition of components and SPD technologies, in order to achieve the new intended values for SPD metrics.

The possibility of removing semantic clashes caused by the intrinsically heterogeneous set of entities making up a a complex system appears as an enabling technology to get over different implementations that underlie an embedded system.

Discovery and composition functionalities shall be discussed to a greater extent in the section Core SPD Services.

5.4.2 Formalized conceptual model

5.4.2.1 Semantic reconciliation model

When nodes have no semantic capabilities due to computational / power limits, semantic reasoning based on ontology model may carry out a reconciliation of heterogeneous formats of parameters exchanged between different layers (also suitable for interaction with legacy agents).

With regards to differences between formats and structures, two types of semantic clashes, i.e. mismatches, are detected in the pSHIELD framework:

- *Loseless*: can be solved without loss of information
- *Lossy*: any (faithful) transformation will cause a loss of information

Here is a list of some identified mismatches for each category

- Loseless mismatches
  - Naming: different labels for the same content
  - Attribute granularity: the same information is split in a different number of attributes
  - Structure Organization: different structures and organization of the same content
  - Subclass-Attribute: an attribute with predefined value set is represented by a set of subclasses

- Schema-Instance: data hold schema information
- Encoding: different format of data or unit of measure

- Lossy mismatches
    - Content: different content denoted by the same concept (typically expressed by enumeration)
    - Coverage: the presence/absence of information
    - Precision: the accuracy of information
    - Abstraction: level of specialisation refinement of the information

Clash removal can be performed by means of a sort of *rules template* that let us reconcile, i.e. remove, the clashes between a Source and Destination that exchange messages.

The formal model for clashes removal relies on a *semantic hub* (ultimately a semantic reasoner) acting a decoupling between interacting entities, based on a set of *ontologies* and reconciliation *rules* stored in a *semantic database*.

A unified conceptual model, namely the *knowledge repository* (shown in Figure 5.11), can be provided, linking together the semantic hub and semantic database functionalities, where the ontologies and related rules are stored.

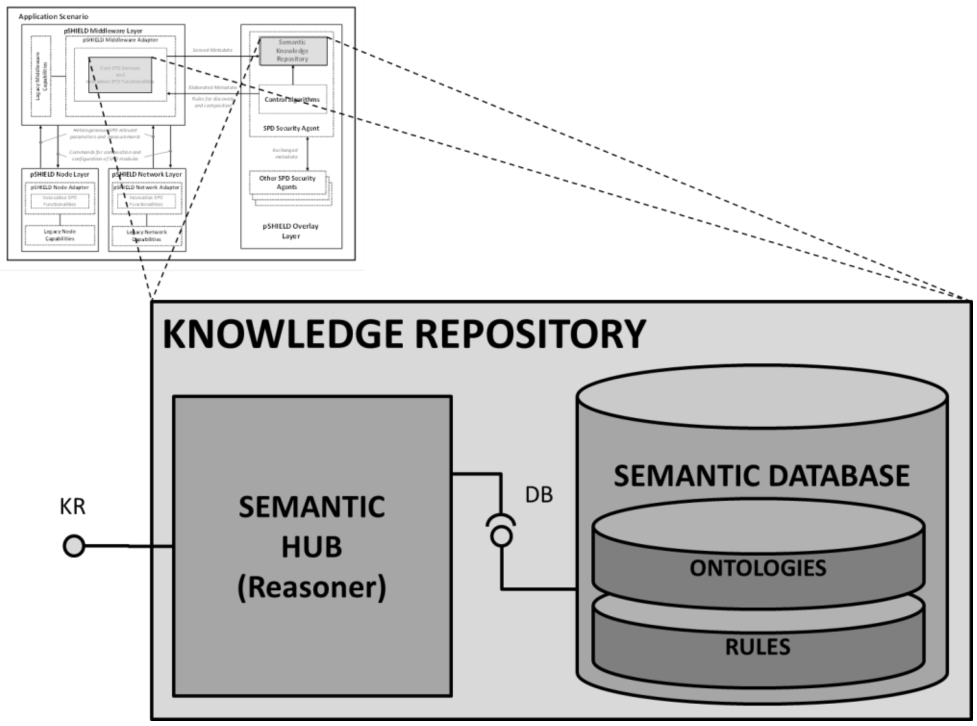

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Exchanged metadata
Other SPD Security Agents
pSHIELD Overlay Layer
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
KNOWLEDGE REPOSITORY
KR
SEMANTIC HUB (Reasoner)
DB
SEMANTIC DATABASE
ONTOLOGIES
RULES

Figure 5.11 Knowledge Repository conceptual model

As shown in Figure 5.11, the Knowledge repository model is located in the pSHIELD Middleware Adapter as well as in the pSHIELD Overlay. This is because the use of semantic (as already explained in section 4.3) enables the pSHIELD seamless approach to SPD. The enabler of the seamless approach is the semantic-aware framework, which is located (within the pSHIELD framework) both in the pSHIELD Middleware Adapter and in the pSHIELD Overlay. The Knowledge repository is composed by a semantic reasoner and a semantic database interacting each other by means of the DB interface. The semantic hub accomplishes the clashes removal according to the following procedure:

- Rules for the mapping between specific (local) models and shared model are created according to one or more suitable templates and stored in the semantic database;
- Mapping always flows from entities to the hub and viceversa without regards to the actual information flow at run-time (that is, no need to hard-wire every tuple of entities, the reasoner will act as a hub in the information exchange);

- At run-time, information shall flow from other semantic-aware components to the semantic hub (by means of the KR interface shown in Figure 5.11), that will take care of dispatching the data in the reconciled format to the right recipient(s) according to the rules fired by the data itself.

5.4.2.2 Formal procedure for pSHIELD ontology formulation

Given a clear procedure on how to build ontology like the one previously described, it is necessary to identify what it is supposed to be described? Starting from that, we can affirm that the context is the one of Embedded Systems; in particular the more specific contest are the SPD functionalities provided by their interaction/composition. The main objectives of the approach with semantic models are:

1. the *abstraction* of the *real word* from a technology-dependent perspective into a technology-independent representation.
2. the *representation* of *functional properties* by means of ontology as well
3. the identification of the relations between real/structural and functional world.

So, as depicted in Figure 5.12, the problem of modelling SPD in the context of ES is reduced to the formulation of three different meta-models describing: i) structure, ii) functions, iii) a relations between structure and functions.

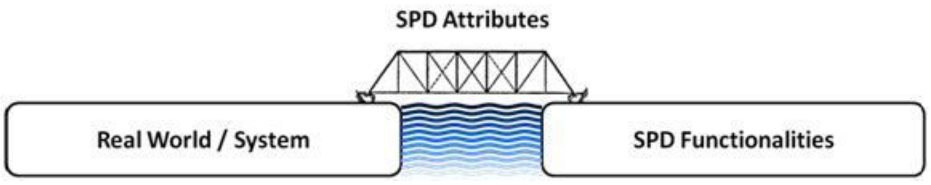


Figure 5.12 Proposed approach to model SPD for ES

In particular the bridge has been built thanks to the introduction of a third metamodel taking into account the atomic attributes that are impacted in this context and to map them in these two worlds. For each of the three models, a justification is provided below:

5.4.2.3 Structural Ontology

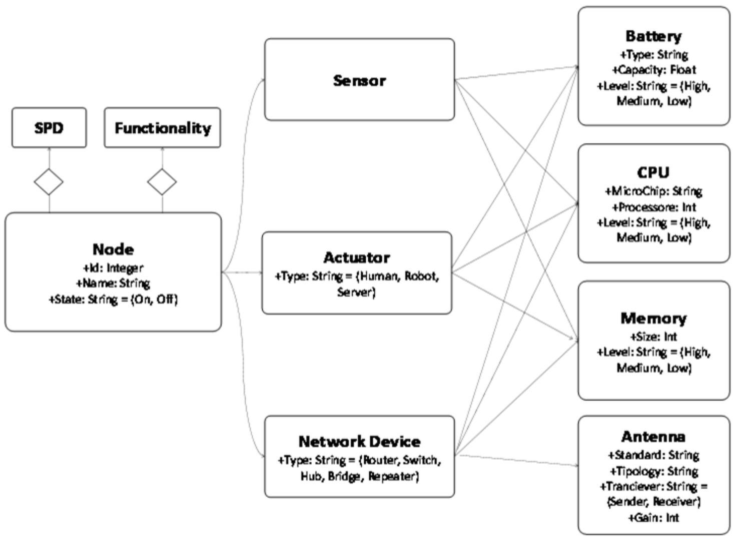

SPD
Functionality
Node
+Id: Integer
+Name: String
+State: String = (On, Off)
Sensor
Actuator
+Type: String = (Human, Robot, Server)
Network Device
+Type: String = (Router, Switch, Hub, Bridge, Repeater)
Battery
+Type: String
+Capacity: Float
+Level: String = (High, Medium, Low)
CPU
+MicroChip: String
+Processore: Int
+Level: String = (High, Medium, Low)
Memory
+Size: Int
+Level: String = (High, Medium, Low)
Antenna
+Standard: String
+Tipology: String
+Tranciever: String = (Sender, Receiver)
+Gain: Int

Figure 5.13 Structural Ontology for a node

The structural ontology (on the left side of Figure 5.12) is the easier to model, because it is a simple description of the Embedded System component. It contains the hardware components and basic functionalities provided by the individual element and the related attribute, all in an SPD relevant environment. For example a node, in a first simplification, is composed by a memory a CPU, a battery and a transmission antenna; furthermore the CPU is characterized by the frequency and bit length and, in SPD relevant context, the possibility of performing hardware cryptography. By doing so, all the components constituting a complex system can be represented. In Figure 5.13 an example of Node model is provided.

5.4.2.4 Functionality Ontology

In order to identify the functionalities provided by the systems components, a preeminent approach based on the Security Certification Process (Common Criteria) has been chosen. This approach is based on the five-steps reported in the following:

- ***Step 1 What is the overall system?***

The overall System is a set of interacting and interconnected Embedded Systems with specific composability and SPF Functionalities.

- ***Step 2 What is the role of the System?***

The system, in SPD relevant context, should assure SPD for a certain asset or goods in a specific scenario. In the following, for convenience, we will replace the expression “assure SPD” with the generic expression “protect”, even if its real meaning is different. In Figure 5.14 this concept is represented: the green boxes represent the interconnected ESs and the gray box is the addressed asset/goods (the SPD functionalities are still missing from this graphical representation because they are identified in the following steps).

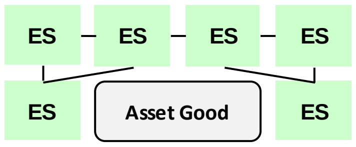


Figure 5.14 Step 1 and 2 representation

- ***Step 3 What are the asset/goods and the scenario addressed by the System?***

The selected scenario (for the purpose of this work) is railways transportation and the asset/goods is the secure and dependable monitoring of freight trains transporting hazardous materials. Moreover, since the System should protect itself, it is an asset/good as well.

- ***Step 4 What are the possible menace and/or attack that could affect the protected asset/goods.***

Once the assets and goods, as well as the application scenarios are clearly identified, it is easy to enumerate all the possible menaces and attacks that could affect the level of Security, Privacy and Dependability of the system. The output of this activity is a fundamental input for next step. The logical step is given in Figure 5.15.

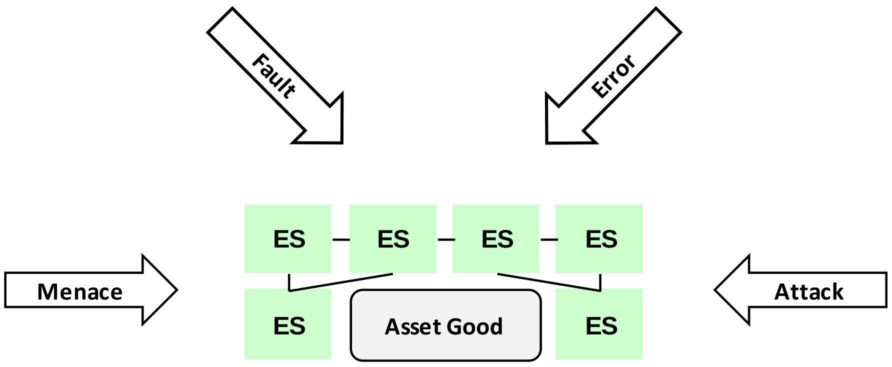


Figure 5.15 System attacks and menaces

- ***Step 5 What are the SPD functionalities that can prevent or minimize the effect of the previously identified menace and/or attack?***

Starting from the identified menaces and attacks, a set of SPD Functionalities is identified that are able to prevent or mitigate them. The functionalities are the ones that we have to represent in our SPD relevant ontology. An example is provided in Figure 5.16. Of course, the node model has relation with the functional ontology.
This ontology represents the semantic description associated to the pSHIELD SPD component (see section 4.4.2.1).

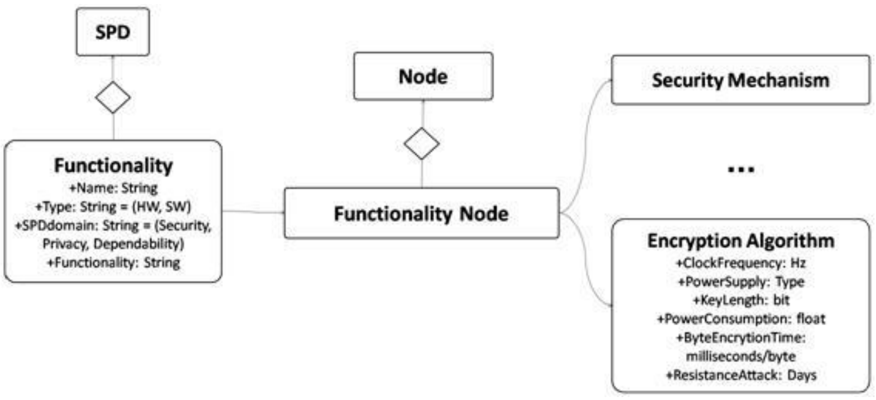


Figure 5.16 Functionality Ontology

5.4.2.5 SPD Ontology

The last model is given by the SPD attributes that allow the link between the structural word and the functional word. This is the most simple and, at the same time, significant ontology. For the purpose of our work we have choose to describe all Dependability, Security (and Privacy) issues by means of six attributes: availability, reliability, safety, confidentiality, integrity, maintainability (see Figure 5.17).

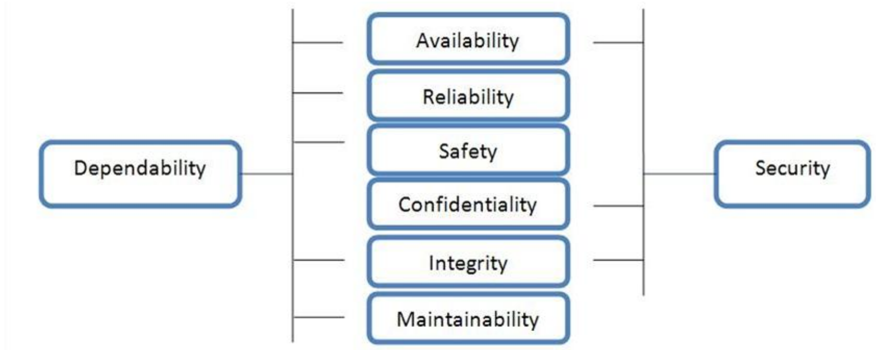


Figure 5.17 SPD attributes Ontology

Each structural and functional component is associated to one of more of the following attributes: for example the cryptography could be related to Confidentiality and the Node's CPU is related to cryptography. Then we know that if we want to address the confidentiality issues of our system we have to take into account the cryptographic algorithms as well as the CPU characteristics.
The example is trivial, but in more complex system the advantage is evident: we have an high level representation by means of ontology that, thanks to inferential engines, can provide us a list of all relations and components relevant to our SPD issues and objective.

5.5 Core SPD services

5.5.1 Description

The core SPD services are a set of mandatory basic SPD functionalities provided by a pSHIELD Middleware Adapter (introduced in section 4.1) in terms of pSHIELD enabling middleware services. The core SPD services aim to provide a SPD middleware environment to actuate the decisions taken by the pSHIELD Overlay and to monitor the Node, Network and Middleware SPD functionalities of the Embedded System Devices under the pSHIELD Middleware Adapter control. The following core SPD services are provided:

- secure service discovery;
- service composition;
- service orchestration;

Secure service discovery allows any pSHIELD Middleware Adapter to discover in a secure manner the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the pSHIELD Embedded System Device (pS-ESD) where it is running. Indeed the pSHIELD secure service discovery uses a variety of discovery protocols (such as SLP[16], SSDP[17], NDP[18], DNS[19], SDP[20], UDDI[21]) to harvest over the interconnected Embedded System Devices (ESDs) all the available SPD services, functionalities, resources and information that can be composed to improve the SPD level of the whole system. In order to properly work, a discovery process must tackle also a secure and dependable service registration, service description and service filtering. The service registration consist in advertising in a secure and trusted manner the available SPD services. The advertisement of each service is represented by its formal description and it is known in literature as service description. The registered services are discovered whenever their description matches with the query associated to the discovery process, the matching process is also known in literature as service filtering. On the light of the above a SPD services discovery framework is needed as a core SPD functionality of a pSHIELD Middleware Adapter. Once the available SPD services have been discovered, they must be prepared to be executed, assuring that the dependencies and all the services preconditions are validated. In order to manage this phase, a service composition process is needed.

Service composition is in charge to select those atomic SPD services that, once composed, provide a complex and integrated SPD functionality that is essential to guarantee the required SPD level. The service composition is a pSHIELD Middleware Adapter functionality that cooperates with the pSHIELD Overlay in order to apply the configuration strategy decided by the Control Algorithms residing in the pSHIELD Security Agent. While the Overlay works on a technology independent fashion composing the best configuration of aggregated SPD functionalities, the service composition takes into account more technology dependent SPD functionalities at Node, Network and Middleware layers. If the Overlay decides that a specific SPD configuration of the SPD services must executed, on the basis of the services' description, capabilities and requirements, the service composition process ensures that all the dependencies, configuration and pre-conditions associated to that service are validated in order to make all the atomic SPD services to work properly once composed.
When the SPD services have been discovered and a feasible service composition has been identified, those services must be deployed, executed and continuously monitored. This is part of the **service orchestration** pSHIELD Middleware Adapter functionality. While service composition works "off-line" triggered by an event or by the pSHIELD Overlay, service orchestration works "on-line" and is continuously operating in background to monitor the SPD status of the running services.
Secure service discovery, service composition and service orchestration operate at pSHIELD Middleware Layer and have access to the information coming from the Middlware, Network and Node layers as well as from the Overlay. These core SPD functionalities can take advantage from the information provided by the running services to "sense" the context or the situation in which the system is operating. Such a capability allow to introduce an additional pSHIELD Middleware Adapter functionality that is *context*

[16] IETF Service Location Protocol V2 - http://www.ietf.org/rfc/rfc2608.txt
[17] UPnP Simple Service Discovery Protocol - http://upnp.org/sdcps-and-certification/standards/
[18] IETF Neighbour Discovery Protocol - http://tools.ietf.org/html/rfc4861
[19] IETF Domain Name Specification - http://www.ietf.org/rfc/rfc1035.txt
[20] Bluetooth Service Discovery Protocol
[21] OASIS Universal Description Discovery and Integration - http://www.uddi.org/pubs/uddi_v3.htm

awareness. The context awareness is a pervasive functionality, that is embedded in the discovery, composition and orchestration processes, thus it does not represent an atomic core SPD functionality but an additional characteristic of the pSHIELD Middlware Adapter functionalities. In pSHIELD we introduce the context awareness of the pSHIELD Middleware Adapter core SPD services with a semantic approach, extending the service description model with context aware requirements[22].

5.5.2 Formalized conceptual model

The formalized conceptual model of the core SPD services has been conceived refining the pSHIELD functional component architecture (Figure 3.16 in Section 4.1) and have been derived from the study of the requirements of the pSHIELD application scenario (see deliverable D2.1.1 for more detail).

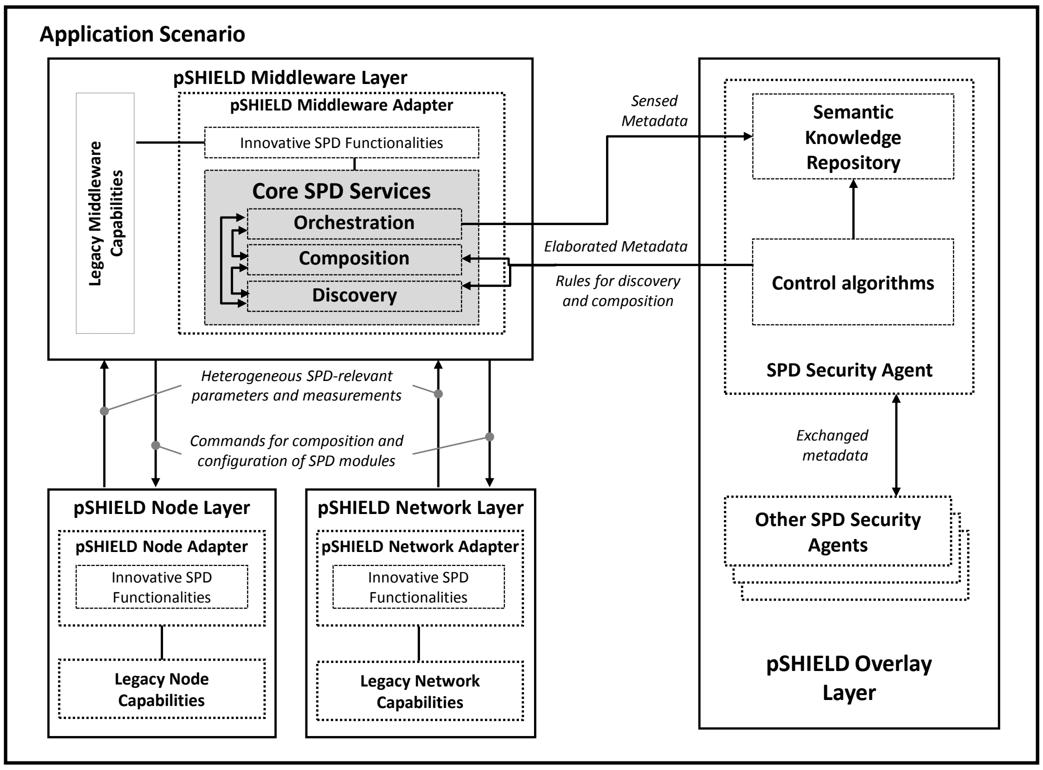

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Innovative SPD Functionalities
Core SPD Services
Orchestration
Composition
Discovery
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
Other SPD Security Agents
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
pSHIELD Overlay Layer

Figure 5.18 Core SPD services in the pSHIELD functional component architecture

The Orchestration, Composition and Discovery functionalities are the enablers (i.e. the sensors and the actuators) of the decisions taken by the pSHIELD Security Agent Control Algorithms residing in the pSHIELD Overlay. The mutual interoperation between the pSHIELD Middleware Adapter and the pSHIELD security Agent enables the pSHIELD Composability concept (as described in section 4.5).
It is worth to note that not all the core SPD services must be necessarily located in each pSHIELD Embedded System Device (pS-ESD). Indeed the pSHIELD component architecture depicted in Figure 5.18 identifies the Discovery, Composition and Orchestration functionalities that must be supported by at least one pS-ESD in a network of Embedded System Devices. Moreover the core SPD services can be deployed applying centralized or distributed approaches. It is a matter of the precise application scenario to decide whether a specific functionality must be supported by each Embedded System Device (ESD). It is obvious that the more ESDs are equipped with the pSHIELD Middleware Adapter (resulting to be a pS-ESD), the more will be the coverage area and the effectiveness of the pSHIELD functionalities to guarantee a certifiable SPD level (based on common shared SPD metrics) over the whole system.

Let see more in detail the formalized conceptual model of the Core SPD services, detailing the architecture depicted in Figure 5.18 and exploding the core SPD services into their functional components, in compliance with the pSHIELD functional architecture described in section 4.1.2 .

[22] V. Suraci, S. Mignanti, "Context-aware Semantic Service Discovery", 16th IST Mobile & Wireless Communications Summit Budapest, Hungary 1-7 July 2007. Proceedings. #273.

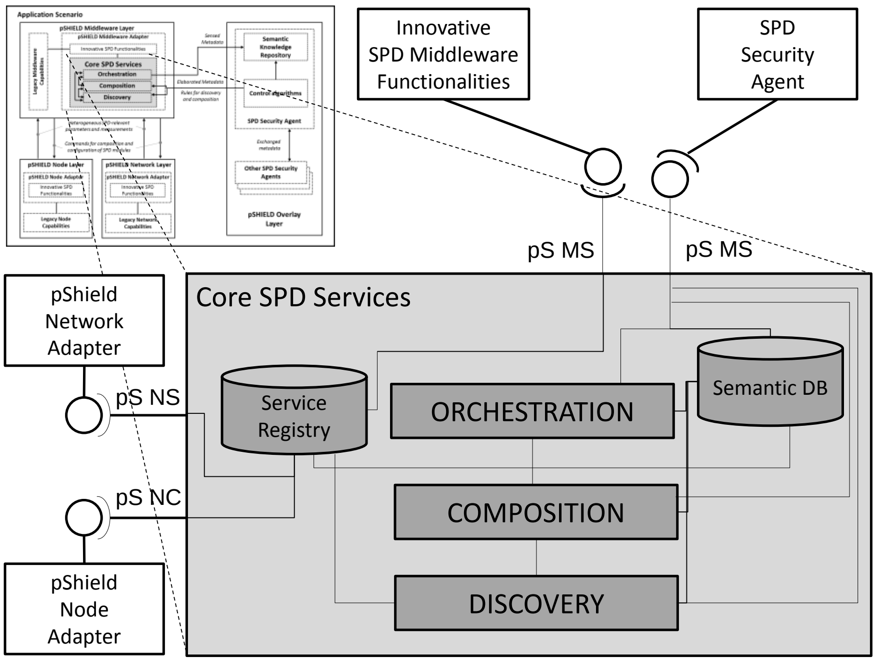


Figure 5.19 Core SPD services conceptual framework

Apart the Discovery, Composition and Orchestration components already described in the previous section, the following additional conceptual entities have been introduced:

- Service Registry: it acts as a database to store the service entries (e.g. the SPD components description described in section 4.4.2.1, that contains a description of functionality, interface, semantic references, etc.). Any pSHIELD Node, Network or Middleware layer component can be registered here to be discovered.
- Semantic DB (Database): it holds any semantic information related to the pSHIELD components (interface, contract, SPD status, context, etc.). The use of common SPD metrics (defined in section 0) and of a shared ontology (derived from the formalized semantic model detailed in section 5.4) to describe the different SPD aspects involved in guaranteeing a precise level of SPD, allows to dominate the intrinsic heterogeneity of the SPD components. Any semantic data is thus technology neutral and it is used to interface with the technology independent mechanisms applied by the pSHIELD Overlay.

Focusing exclusively on the Core SPD services located in the pSHIELD Middleware Adapter, we can describe how it works when it is in an operative status. Let consider a typical situation, where the whole system is properly working at runtime. The Orchestration functionality is in charge to monitor continuously the Semantic DB with the updated status of the functionalities operating at node, network and middleware layers. As explained in Section 3.1.1, the pSHIELD Adapters are in charge to update in the Semantic DB (by proper Semantic hubs that, for the sake of simplicity, have not been shown in Figure 5.19) their status. Whenever the needed application SPD level, for any reason, due to external/internal unforeseen/ predictable events, changes and go beyond the threshold, the Orchestrator triggers the Overlay. The Overlay try to react and to restore the SPD level back to an acceptable level identifying the best configuration rules. The Discovery and Composition are then triggered by the pSHIELD Overlay with the aim to apply the configuration rules. On the basis of the configuration rules, the Composition service make use of the Discovery service to search for all the needed and available SPD components. The Composition service analyzes the SPD components interfaces and contracts to determine which SPD components are required, which should be activated and in which order to make the configuration of SPD components properly work. Thus while the Overlay operate in a technology independent fashion, the Composition service operates all the needed low-level, technology-dependent activities to actuate the Overlay decisions.

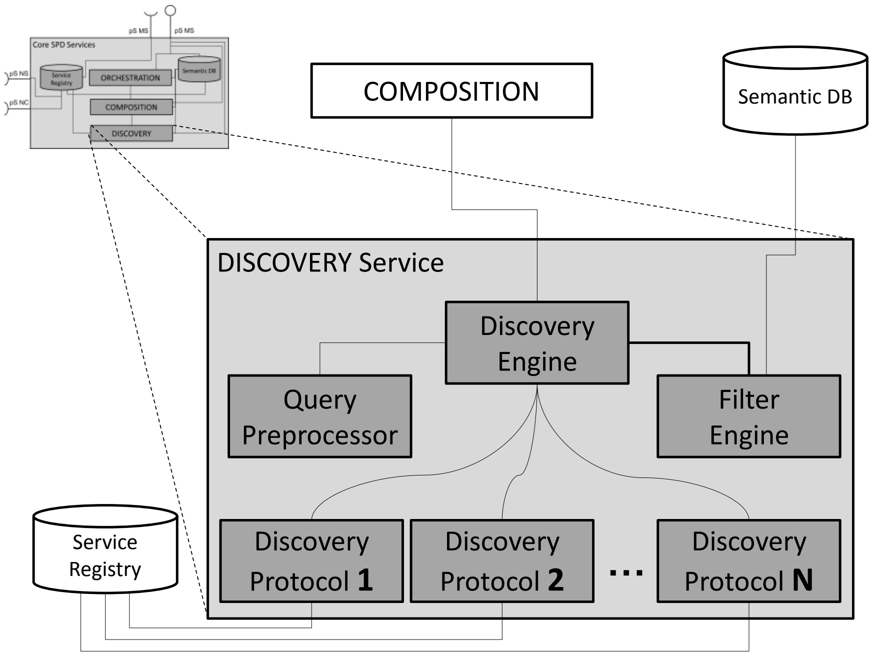


Figure 5.20 Details of the Discovery core SPD service

Zooming more in the detail the Discovery service, as shown in Figure 5.20, the following elements can be distinguished:

- **Discovery Engine**: it is in charge to handle the queries to search for available pSHIELD components sent by the Composition service. The Discovery Engine manages the whole discovery process and activates the different functionalities of the Discovery service: (i) the query pre-processor to enrich semantically and contextually the query, (ii) the different discovery protocols to harvest over the interconnected systems all the available pSHIELD components, (iii) the Filter Engine to discard those components not matching with the enriched query;
- **Query Pre-processor:** it is in charge to enrich the query sent by the Composition service with semantic information related to the peculiar context. The query pre-processor can be configured by the Overlay to take care of the current environmental situation;
- **Discovery Protocol:** it is in charge to securely discover all the available SPD components description stored in the Service Registry, using a specific protocol (e.g. Service Location Protocol – SLP or Universal Plug and Play Simple Service Discovery Protocol – UPnP SSDP, etc.). Indeed the SPD component descriptions can be registered in different types of Service Registries, located everywhere in the network, using heterogeneous protocols to be inquired;
- **Filter Engine:** it is in charge to semantically match the query with the descriptions of the discovered SPD components. In order to perform the semantic filtering, the Filter Engine can retrieve from the Semantic DB the information associated to the SPD components, whose location is reported in the description of the SPD component.

The composition engine tries to accomplish the pSHIELD Overlay configuration rules applying the following procedure:

1. Composition service triggers the Discovery service, sending a SPD component request, looking for those SPD components defined in the configuration rules provided by the Overlay;
2. The Discovery Engine sends the request to the Query Preprocessor;
3. The Query Preprocessor enriches the service request with contextual information and sends it back to the Discovery Engine;
4. The Discovery Engine applies a global service discovery using heterogeneous Discovery Protocols, in order to collect as much available SPD functionalities as possible over the networked Embedded System Devices;

5. Each Discovery Protocol interacts with the Service Registries reachable in the network and retrieves the SPD components' descriptions and provides them back to the Discovery Engine;
6. The Discovery Engine collects the discovered descriptions and sends them to the Filter Engine;
7. The Filter Engine applies a semantic filtering, retrieving the semantic metadata from the semantic DB, accordingly with the references reported in each SPD component description. The filtered list of component is then sent back to the Discovery Engine;
8. The Discovery Engine sends the list of available, filtered SPD components to the Composition service;
9. If the Composition service, considering the available SPD components is able to provide a new configuration, these components are activated, otherwise the Composition service advise the Overlay that it is not possible to apply its decision.

It is important to note that the validity of this conceptual framework model is independent from the specific application scenario. On the basis of this conceptual framework it is possible to derive a number of possible alternative implementations, belonging to different pSHIELD compliant technology providers. If the interfaces and the operation between the different elements are respected, it is possible to setup heterogeneous systems with the enhanced pSHIELD SPD functionalities.

However being pSHIELD a pilot project, a targeted demonstrator will be setup starting from the conceptual framework, deriving an instance of the presented framework oriented to achieve the target SPD objectives defined in the specific application scenario.

5.6 Hybrid Automata

5.6.1 Description

In section 4.4 the composability mechanism has been described, starting from the atomic pSHIELD SPD components, and arriving to the final topology of the interconnected functionalities. However, up to now, it has not been addressed the most important point at the basis of the Overlay control algorithms: how to choose the most suitable configuration rules for those components that must be composed.

The control algorithms receive as input a reference value (expressed using the SPD metrics described in section 4.6) representing the desired SPD level associated to the specific application scenario. They have to provide, as output, the correct configuration that assures the desired SPD level. In order to simplify the definition of the algorithm, we assume that all the configuration examined by it are feasible, meaning that the non-feasible configurations are automatically recognized and ignored (for example by the discovery service while populating the semantic repository of the components).

A trivial solution is a "brute force" approach that uses the metrics evaluation procedure defined in M0.2 to quantify all the potential solutions and choose the one closer to the desired level of SPD (if more than one configuration is suitable, then the algorithm will chose the first one computed).

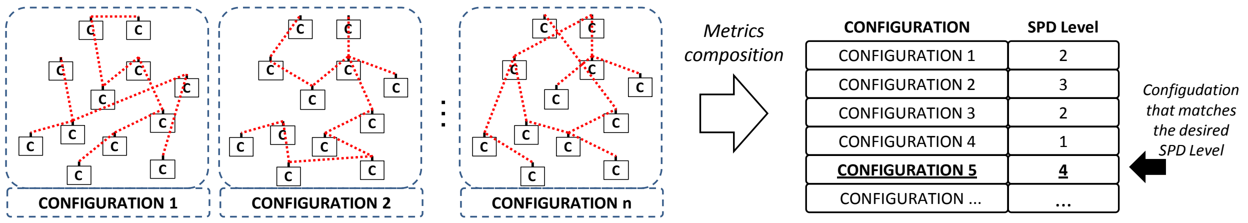


Figure 5.21 Example of a brute force approach to choose the most suitable configuration

This approach will for sure lead to a suitable solution, but this solution is not necessarily "optimal" and, moreover, this kind of algorithm is not so reactive to system's change and doesn't provide the dynamic behaviour that the Overlay is supposed to offer, i.e. the capability of adapting to the changing context.

In fact if we imagine that, at run time, one of the system components is subject to a fault, we are not efficiently able to assert automatically whether the configuration is still valid or not (in terms of SPD assurance) and if we want to perform a corrective action to increase, for example, the SPD level again, we have to compute all the configurations from the beginning and to find a new solution. If we need short time-response, we are not sure that this solution is optimal again.

Since we want to perform dynamic, reactive and proactive control of the composing elements, we need to build a model with additional information: a model of the system's configuration plus its evolving dynamics in different operating conditions.

Since the context of ES is heterogeneous, the choice of the adequate mathematical framework to build this model has to be done carefully.

5.6.2 Formalized conceptual model

The starting point is to model a set of heterogeneous "entities" interacting together and providing different functionalities.

A standard I/O model is not suitable, since there is no clear information (energy) flow among the components.

A basic assumptions is made: the model should capture "global" properties, without focusing on individual properties (we want to provide holistic/vertical SPD). By doing so, the problem is moved to an upper abstraction layer and each atomic component provides a little contribution to the overall description.

The resulting composition is a mix of both continuous and discrete dynamic, so an hybrid formulation is needed.

Last, but not least, the resulting composition should model and replicate the evolution of the system (in nominal condition or in presence of faults), so an automaton formulation is needed.

Merging together all these needs the most suitable approach seems to be based on Hybrid Automata .

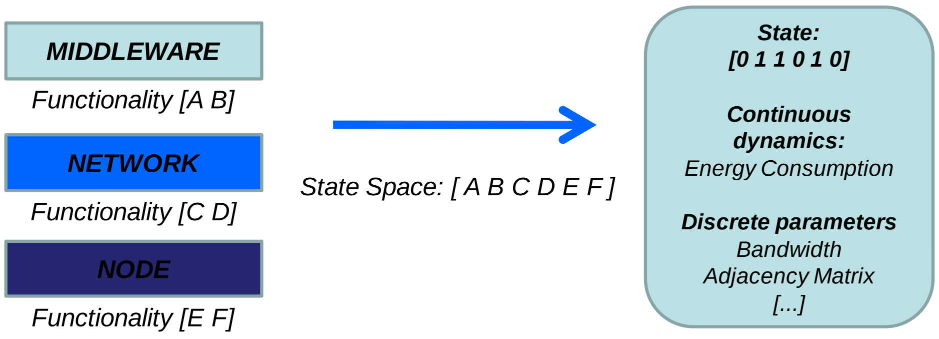


Figure 5.22 Example of a Hybrid Automaton representing a specific state

Now that the formal modelling language has been identified, the procedure is very simple:

- At first we should identify the system "state", i.e. the set of active functionalities (see, as an example, Figure 5.22). This will be the system identifier in a specific circumstance and represents simply the configuration of active components (a boolean vector with "1" for active component, and "0" for switched-off component).
- Then we must retrieve the *continuous dynamics* that represents the system evolution in this specific configuration as well as the *discrete parameters* related to the global properties for the current state. They can be, for example, the dynamics describing the evolution of the battery of the nodes or the percentage of CPU used, or the level of congestion of the network, etc., while the discrete parameters can be the associated SPD level, the overhead of the current encryption algorithms, the resilience level of a routing protocol, the bandwidth of the network, data rate of generated output, etc.
- At last we must identify the transition, i.e. connect each state with the states that are reachable from him (i.e. by changing the status of only one component). We assume that only one component at a time can change its status (this is a realistic assumption if the time discretization sample is small enough). Transitions may be voluntary and expected (control action) or not (fault): in both the cases the model still works.

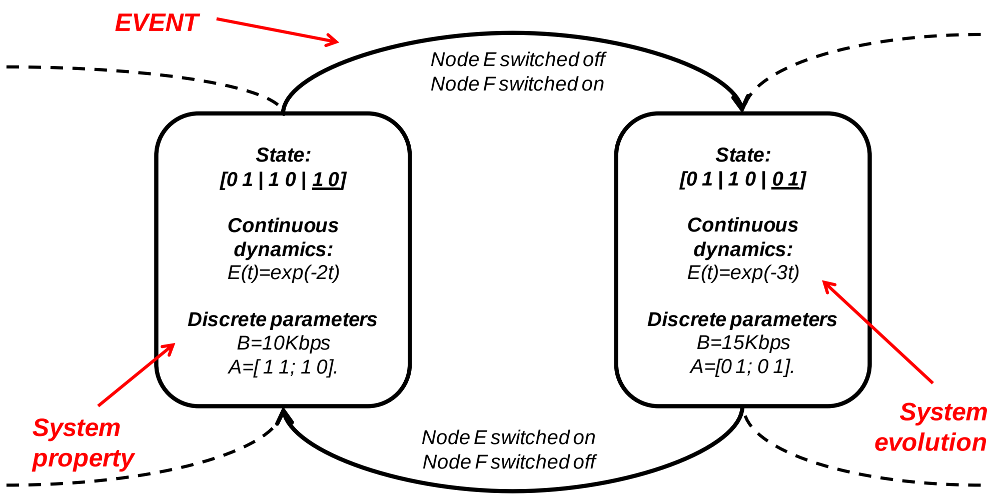


Figure 5.23 Hybrid Automata representing the system evolution

Now the model for the Overlay control algorithms is ready: we have created a map of all the possible configurations enriched with information about the systems evolution that will bring (or come from) those configurations; at any time we can give a look to the system and be aware of the state in which we are (current configuration), the associated SPD level and the potential corrective actions (adjacent transitions) that we can perform if a component fault occurs.

Moreover, when more than one option is available, the algorithm can choose the best action according to pre-defined objective function associated to the discrete parameters or to the evolving dynamic, thus driving the choice in a more intelligent way and realizing a context-aware control (for example if we need to activate one additional node as a consequence of a fault and we can choose between two nodes that satisfy the desired SPD level, than we could choose the one with highest battery level or highest throughput).

Once the configuration or corrective action is computed, then it is translated by the pSHIELD semantic framework into composition rules and back again into commands sent to the pSHIELD Adapters to implement the control action (that is technology dependent).

Of course this approach overcome the limits of a simple brute force approach in terms of optimal solution and smartness of the control, but doesn’t fully solve the problem of scalability, that in composed system is the biggest trouble. However, in the scope of the pSHIELD project, the demonstrator (see section 5.8) is willing to demonstrate, as first step, the validity of this approach, leaving to further investigations the solving of scalability related issues (nSHIELD).

Nevertheless some modifications to the Hybrid Automata approach are already under investigation, basing on the assumption that, since a linear increase in the number of components results in an exponential increase in the number of configurations, it could be better to model with an Hybrid Automata the component itself and not the configuration.

5.7 Common Criteria approach

5.7.1 Description

pSHIELD purpose can be summarized in a sentence as “a pilot for SPD oriented assurance definition for embedded system” *and its graphic representation is the following scheme:*

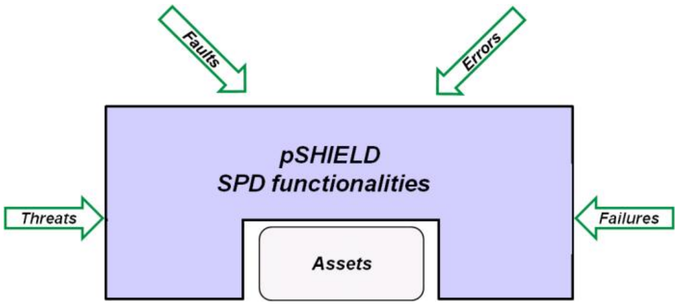


Figure 5.24 pSHIELD as a pilot for *SPD oriented assurance for embedded system*

Figure 5.24 and the process that it represents can be explained through the following sentences:

- Assets to protect and in particular SPD attribute of these assets definition
- Threats identifications (Fault → Errors → Failures); in our approach faults are grouped in HMF (FUA, NFUA) and NHMF.
- SPD functionalities (whose purpose is to mitigate threats) are implemented to meet pSHIELD SPD objectives

This last sentence is the starting concept to metrics definition and in particular to Common Criteria approach adopted.

5.7.2 Formalized conceptual model

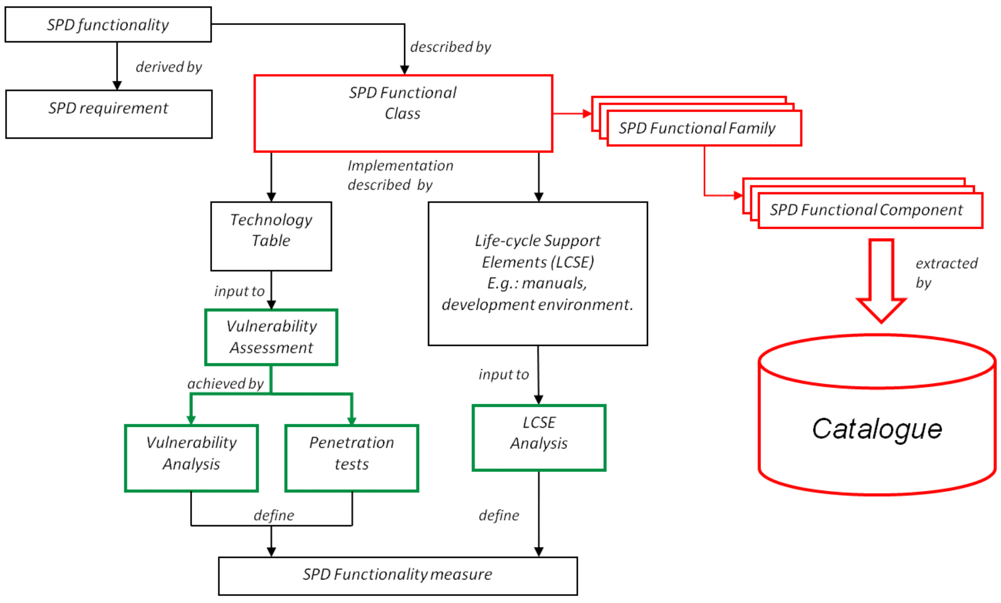


Figure 5.25 conceptual model for the calculation of SPD status associated to a generic pSHIELD SPD functionality (i.e. SPD component)

As we can see in Figure 5.25, each SPD functionality description can be extracted by a catalogue (based on Common Criteria part 2[23] for FUA mitigation and Common Criteria part 3[24] for NFUA and NHMF mitigation) where SPD functionalities are grouped in Classes and Families (Figure 5.25 – red evidenced part),

The two parallel process (left FUA, right NFUA and NHMF) identified in Figure 5.25 (green evidence part), which lead to SPD functionality measure, are both based on Common Criteria.

In particular the first one is based on a vulnerability assessment and the second one on evaluation of defined Life-Cycle supports element (e.g.: guidance, manuals, development environment, etc.).

After calculating measures for all SPD functionalities, it is possible to define a single measure for the whole pSHIELD system considering the composability approach identified in deliverable M0.2[25].

[23] Common Criteria for Information Technology Security Evaluation- Part 2: Security functional components – July 2009 – Version 3.1 – Revision 3 – Final – CCMB-2009-07-002
[24] Common Criteria for Information Technology Security Evaluation- Part 3: Security assurance components – July 2009 – Version 3.1 – Revision 3 – Final – CCMB-2009-07-003
[25] Project no: 100204 – pSHIELD – D0.2: Proposal for the aggregation of SPD metrics during composition

5.8 pSHIELD Demonstrator

5.8.1 Description

The reference application scenario for pSHIELD project is the monitoring of freight trains transporting hazardous material. As witnessed by the results of risk assessment and accidents happened in recent past, this is an important problem to be addressed in the context of critical infrastructure protection and railway security. Therefore, the detection of abnormal operating or environmental conditions on board of vehicles as well as threats of burglary represents an example application of great interest for the freight train monitoring. In this scenario, both natural and malicious faults can have an impact on system availability and indirectly on safety.

The scope of this application scenario is to validate the mentioned key concepts (described in section 4) of pSHIELD project, in order to provide a preliminary, but exhaustive investigation and proof of concept of the innovative SPD functionalities (described in section 5) as well as of the overall pSHIELD framework (described in section 3), which will be definitely completed in the nSHIELD project. In particular, in this scenario, the assurance of a certain SPD level in the monitoring of the hazardous material and in the handling of critical information regarding hazardous material represents an important requirement to be fulfilled.

The overall monitoring system is highly heterogeneous in terms not only of detection technologies but also of embedded computing power and communication facilities. Since the ESs can differ in their inner hardware-software architecture, they can differ also in the capacity of providing information security, privacy and dependability. The main challenges addressed in the pilot demonstration are the following:

- the autonomous adaptation of the ESs to the changing internal or external conditions, by means of a proper overall process, rather than from the viewpoint of the single ES;
- the composability of SPD enabling technologies enhanced with pSHIELD SPD functionalities, by means of the integration of the Overlay with the other three functional layers (realizing thus the pSHIELD framework), in order to assure a SPD level assessment based on the use of the pSHIELD SPD metrics.

The main objectives of the pilot demonstration are:

- to monitor on-carriage environment;
- to integrate the sensors at the wagon with the M2M platform;
- to allow secure interoperability of sensor information (between railway infrastructure and third party service provider).

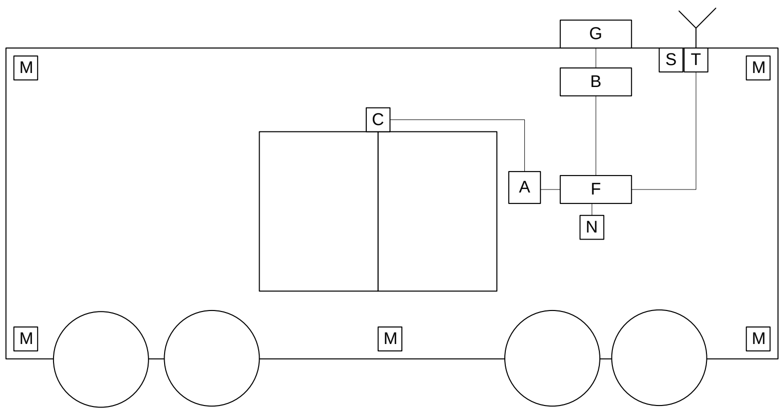


Figure 5.26 Architectural scheme for the remote monitoring

In Figure 5.26, we can see an architectural scheme for the remote monitoring of unpowered freight cars possibly used to transport valuable and/or hazardous materials, based on the following devices:

- A: Access control device powered on-demand (e.g. numeric keypad featuring “ON” button);
- B: Long lasting battery pack;
- C: Magnetic contact or other very-low consumption intrusion detection device;
- F: Central control unit featuring embedded CPU and OS;
- G: Low size power generator (e.g. eolic, solar panel, piezoelectric pad) – optional;
- M: Self-powered smart wireless sensor measuring vibrations, temperature, humidity, light, sound;
- N: Gateway node for the wireless sensor network;
- S: Satellite positioning antenna;
- T: Wide area wireless transceiver (GSM, GPRS, UMTS, EDGE).

The monitoring is aimed to the detection of abnormal operating or environmental conditions on board of vehicles as well as threats of burglary. So, the basic working logic of the system is as follows:

- Case of abnormal environmental or operational conditions: whenever an abnormal event (e.g. very high temperature or out of range vibrations) is detected by M, its transmission unit is activated and data is received by N. F validates data and - if the anomaly is confirmed - it activates S to achieve the current position and T to send an appropriate warning message to the control center.
- Case of intrusions: whenever C detects an opening while the alarm system is activated (by A), F activates S to achieve the current position and T to send an appropriate alarm message to the control center.

More in detail, in the described context, several problems are to be solved. They are referred to:

- *Availability*: information (measured data, real-time streams from audio/video-surveillance devices, smart-sensor alarms, etc.) must be provided continuously according to soft or hard real-time constraints.
- *Integrity*: sensed, stored or transmitted data must not be corrupted accidentally or in a malicious way.
- *Privacy*: information must be accessed only by authorized users, for confidentiality reasons.
- *Integration/expansion dependability*: whenever any new ES needs to be integrated into the system, there should not be the need to develop a new protocol and/or driver and it should be also possible to easily evaluate the impact of the modification on the overall system dependability.

Finally, the holistic assurance and evaluation of SPD parameters (e.g. for assessment/certification purposes) represent another important task to carry out to validate the key concepts of the pSHIELD project.

5.8.2 Modelling of the Intelligent Wagon

The pSHIELD scenario is addressing rail transportation of dangerous goods by a wagon. This section introduces the concept of the Intelligent Wagon (IW) as in Figure?.
The security of such IW that is transporting dangerous goods must guarantee the safety level of the overall system[26]. In the pSHIELD project will focus on the Intelligent Wagon Infrastructure (IWI). Since the rail transportation system ore not well covered by specific standards that are taking in consideration

[26] H. Hadj-Mabrouk et all., “Example of topologie d’accidents dan le domaine des transport guides,” Revue generale des chemins de fer, march 1998, pp. 17-55.

particularities of the pSHIELD scenario, the design of such system becomes difficult in absence of standardised design requirements of this system. This is due to the recent innovative changes posed by the European new legislative standards for rail transport, which are demanding a high level security for the new transportation systems.

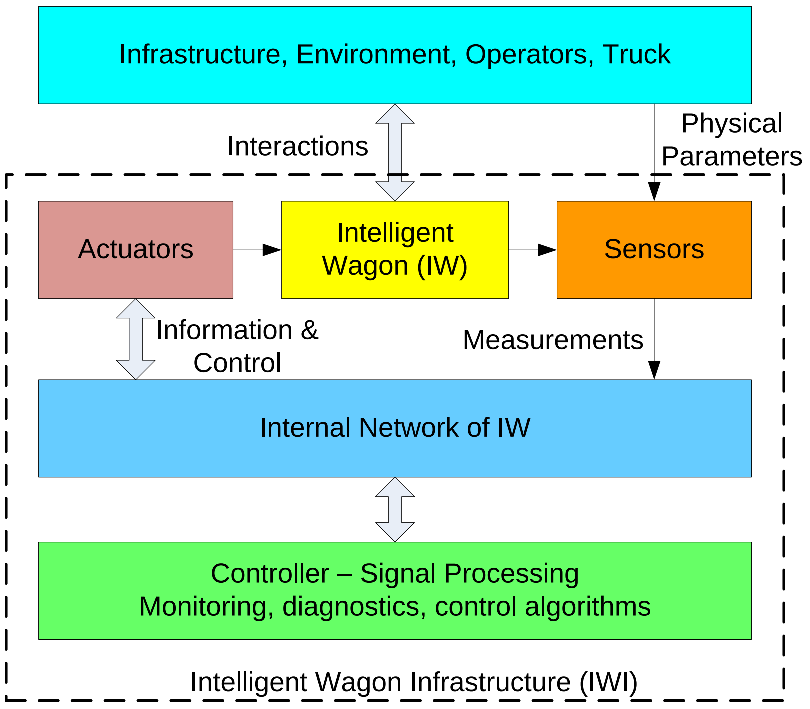

Infrastructure, Environment, Operators, Truck
Interactions
Physical Parameters
Actuators
Intelligent Wagon (IW)
Sensors
Information & Control
Measurements
Internal Network of IW
Controller – Signal Processing
Monitoring, diagnostics, control algorithms
Intelligent Wagon Infrastructure (IWI)

Figure 5.27 Model of the Intelligent Wagon (IW)

Why we need an IW? In current system the wagon play a passive role in terms of security and dependability. This transportation wagon has no power on it, or sensors and actuators (S&A), to deal with possible risks. In D2.1.1 document several requirements are proposed in terms of Security, Privacy and Dependability (SPD). For example two networked SPD nodes has non-functional properties, such as security and privacy (e.g., confidentiality of data), dependability (e.g., a continuous communication without interruption). Modelling system security and dependability is not common practice in software project yet. There is not a widely accepted methodology which unifies the actual heterogeneity of SPD issues when addressing a whole set of SPD specifications. The situation is even worst since there is not a standard common notation for carrying out the SPD specifications. The use of pSHIELD framework can overcome all these open issues. On the other hand, tight relationship between dependability and security was clearly given by Avizienis, Laprie and Randell[27]
The aim of SDP requirements for a generic scenario is to ensure security, availability, and dependability as well as privacy and trust. Thus, the resulting requirements are:

- multi- purpose (to cover more than one scenario)
- multi-functional (to cover all SPD functionalities envisioned for different scenarios)
- multi-technology (to include all fundamental technologies, existing and new one for different scenarios).

For the selected scenario in pSHIELD “transportation of dangerous goods” these generic requirements will be reduced to a sub-set of requirements and specifications that will define the work in WP3 (Node Layer), WP4 (Network Layer), WP5 (Middleware and Overlay Layer). As result we will have an automated system adapted to the rail transportation of dangerous goods. This automated system will be defined by the concept of the Intelligent Wagon (IW).

[27] A. Avenzienis, J. C. Laprie, B. Randell, “Basic Concepts and Taxinomy of Dependable and Secure Computing, IEEE Trans. On Dependable and Secure Computing, pp. 11-33, 2004.

Figure 5.27 shows a model of the IW. This IW has power node, nano/micro personal nodes, controllers, computers, etc. The main task of this Intelligent Wagon Infrastructure (IWI) is to perform monitoring, diagnostics and control functionalities. SPD services and functionalities play the key role in the IWI concept. The wagon has a network that connects the processing part of the system with S&A. This network enables to transmit signals to actuators and retrieve information from the actuators and in the same time pass the measured information from the sensors that are achieved from the environment inside or outside of the wagon. Designing such automated system requires defining well all its operational phases. This identification will be done by a functional analysis. By defining all these functions it is possible to determine an optimised pSHIELD architecture derived from the pSHIELD framework presented in the previous sections.

5.8.2.1 Two steps design procedure

Designing an IW (i.e. find the best matching instance of the pSHIELD framework) will be performed by two steps: modelling and optimisation. The modelling step determines the functional model characteristics. It will be done in a hierarchical structure. The optimisation is based on a set of solutions that will be analysed in such a way that an optimal solution will be derived from them[28].

5.8.2.1.1 Functional modelling

For functional modelling is important to represent all realisable component architectures. This model is based on two dependable architectural procedures. The first one is to describe the system based on its functional decomposition. The second one is related to the security and dependability models and their relationships.

The functional model is based on its methodology, e.g., functional analysis. To realise a demonstrable architecture as in Figure 5.26 it is important to include different types of pSHIELD nodes:

- Nano, micro/personal nodes
- Power nodes

In the pSHIELD scenario we have pSHIELD Embeeded System Devices (hereafter referred to as SPD nodes) and Legacy Embedded System Devices (hereafter referred to as ED nodes). SPD nodes have SPD functionalities. Alternative nodes are Legacy ED nodes. There can be also simple nodes as ordinary sensors (e.g., Component on the Shelf – COTS). Sensor 1, 2, 3 and 4 can be realised a combination of SPD node and/or alternative node, and/or even an elementary node. SPD node expresses a complex function that is composed of a set of SPD functions. Alternative node offers various possibilities for performing a function. The elementary node corresponds to a basic function associated with single sub-function or component. The alternative nodes can be adapted to SPD.

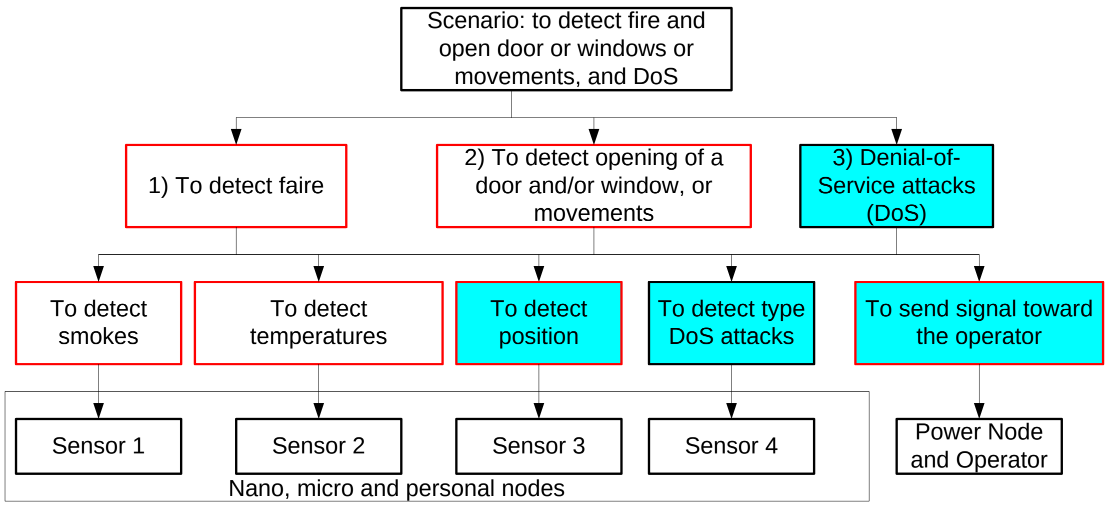

Scenario: to detect fire and open door or windows or movements, and DoS
1) To detect faire
2) To detect opening of a door and/or window, or movements
3) Denial-of-Service attacks (DoS)
To detect smokes
To detect temperatures
To detect position
To detect type DoS attacks
To send signal toward the operator
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Power Node and Operator
Nano, micro and personal nodes

Figure 5.28 Functional three of three protection systems for the IW

[28] J. Clarhaut et all., "Optimal design of dependable control system architectures using temporal sequences of failures," IEEE Trans. On Reliability, vol. 58, No. 3, pp. 511-522, 2009.

Figure 5.28 illustrates the components that can be used for realisation of the protection system that addresses 1) fire, 2) opening door or windows, movements and 3) DoS attacks. For each of this protection system several redundancies can be realised thanks to alternative or simple nodes. These three protection functions heave several sub-functions. The functions: to detect smokes, temperatures, positions, and DoS are shared functions for three main protection sub-systems. Redundancy can be realised as serial or parallel connection of two sensors.
The functional model is the skeleton of multi-fault three. For that we need to associate to each node a set of failure modes. For a complex function the relationships between their failure and those of their sub-functions can be realised by using operator like AND, OR, etc.

5.8.2.1.2 Optimisation

The optimisation of a system has aim to determine the best system's architecture among all potential architectures. An approach is proposed by D'Ariano[29]. Each component is supposed to have a cost and a level of dependability defined. As an initial step is the definition of a set of the architectural solutions. This set is made by several feasible solutions. Each solution is composed of one possibility of component for each sub-function, starting with the lowest sub-function of the three. Then, each feasible solution is compared to the others. Only the first optimal solutions are selected. This set is built gradually as the top of the three is reached. This optimisation procedure can be used only under hypothesis that the functions are independent. This assumption means that every function, sub-function and component is only used one time in the functional model. From this short introduction in the optimisation process is obvious that such optimisation cannot immediately improve such systems. Based on the number of shared functions two approaches can be used:

1. If the shared functions are few and / or separate (independent on other shared functions), transposition of the functional model into a functional model without shared functions are performed.

2. If there are many shared functions and/or interdependent, the use of another optimisation algorithm can be a solution. This work is in progress.

In the above short description a general concept of the rail transportation of dangerous goods with an IW is presented. The example of three protection systems has been introduced. These protections systems are demonstrating that the concept of the IW can be modelled. Future works will focus on the optimisation algorithms with few independent and more interdependent shared functions.

5.8.3 Integration and Interoperation

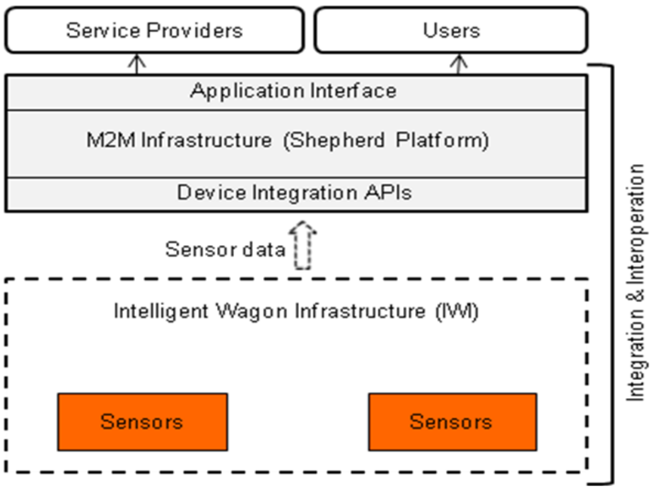

Service Providers
Users
Application Interface
M2M Infrastructure (Shepherd Platform)
Device Integration APIs
Sensor data
Intelligent Wagon Infrastructure (IWI)
Sensors
Sensors
Integration & Interoperation

Figure 5.29 Integration & Interoperation concept of heterogeneous services, systems and devices in pSHIELD demonstrator.

[29] A. D'Ariano et all. "A branch and bound algorithm for scheduling trains in a railway network," European Journal of Operational Research, vol. 183-2, pp. 643-657, Science Direct Ed., 2006.

One of the objectives of pSHIELD is to achieve Integration and Interoperation of heterogeneous services, systems and devices. The pSHIELD demonstrator will address the interoperability of information from Railway Infrastructure domain to third party service providers through an M2M platform provided by Telenor Objects, Norway. The demonstrator will make the information coming from the sensors at the Intelligent Wagon Infrastructure (IWI) accessible from anywhere at any time. However it is assumed that only authorized services providers can access them. pSHIELD demonstrator will address the integration of sensor data with the M2M platform. Figure 5.29 illustrates the concept of Integration and Interoperation of heterogeneous services, systems and devices in pSHIELD demonstrator.

Shepherd platform provided by Telenor Objects acts as M2M platform. Shepherd allows any pluggable objects (here micro and power nodes) to be connected to the platform through devices APIs and makes the sensor information securely and reliably available to the service providers or users through application interface.

5.8.4 Instances of demonstrator equipment

This section briefly describes the examples of few of the pSHIELD demonstrator equipments such as nodes and M2M platform.

5.8.4.1 pSHIELD power node

5.8.4.1.1 Introduction

The VIA Embedded Board (VIA EPIA N700) is considered as a potential power node for the pilot demonstrator that will be used by the Norwegian members of the consortium. The VIA EPIA N700 offers total system stability at extreme temperatures ranging from -20°C to 70°C, an ideal solution for our Norwegian rail use case to meet the extreme weather condition of Norway.

5.8.4.1.2 Technology description

The VIA EPIA N700 is a compact, low heat, power-efficient Nano-ITX board, ideal for compact industrial PCs and embedded automation devices. The board is integrated with the VIA VX800 media system processor, an all-in-one chipset solution that provides an extensive feature set while using less real state, helps to make the VIA EPIA N700 a superb choice for compact systems. It is based on Nano-ITX form factor (12cm x 12cm. VGA, USB, COM, Compact Flash (CF) and Gigabit network ports are provided on the board to help reduce system foot-print size and eradicate cluttered cabling for improved air-flow and enhanced stability in always-on systems.
The VIA VX800 offers an integrated DirectX9 graphics core and excellent hardware accelerated video playback for MPEG-2, WMV9, VC1 video formats. An on-board VGA port is provided along with support for DVI and a multi-configuration 24-bit, dual channel LVDS transmitter, enabling display connection to embedded panels.
The VIA EPIA N700 is equipped with a power-efficient 1.5GHz VIA C7, supports up to 2GB of DDR2 system memory and includes two onboard S-ATA connectors, USB 2.0, COM and Gigabit LAN ports. Expansion includes a Mini-PCI slot with an IDE port, additional COM and USB ports and PS/2 support available through pin-headers.

5.8.4.2 pSHIELD micro node

5.8.4.2.1 Introduction

The pilot demonstrator of pSHIELD uses the Sun SPOT sensor platform as micro node. Sun SPOT is a useful platform for developing and prototyping application for sensor network and embedded system. Sun SPOT is suitable for application areas such as robotics, surveillance and tracking.

5.8.4.2.2 Technology description

Hardware components

The main units are Sunspot devices with embedded sensors and base station. Each Sunspot has a so-called eSPOT with battery, while the base station is not equipped with battery and must be powered from the host computer via an USB cable. The Sunspot does not need to run any operating system, it needs only JVM that runs on bare metal, and executes directly out of flash memory. Stack-boards composed of specific sensors and actuators such as accelerometers, light sensor and temperature sensor. The hardware components of a sensor board is as follows:

- 180MHz for 32-bit ARM920T core processor with 512K RAM and 4M Flash, runs on Squawk
- 2,4GHz based IEEE 802.15.4 radio (radio ChipCon TI CC2420) which is integrated in the antenna
- USB interface for connecting to a host computer
- 3.7V battery (720 mAh), Sleep mode (32 uA)
- 3 axis accelerometer (2G/6G)
- 8 tri-color LEDs, 2 push-buttons control switches
- digital I/O pins, 6 analog inputs, 4 digital outputs

Integrated sensors

- Temperature Sensor: Chip-type is ADT7411 sensor that measures temperature with ADC. ADC is integrated into eDemo, and can measure temperatures between -40℃ to +125℃.
- Accelerometer sensor: 3-axis accelerometer of the type LIS3L02AQ, designed by ST Micro Systems and is in eDemo Board. This sensor can measure the x-axis, y-axis and z-axis in the direction up and down with the value either ±2G or ±6G. When the Sunspot is at rest, it measures x = y = 0 and z = 1G.
- Light sensor is of the type TPS851, designed by Toshiba. The sensor can measure the voltage between 0.1V (dark) - 4.3V (light), and converts the voltage to the brightness of Luminance (lx) 3.

Software components: SUN SPOT JVM

Squawk is open source and has been written in the Java programming language. It is a virtual machine, and is a highly portable Java VM. The advantage of Squawk is that it can run on bare metal instead of being run on top of the operating system. This means that applications can be isolated and be treated as application objects. This allows multiple applications running on the same virtual machine. Squawk also supports CLDC 1.1 that facilitates connectivity to mobile phones.

5.8.4.3 M2M platform

5.8.4.3.1 Introduction

In order provide services to relevant industrial actors, the nodes in the node layer need to be available and accessible by concerned entities anywhere and anytime. The connectivity with nodes and interoperability of such functionality will be addressed by introducing an instance of a middleware platform that the Norwegian consortium members will use for pSHIELD pilot demonstration. The planned semantic model is expected to facilitate the interoperability issues of such M2M platform. This section will briefly introduce its technological overview.

5.8.4.3.2 Technology description

Telenor Object, Norway have introduced a platform (named as Shepherd®) for interoperability and integration that supports communication between connected devices (or nodes) and makes them accessible from anywhere at any time. The Shepherd® is a platform for connected objects. This means that any the pluggable component can be connected, and be integrated in Shepherd® platform as a connected object.

The platform offers number of service including:

- Service Management for monitoring, device configuration, SLAs, and supporting.
- Service Enabler has a specific API that allows further access to other modules.
- Message Engine handles and secures the process of message flow, including capturing, processing, routing and storage of data in an environment.
- Notification services that inform about the status of devices and applications.

- Device library consists of interfaces for tools and services recognition.

Shepherd® offers two methods for establishing connectivity with nodes. These include:

- HTTP Connection API - This mechanism establishes a direct connection to the Shepherd by using the HTTPS protocol.
- Connected Objects Operating System (COOS) - is an open source and has been written in Java.

- This page intentionally left blank -

References

About the application scenario: Trail transportation of dangerous goods and rail transportation standards

[01] EN 501126 Railway Applications The Specification and Demonstration of Dependability, Reliability, Availability, Maintainability and Safety (RAMS) Issue: March 2000.

[02] H. Schabe, "The safety Phylosophy Behind the CENELEC Railway Standards, ESRL 202, Lyon, pp. 19-21, March 2002,

[03] Rail safety: transport of dangerous goods by rail, Summary of EU legislations, http://europa.eu/legislation_summaries/transport/rail_transport/l24061_en.htm

[04] Charlotte BOUISSOU et al., "A new QRA Model for rail transportation of Hazardous Goods," http://www.otif.org/otif/_defpdf/04_04_QRA-Model-F_E.pdf

[05] Joffrey Clarhaut et al., Design of dependable system architecture for railroad smart wagon using shared functions," Annual Conference on Intelligent Transportation Systems, Madeira Island, Portugal, pp. 1905-1910, September 2010.

About Embedded Systems with emphasis on security, privacy, dependability

[06] European *Dependability* Initiative: Extensively Deployed and. *Networked Embedded* Systems. Final report. 20th April 1998. ftp://ftp.cordis.europa.eu/pub/esprit/docs/stdepc.doc

[07] The ARTEMIS JU Annual Work Programme 2010, www.**artemis-ju**.eu/attachments/127/AWP_2010.pdf

[08] M. Knezevic et al., "Design Methods for Embedded Security," Telfor Journal, Vol. 1, No. 2, pp. 69-72, 2009.

[09] Tilman Wolf et al., "Collaborative Monitors for Embedded System Security," *Proc. of First International Workshop on Embedded Systems Security* in conjunction with *6th Annual ACM International Conference on Embedded Software (EMSOFT), Seoul, Korea, Oct. 2006.*

[10] Antonio Kung, "Enabling Trust for Safety and Security in Embedded Systems," M – ITEA 2 Magazine, *December 2010 – no. 8.*

[11] Jörg Henkel et al., "Security and Dependability of Embedded Systems: A Computer Architects' Perspective. VLSI Design 2009: pp. 30-32.

[12] Dimitrios N. Serpanos, and Jörg Henkel, "Dependability and Security Will Change Embedded Computing," IEEE Computer 41 (1): pp103-105 (2008).

[13] Andreas Heinig et al., "Using Application Knowledge to Improve Embedded Systems Dependability," http://www.usenix.org/events/hotdep10/tech/full_papers/Heinig.pdf

[14] Elisabeth A. Strunk and John C. Knight, "Functionality/Dependability Co-design in Real-Time Embedded Software,"

[15] Francesco Flammini, "Formal Evaluation of a Majority Voting Concept to Improve the Dependability of Multiple Technology Sensors," Journal of Physical Security 4(1), pp. 1-9, 2010.

About related Embedded System and SPD issues projects mentioned in TA

[16] CESAR - Cost-efficient methods and processes for safety relevant embedded systems, Artemis JU project, http://www.cesarproject.eu/

[17] CHARTER - Critical and High Assurance Requirements Transformed through Engineering Rigour, Artemis JU project, http://charterproject.ning.com/

[18] EMMON – Embedded Monitoring, Artemis JU project, http://www.artemis-emmon.eu/

[19] SMART – Secure, mobile visual sensor networks architecture, Artemis JU project, http://www.artemis-smart.eu/home.aspx

[20] IMSK – Integrated Mobile Security Kit, FP7 IP Security Project, http://www.imsk.eu/

[21] *ECRYPT II - European Network of Excellence for Cryptology II, FP7 NoE,* http://www.ecrypt.eu.org/

About Security and dependability with emphasis conceptual models

[22] A. Avizienis, J.-C. Laprie, and B. Randell. Dependability of computer systems: Fundamental concepts, terminology, and examples. Technical report, LAAS-CNRS, October 2000.
[23] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," TDSC, vol. 1, no. 1, pp. 11–33, 2004. [Online]. Available: http://csdl.computer.org/comp/trans/tq/2004/01/q0011abs.htm
[24] Arnaud Liefooghe, Lettitia Jourdan and El-Ghazail Talbi, "A software framework based on a conceptual unified model for evolutionary multiobjective optimisation: ParadisEO-MOEO," European Journal of Operational research 2010, Elsevier.
[25] E. Jonsson, "Towards an Integrated Conceptual Model of Security and Dependability," in Proc. of ARES'06. Los Alamitos, CA, USA: IEEE CS Press, 2006, pp. 646–653.
[26] Yudistira Asnar and Paolo Giorgini, "Multi-Dimensional Uncertainty Analysis in Secure and Dependable Domain," http://troposproject.org/files/ASNA-GIOR-10-ARES.pdf
[27] M. Kaaniche et al. "Modelling of Resilience of Large and Evolving Systems," International Journal of Performability Engineering, 4(2): pp. 153-168, April 2008.
[28] R. J. Rodrigueez, J. Marseguer, and S. Bernardi, "Modelling and Analysing Resilience as a Security Issue within UML," DISC FP7 project.
[29] K.S. Triverdi et al., "Dependability and Security Models," 7th Int. Workshop on the Design of Reliable Communication networks (DRCN 2009), Washington DC, October 2009.
[30] Erland Jonsson, "Towards an Integrated Conceptual Model of Security and Dependability," citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.1737&
[31] J.P. Holmegaard, J. Knudsen, P. Makowski and A.P. Ravn, "Formalization in Component Based Development," Mathematical Frameworks for Component Software. Liu, Z. & He, J. (eds.). World Scientific p. 255-281, 2006.

About Certifications and Verification of Embedded Systems and SPD related issues

[32] Gerwin Klein et al. "seL4: Formal Verification of an Operating-System Kernel," *Communications of the ACM,* 53(6), 107–115, (June, 2010)
[33] June Andronick et all, "Formal Verification of Security Properties of Smart Card Embedded Source Code," FM 2005, LNCS 3582, pp. 302-317, Springer-Verlag Berlin Heidelberg 2005.
[34] B. Adler et al., "From Model-Based Design to Formal Verification of Adaptive Embedded Systems," ICFEM 2007, LNCS 4789, pp. 75-95, Springer-Verlag Berlin Heidelberg 2007.
[35] Marco Anisetti et al., "Toward Certification of Services," http://infolab.uvt.nl/workshops/BSME-2010/papers/BSME-2010-AnisettiAD.pdf
[36] Philip Koopman, "Verification, Validation & Certification: Distributed Embedded Systems," http://www.ece.cmu.edu/~ece649/lectures/16_vvc.pdf
[37] CCRA org.– Common Criteria Recognition Agreement, http://www.commoncriteriaportal.org/

About Security and dependability for networked ESs with emphasis on the Wireless Sensor Networks: Node Layer targeted objectives

[38] Antonia Bertolino et al., "Dependability in dynamic, evolving and heterogeneous systems: the CONNECT approach," CONNECT (Emergent Connectors for Eternal Software Intensive networked Systems) FP7 FET Proactive IP Project, http://connect-forever.eu/index.html
[39] A. Taherkordi et al., Dependability Consideration in Wirless Sensor Networks Applications," Journal of Networks, vol. 1, No. 2, pp.28-35, December 2006.
[40] S. Attouche, S. Hayat, M. Staroswiecki, "An efficient algorithm for the design of fault tolerant multi-sensors systems," Proceedings of the 40th IEEE Conference on Decision and Control (CDC01), 2001.
[41] D.R. Raymond and S.F. Midkiff, "Denial-of Service in Wireless Sensor Networks: Attacks and defences," IEEE Pervasive Computing, pp. 74-81, 2008.

[42] EVITA - E-safety vehicle intrusion protected applications, FP7 project, http://evita-project.org/deliverables.html

[43] Michele Nogueira Lima, Aldri Luiz dos Santos, Guy Pujolle, “A Survey of Survivability in Mobile Ad Hoc Networks,” http://www.lip6.fr/lip6/reports/2007/lip6-2007-001.pdf

[44] Phillip Stanley-Marbell et al., “System Models in Wireless Sensor Networks,” Eindhoven University of Technology, ES Report, May 2008.

About Security and dependability for networked ESs with emphasis on ICT Next Generation Networks: Network Layer targeted objectives

[45] M. Al-Kuwaiti et al., “A Comparative Analysis of Network Dependability, Fault-tolerance, Reliability, Security, and Survivability,” IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 11, NO. 2, SECOND QUARTER 2009.

[46] T. Charles Clancy, Nathan Goergen, “Security in Cognitive Radio Networks: Threats and Mitigation,”

[47] S.C.Lingareddy et al., “Wireless Information Security Based on Cognitive Approaches,” IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.12, pp. 49-54, December 2009

[48] Jan Peleska, “Formal Methods and the Development of Dependable Systems,”

[49] Martin ARNDT, “Towards a pan European architecture for cooperative systems,” ETSI Status on Standardization, 2009,

About Security and dependability for networked ESs with emphasis on the Middleware: Middleware Layer targeted objectives

[50] Liu Jing Yong et al., “Middleware-based Distributed System Software Process,” International Journal of Advanced Science and Technology, volume 13, pp.27-47, December 2009.

[51] Markus Eisenhauer et al., Middleware for Wireless devices and Sensors,” 2010 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks SECON (2010).

[52] Jameela Al-Jaroodi, Alyaziyah Al-Dhaheri, “Security Issues of Service-Oriented Middleware,” IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, pp. 153-160, January 2011.

[53] Gang Huang, “SOAR: Towards Dependable Service-Oriented Architecture via Reflective Middleware,” International Journal of Simulation and Process Modelling, InderScience Publishers, 2007, 3(1/2): pp. 55-65.

[54] Stefan Poslad et al., “Middleware for semantic-based security and safety management of open services,” Int. J. Web and Grid Services, Vol. 1, Nos. 3/4, pp. 305-327, 2005.

About Security and dependability for networked ESs with emphasis on the Overlay Layer/Cross-Layer: Overlay Layer targeted objectives

[55] Jaime Galán-Jiménez and Alfonso Gazo-Cervero, “OVERVIEW AND CHALLENGES OF OVERLAY NETWORKS: A SURVEY, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011

[56] Kalpana Sharma and M.K. Ghose, “Cross Layer Security Framework for Wireless Sensor Networks,” International Journal of Security and Its Applications Vol. 5 No. 1, January, 2011.

[57] S. Paris, C. Nita-Rotaru, F. Martignon, and A. Capone, “EFW: A Cross-Layer Metric for Reliable Routing in Wireless Mesh Networks with Selfish Participants**,”** To appear in IEEE Infocom Mini Conferences (INFOCOM-Mini), Shanghai, China, Apr. 2011.

[58] Dilip Krishnaswamy et al., “A Cross-Layer Cross-Overlay Architecture for Proactive Adaptive Processing in Mesh Networks,” http://hpshiang.bol.ucla.edu/Publications/WiMeshXLayerXOverlayFinal.pdf

[59] Gustavo Carneiro et al., “Cross-Layer Design in 4G Wireless Terminal,” IEEE Wireless Communications April 2004, pp. 7-13.

[60] Christian Henke et al., “Scenarios for a Future Internet based on Cross-Layer Functional Composition,” http://www.future-internet.org/files/2010/Folien/Folien_Henke.pdf

[61] Marco Pugliese, Luigi Pomante, Fortunato Santucci, “Agent-based Scalable Design of a Cross-Layer Security Framework for Sensor Networks Monitoring Applications,” ICUMT 2009: pp. 1-7, http://www.webalice.it/marco_pugliese/SASN09.pdf

[62] Nicola Baldo and Michele Zorzi, “Fuzzy Logic for Cross-Layer Optimisation in Cognitive Radio Networking,” IEEE Communication Magazine, April 2008.

[63] Edward D. Moreno and Fabio D. Pereira, “Embedded System for Security Service Integration,” http://www.eatis.org/eatis2010/portal/paper/memoria/html/files/42.pdf

[64] J. Zhou et al., “State of the Art on Pervasive Service Computing,” http://202.120.40.15/~jzhou/research/publications/2010_ubihealth.pdf

[65] B.Padmaja Rani et al., “Architecting Secure Web Services using Model Driven Agile Modeling,” International Journal of Engineering Science and Technology, Vol. 2(9), pp. 4603-4609, 2010,

<u>About semantic ontology for SPD issues</u>

[66] Glen Dobson, Peter Sawyer, “Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web,” citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.5362...

[67] Carlos Blanco, et al.,"A Systematic Review and Comparison of Security Ontologies." In Proceedings of the 2008 Third International Conference on Availability, Reliability and Security - Volume 00: IEEE Computer Society, 2008.

[68] Helder Gomes, André Zúquete, Gonçalo Paiva Dias, “An Overview of Security Ontologies,” http://www.ieeta.pt/~avz/pubs/CAPSI09.pdf

[69] Bhavani Thuraisingham, “Security standards for the semantic web,” Elsevier, Computer Standards & Interfaces 27 pp.257–268, 2005.

[70] Andreas Ekelhart et al., “Ontological Mapping of Common Criteria’s Security Assurance Requirements,” http://publik.tuwien.ac.at/files/PubDat_186978.pdf

[71] Golnaz Elahi, Eric Yu, and Nicola Zannone, “A Modeling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations,” Conceptual Modeling - ER 2009, Lecture Notes in Computer Science, 2009, Volume 5829/2009, pp. 99 -114.

[72] Cataldo Basile, Antonio Lioy, Salvatore Scozzi, and Marco Vallini, “Ontology-based Security Policy Translation,” Journal of Information Assurance and Security 5, pp. 437-445. 2010.

[73] Atta Badii et al., “Semantic Resolution of Security and Context in the Hydra Middleware Architecture,”

<u>About Composability, SPD, Network</u>

[74] Daniel Stevenson et al., “ON THE SUITABILITY OF COMPOSABLE SERVICES FOR THE ASSURABLE FUTURE INTERNET,” In Proceedings of MILCOM 2007, October 29-31, 2007

[75] Stefan Schmid et al., *“TurfNet:* An Architecture for Dynamically Composable Networks,” Proceedings of IFIP Workshop for Autonomic communications (WAC 2004), NLE-PR-2004-22.

[76] J. Pujol, S. Schmid, L. Eggert and M. Brunner, *The Scalability Analysis of the TurfNet Internetworking Architecture*, IEEE Globecom, 2007

[77] Matthias Werne et al., “Composability Concept for Dependable Embedded Systems,” Proceedings of the International Workshop on Dependable Embedded Systems at the 22nd Symposium on Reliable Distributed Systems (SRDS 2003), Florence, Oct. 2003.

[78] Jan Richling, "Architectures and Composability Dependable Systems," https://flint.kbs.cs.tu-berlin.de/svn/DS2008/web/12-Architecture.pdf

[79] *SCALOPES* - SCalable LOw Power Embedded platforms, JU Artemis, http://www.scalopes.eu/

[80] Ivica Crnkovic, Magnus Larsson, "Concerning Predictability in Dependable Component based Systems: Classification of Quality Attributes," Architecting Dependable Systems III,, p pp. 257 – 278, Springer, LNCS 3549, Editor(s): R. de Lemos et al. (Eds.):, 2005.

[81] Arnaud Lanoix et al., "Enhancing Dependability of Component-based Systems," Proceedings of the 12th international conference on Reliable software technologies Springer-Verlag Berlin, Heidelberg 2007.

About Interoperability of heterogeneous networked ESs and networks

[82] G. A. Lewis et al., "Why Standards Are Not Enough To Guarantee End-to-End Interoperability," IEEE Computer Society 7th Conference on Compostion-Based Software Systems, pp. 164-172, 2008.

[83] U. Javaid et al., Toward Universal Convergence in Heterogeneous Wireless Networks Using Ad-hoc Conectivity," Proceedings of 9th International Conference on Wireless Personal Multimedia Communications(WPMC'06), San Diego, September 2006.

[84] Ina Minei and Julian Lucek, "MPLS Enabled Applications, Emerging Technologies and New Technologies," 2nd Eddition, John Wiley & Sons, 2008.

[85] Ivica Crnkovic, Ivano Malavolta, Henry Muccini, A Model-Driven Engineering Framework for Component Models Interoperability, Proceedings of the 12th International Symposium on Component-Based Software Engineering (CBSE 2009), Springer, LNCS 5582, Editor(s):Christine Hofmeister, Grace A. Lewis, Iman Poernomo, June, 2009

Smart Objects

[86] A. Dunkels and J.P. Vasseur, "IP for Smart Objects," *The Next Internet*. Morgan Kaufmann, 2010.

[87] IPSO alliance http://ipso-alliance.org/

[88] Telnor Objects Vision Paper http://ipso-alliance.org/ http://www.telenorobjects.com/media/1635/telenor%20objects%20vision%20paper%20_190609.pdf

[89] M2M: the Internet of 50 billion devices http://www.electric-words.org/m2m/win-win%204m.pdf

[90] Terje Jensen "Next Generation Network Architecture," pp. pp. 134-156, Telektronikk 2.2009.

[91] Kashif Nizam Khan, "State-of-the-art Study and Design of a Small Footprint Version of the COOS Plugin Framework," Master's Thesis Espoo, June 30, 2010