Enabling the business-based
Internet of Things and Services

(FP7 257852)

D5.2.1 Architecture for intelligence integration

Published by the ebbits Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2009.1.3: Internet of Things and Enterprise environments

Document control page

Document file: D5.2.1 Architecture for intelligence integration.docx
Document version: 1.0
Document owner: Ferry Pramudianto (Fraunhofer FIT)

Work package: WP5 – Architecture for intelligence integration
Task: T5.1 – Architectural analysis and description of the centralized and distributed intelligent service structured
Deliverable type: PU
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Amro Al-Akkad Andreas Zimmermann Christian Prause Ferry Pramudianto Markus Eisenhauer	2010-12-9	Table of Contents, Initial Requirements Clustering, Initial idea for sensor fusion, context, and control management
0.2	Ferry Pramudianto	2010-12-9	Sensor Fusion Architecture
0.3	Andreas Zimmermann Ferry Pramudianto	2010-01-10	Context Management Architecture
0.4	Andreas Zimmermann Mark Vinkovits	2011-02-1	LinkSmart Middleware
0.5	Ferry Pramudianto	2011-02-10	Control Management Architecture, Finalizing the first draft
0.6	Ferry Pramudianto	2011-02-20	Improve the deliverable according to the comment from IS
1.0	Ferry Pramudianto	2011-02-25	Improve the deliverable according to the comment from SAP
			Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Karol Furdik (IS)	2010-02-15	Approved with some comments on content and structure of sections.
Martin Knechtel (SAP)	2010-02-25	Approved with comments.

Table of Contents:

1. Executive summary

The ebbits project aims to develop architecture, technologies and processes, which allow businesses to semantically integrate the Internet of Things into mainstream enterprise systems and support interoperable real-world, on-line end-to-end business applications. It will provide semantic resolution to the Internet of Things and hence present a new bridge between backend enterprise applications, people, services and the physical world, using information generated by tags, sensors, and other devices and performing actions on the real world. Ebbits opens possibilities to offer a wide range of new business services based on orchestration of physical devices, software services, and people that are introduced as Internet of People, Thing, and Services (IoPTS).

The ebbits project follows IEEE 1471:"Recommended Practice for Architectural Description of Software-Intensive Systems" for specifying of the system design and software architecture, which defines core elements like viewpoint and view. In order to implement and execute this methodology, we follow the approach introduced by Rozanski and Woods (Rozanski and Woods 2005). Chapter 3 describes the process of the designing architecture in ebbits which is started by defining scope, engaging stakeholders, capture the first-cut concerns, then define the architecture. Then, the process to refine the architecture is also explained followed by six viewpoints that are recommended by Rozanski and Woods. Chapter 4 describes the initial requirements that are related to work package 5. The requirements are categorized in several categories that can be assigned to the tasks in WP5. The functional requirements include categories such as Transparent Communication, Context Awareness, Information and data logging, Multi Sensor Data and Information Fusion. The non functional requirements include security and performance metrics. Moreover, several stakeholders were identified including Operators, Technicians, End users of business applications, Software Developer and Integrators, Regulatory bodies, and Consortium members.

Chapter 5 describes LinkSmart middleware. It provides security, device discovery, semantic infrastructures, and data acquisition component. Security is provided by encrypting the messages being exchanged among web services and their consumers, and secondly by policy based access restriction to web service calls within the LinkSmart network. Several components must be redesigned to meet ebbits requirements such as context management and data persistence.

Chapter 6 contains the initial system design of distributed and centralized intelligence services which is discussed from the functional and deployment viewpoints. It provides a layered architecture that describes the relation of the new components to the LinkSmart architecture as well as to business applications such as data warehouse, reporting and business intelligence in general. The state of the art and the proposed architecture models for multi data sensor fusion, context, and control management are discussed from the functional viewpoint. We propose to follow the JDL model for processing data acquired from multiple sensors. The JDL model recommends a fusion process that includes 5 processing levels which process raw signals from multiple sources resulting in a situation assessment. The accumulated situation over time can be used to infer the context based on the context models defined by experts. High level context information can be derived from the past, present and future situations of the environment and the entities in it. This approach enhances LinkSmart to be more flexible and extensible since a high level context model can be reused and exchanged between systems. For the control management we propose a proxy based solution that is able to overcome the synchronization problem between the embedded and the PC world.

2. Introduction

The goal of ebbits project is to research and integrate Internet of Things(IoT) technologies in the business domain. The outcome of ebbits will allow business applications to incorporate physical objects, services and people into mainstream enterprise systems and support interoperable real-world, on-line end-to-end business applications. Dealing with a massive amount of heterogeneous devices and business applications, as envisioned in the IoT, ebbits will use semantic technology that allows automatic processing of information and autonomous collaboration among devices. Ebbits will open new possibilities to offer a wide range of new business services based on the orchestration of physical devices, software services, and people. The ebbits project introduces this as the Internet of People, Thing, and Services (IoPTS).

2.1 Purpose, context and scope of this deliverable

The purpose of this deliverable is to provide firstly the standard methodology to describe architectural designs that will be used in work package 5. Secondly, an architecture formalization of the initial system design within work package 5 will be based on the related requirements that we have gathered in work package 2. The architecture will initially describe how the main block components in work package 5 are related to LinkSmart middleware as the foundation of ebbits development activities and secondly how it is related to applications within the business domain. We also explain what the functionality of these main blocks (functional view) will be. Furthermore, the first design of the deployment (deployment view) will be proposed. In the next iterations, more architectural views will be explained in detail.

2.2 Deliverable Organization

This deliverable is organized as follows:

- Chapter 3 describes the methods and principles applied for software architectural design that follows standard IEEE 1471:"Recommended Practice for Architectural Description of Software-Intensive Systems".
- Chapter 4 lists the initial set of functional and non-functional requirements that are related to work package 5 of the ebbits platform.
- Chapter 5 reviews the LinkSmart Architecture that is related to the work package 5 in order to analyse the interfaces between the new components.
- Chapter 6 provides an overview of the initial system design of distributed and centralized intelligence services which will be discussed from the functional and deployment viewpoints.
- Appendix A illustrates the work package 5 initial requirements in a table that follows the Volere template.

3. Methodology

3.1 Software Architecture Design Fundamentals

We have based our process on the standard IEEE 1471 "Recommended Practice for Architectural Description of Software-Intensive Systems" which defines core elements like viewpoint and view. It also describes that stakeholders need to be involved and how to apply stakeholders needs to the architecture. This will be supported by the introduction of "architectural perspectives" which was introduced by Rozanski and Woods(Rozanski and Woods 2005).

3.1.1 Requirements & Architecture

We have established a process to gather requirements in a structured way as is laid out in deliverable D2.4 Initial Requirements Report(ebbits-consortium 2010b). For this we have conducted discussion rounds with focus groups of expert developers which possibly will use the LinkSmart middleware. The requirements were then prioritized and a fit criterion selected which allows to measure if a requirement is met or not. We have not only deduced requirements from the focus group discussions but also from other sources e.g. standards, best practices, each partner´s experience and so on. In this way we made sure that we collect a broad range of requirements to reflect the wide range of stakeholders.

Requirements and architecture influence one another. Requirements are an input for the architectural design process in that they frame the architectural problem and explicitly represent the stakeholders needs and desires. On the other hand during the architecture design one has to take into considerations what is possible and look at the requirements from a risk/cost perspective.

3.1.2 Viewpoints & Views

The IEEE 1471 standard defines viewpoint and view as follows:

Definition: Viewpoint and View

> *A **viewpoint** is a collection of patterns, templates and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint, and guidelines and principles and template models for constructing its views.*
>
> *A **view** is a representation of all or part of an architecture, from the perspective of one or more concerns which are held by one or more of its stakeholders.*

A viewpoint defines the aims, intended audience, and content of a class of views and defines the concerns that views of this class will address e.g. Functional viewpoint or Deployment viewpoint.

A view conforms to a viewpoint and so communicates the resolution of a number of concerns (and a resolution of a concern may be communicated in a number of views).

3.2 Software Architecture Design Process

3.2.1 Architecture Definition Activities

Rozanski and Woods have based the architectural design process on the following definition:

Definition: Architectural Design Process

> *"Architecture Definition is a process by which stakeholder needs and concerns are captured, an architecture to meet these needs is designed, and the architecture is clearly and unambiguously described via an architectural description." (Rozanski and Woods 2005)*

We have to consider a broad set of principles if the architectural design should be of good quality. We need to engage stakeholders to collect their concerns so the requirements can be balanced if there are conflicting or incompatible ones. The architectural design must allow for effective communication between all stakeholders and it must be structured to ensure continuous progress. Given the complexity of the project the design and also the process have to be flexible so we can react quickly to changing requirements and environments. Architecture should be technology neutral

but in the case of ebbits we have to ensure that it is applicable to a wide range of technologies that the ebbits platform may include.

The foundation for our process is the IEEE 1471 standard and we have used the process proposed by Rozanski and Woods, which is aligned to this standard and shown in:

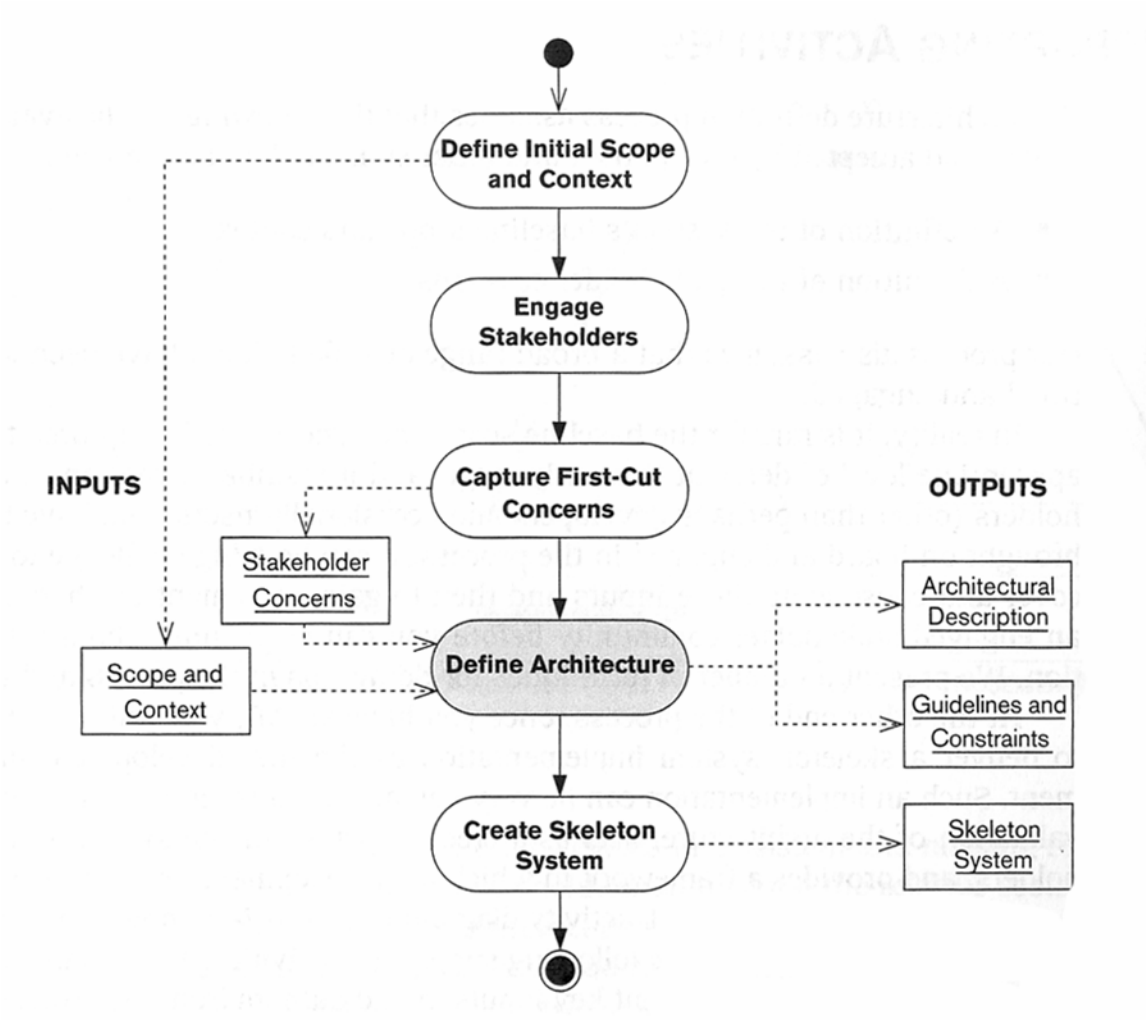

Figure 1: Architecture Definition Activities

The process implemented in the ebbits project clearly reflects this approach. We started with the initial scope and context that we acquired from the involvement of stakeholders in the process of the scenario development(ebbits-consortium 2010c)), the subsequent requirements process in WP2 (ebbits-consortium 2010b), and the state of the art analysis that we have documented in D5.1.1 (ebbits-consortium 2010a). The stakeholders or their representatives were included to express their needs and desires and capture quality properties that increase the success of ebbits platform. Those requirements from the discussion rounds together with requirements from other sources are the input for the current architecture design phase where we create a first draft of the architectural description (AD). Based on this architectural description, the first prototype has been created, which can be seen as a proof of concept system with minimal functionality on top. The experiences gained from these development efforts constitute a valuable source for the derivation of additional requirements and the revision of already existing ones. The following diagram reflects the details of the process:

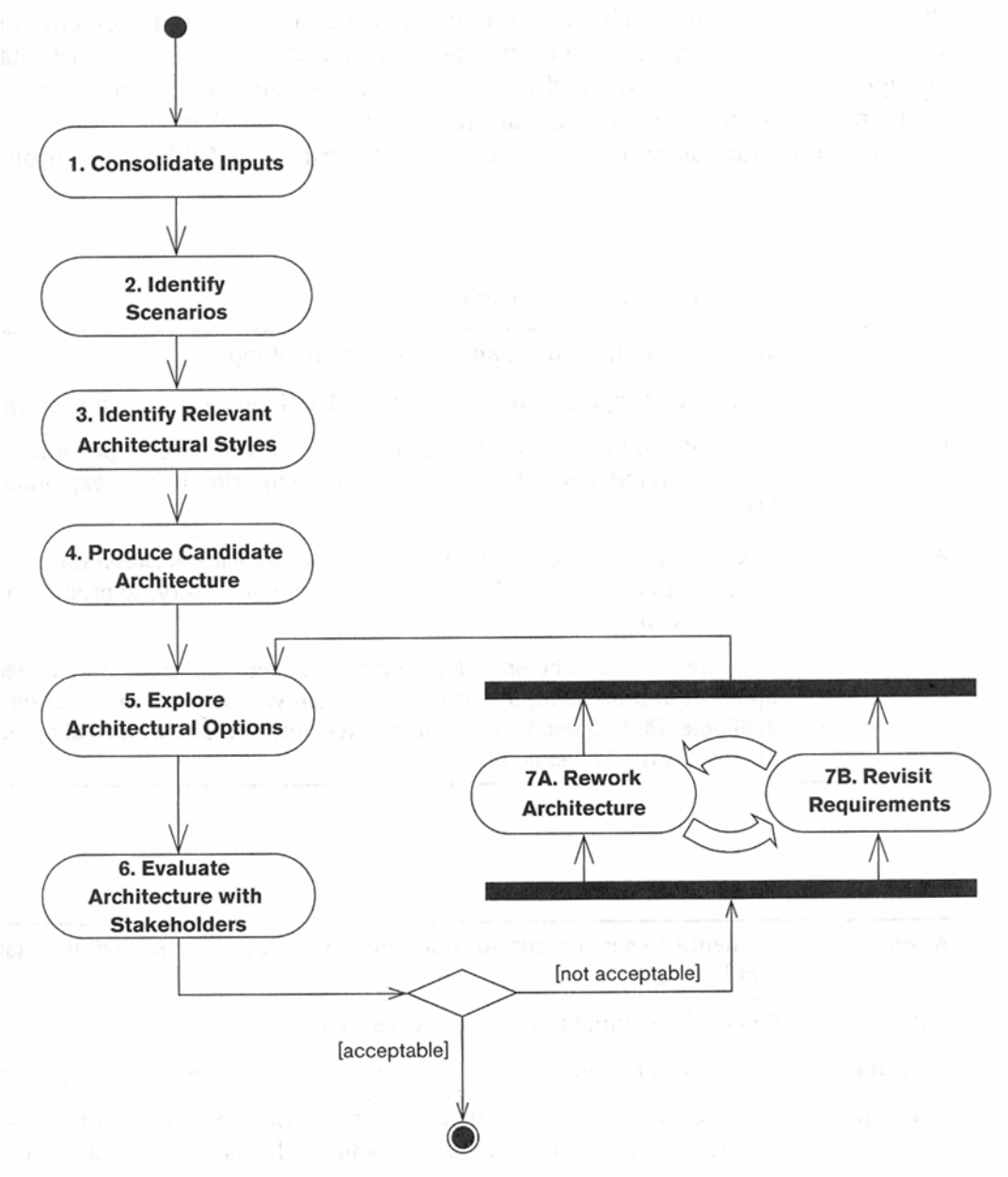

Figure 2: Architecture Definition Activities Details

Steps 1 and 2 are reflected in the requirements process and steps 3 and 4 were basically defined by the DOW. In the DOW we have decided to implement a middleware based on a service-oriented architecture (SOA) through the use of Web Services. With this as a framework the candidate architecture was set so we would only chose another architectural style if we would face insurmountable problems which are very unlikely.

The steps 5 to 7 (A and B) reflect our iterative approach on constantly refining the architecture and checking back with the stakeholders if the architecture meets their needs. After this iteration cycle the next steps of implementation and testing the revised architecture will follow but are not scope of this document.

3.2.2 Viewpoint Catalogue

The viewpoint catalogue proposed by Rozanski and Woods contains the following viewpoints:

Functional: The system´s functional elements, their responsibilities and primary interactions with other elements will be described. This is usually the most important

viewpoint as it reflects the quality properties of the system and influences the maintainability, the extensibility and the performance of the system.

Information: Describes the way that information is stored, managed and distributed in the architecture.

Concurrency: Describes the concurrency structure of the system and identifies components that can be executed concurrently and how this is coordinated and controlled.

Development: Describes how the architecture supports the development process.

Deployment: Describes the environment that the system will be deployed into and also documents the hardware requirements for the components and the mapping of the components to the runtime environment that will execute them.

Operational: Describes how the system will be operated, administered and supported while it is running and strategies and conflict resolutions will be documented here.

The following diagram shows how the views relate to each other.

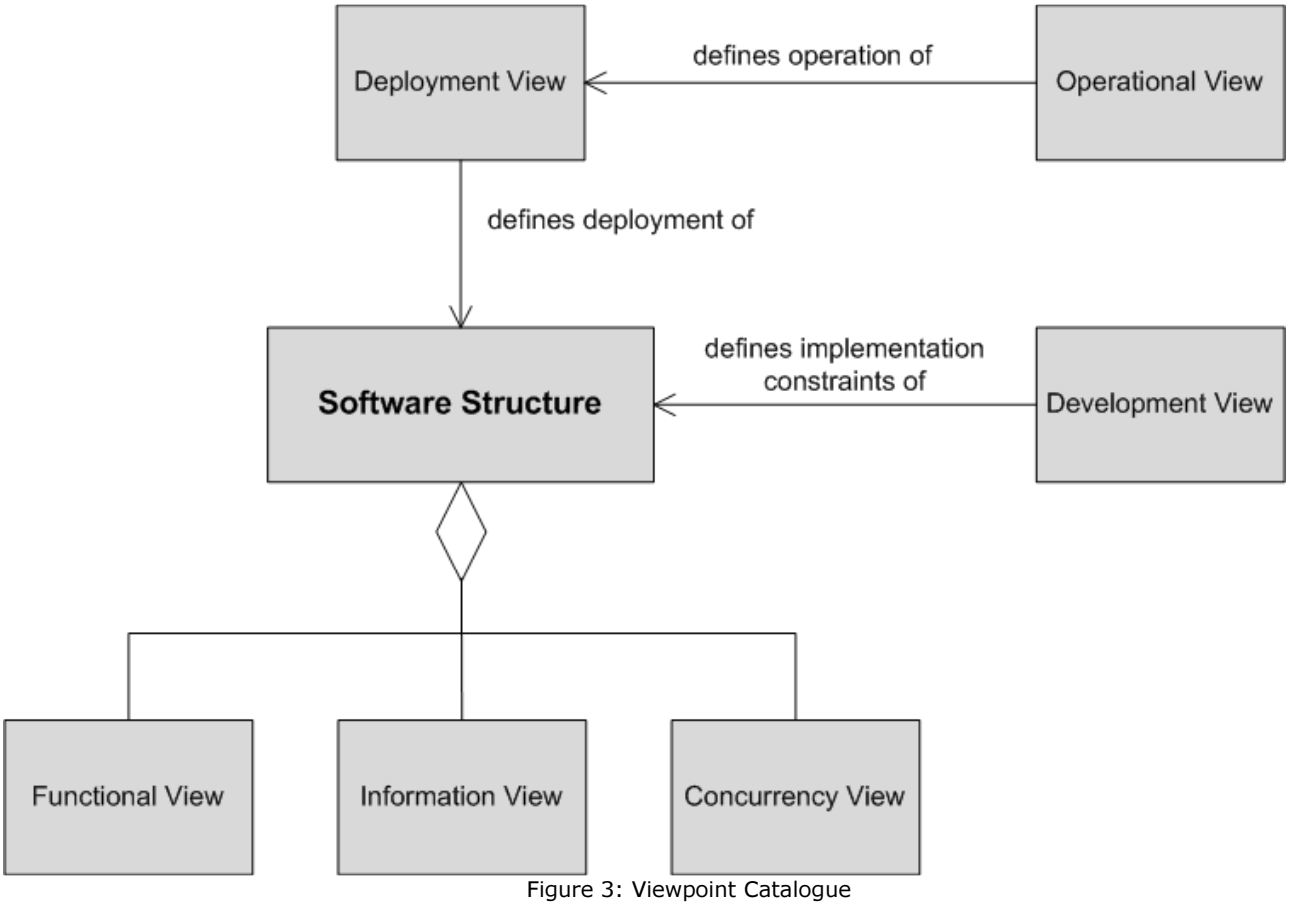

Figure 3: Viewpoint Catalogue

During the course of the project this document will be continuously and successively extended by additional views.

3.2.3 Architectural Perspectives

The term "Architectural Perspectives" was coined by Rozanski and Woods.

Definition: Architectural Perspective

*"An architectural **perspective** is a collection of **activities, checklists, tactics** and **guidelines** to guide the **process** of ensuring that a system **exhibits** a particular set of closely related **quality properties** that require consideration*

*across a **number** of the system's architectural **views**." (Rozanski and Woods 2005)*

The architectural perspectives ensure that quality properties are not forgotten in the process because the viewpoint and view approach per se does not explicitly consider those quality properties. But those properties are critical to the success of the project and to reflect them properly one usually needs cross-view considerations while the viewpoints are relatively independent.

Rozanski and Woods propose the perspectives on security, performance, availability, maintenance, location, regulation etc. Not all combinations of perspectives and views are needed and artefacts created according to those perspective/view combinations need to be carefully chosen. The perspectives and the combination possibilities with views that Rozanski and Woods propose are shown in the following diagram:

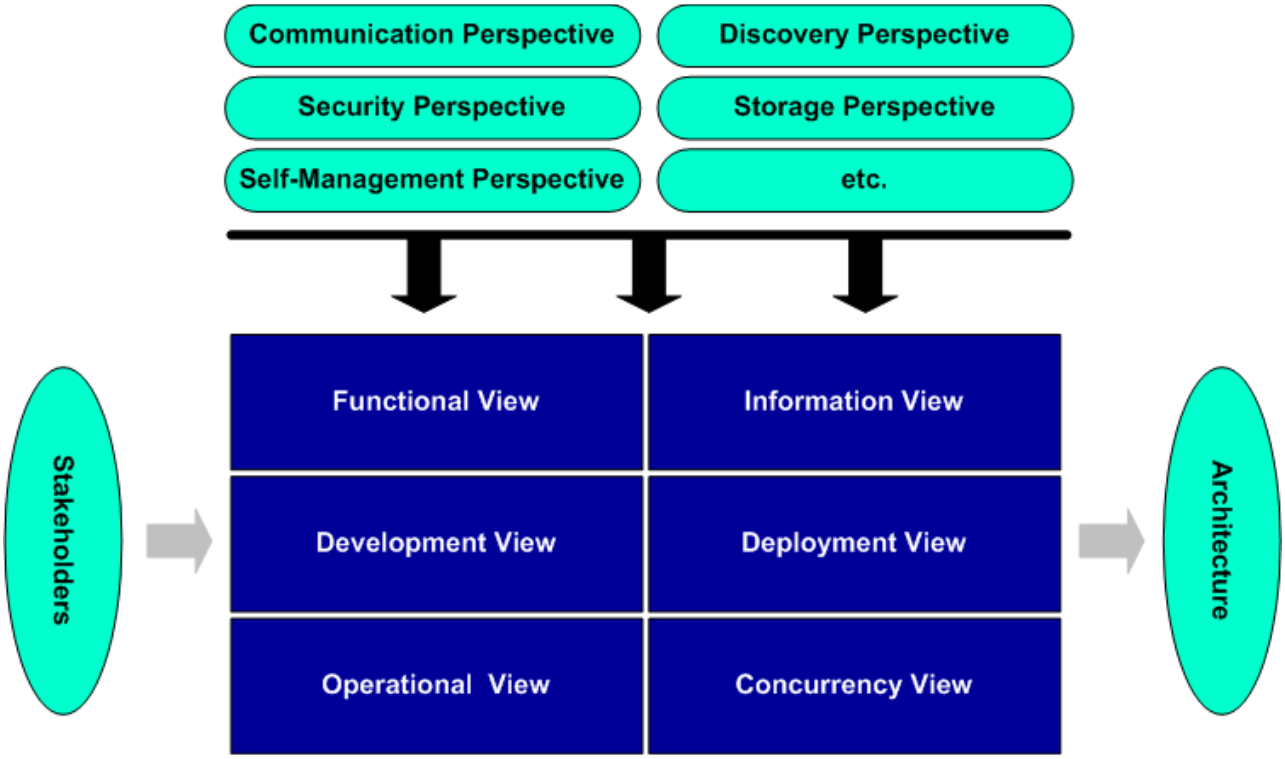

Figure 4: Architectural Perspectives and Views

4. Requirements

The purpose of this section is to provide top-level user requirements of future use of the ebbits platform that are related to work package 5 in the two selected areas, Automotive Manufacturing and Food Traceability. During the initial discussions of the project objectives and the work plan we decided to take a slightly different approach to the Scenario Thinking method for the first iteration. We interviewed the user partners as experts in their field and building the vision scenario workshops around them, rather than involving external experts. It was agreed that the proposed business applications are sufficiently anchored in the present to make this process adaptation viable. To further benefit and expedite the requirements elicitation process it was agreed to organize combined Scenario Workshop/Focus Group sessions involving user partners as well as developer partners. The detail methodology and requirements are documented in the *D2.4 Initial Requirements Report*.

The requirements related to WP5 are clustered in several categories that correspond to the tasks in WP5. These clusters have been introduced in D2.4(ebbits-consortium 2010b) and now have again been improved to meet the objectives of WP5 in general. The functional requirements include (the ids of the requirements have been updated in the GForge database due to data migration at the beginning of the project):

- Transparent Communication
  - #93 bring data from Fieldbus network to Ethernet network (#65 in D2.4)
  - #44 Farmers are able to retrieve optimized models from research (#19 in D2.4)
  - #27 Product-related information should be represented in a machine-readable format (#6 in D2.4)
  - #45 System can feed the farms data to research(#20in D2.4)
  - #130 Item identification system should provide open interfaces to other systems (#81 in D2.4)
- Context Awareness
  - #134 Ability to self-adaptation (#85 in D2.4)
  - #157 Different Views on the Data is necessary (#108 in D2.4)
  - #139 Support runtime reconfiguration(#90 in D2.4)
  - #49 Access to energy-related information from production machines needs to be provided. (#23 in D2.4)
  - #47 Resilience and adaptable to environment condition changes(#22 in D2.4)
  - #75 System should aware of what which livestock are in the building(#47 in D2.4)
  - #81 System should show Energy Cost for different granularity of production processes(#53 in D2.4)
  - #103 automatic calibration(#66 in D2.4)
- Information and data logging
  - #159 End-users need to be able to manage their distributed data(#110 in D2.4)
  - #64 Logging of Quality related information of each Manufacturing Part(#37 in D2.4)
  - #39 Retrieve manufacturing data history of any relevant event during production(#14 in D2.4)
- Control Management

- #36 Controlling of machines/stations in manufacturing plant remotely(#11 in D2.4)
- #135 Protection of System Integrity(#86 in D2.4)

- Multi Sensor Data Fusion
  - #141 Report errors in devices(#96 in D2.4)
  - #155 Synchronization of Acquired Data is necessary(#106 in D2.4)
  - #91 filter/fusion information for each operational process(#63 in D2.4)
  - #92 early maintenance notification when needed(#64 in D2.4)
  - #66 correlate problems found with production batches(#39 in D2.4)
  - #35 Hazardous Environmental Monitoring of Manufacturing Plant(#10 in D2.4)
  - #50 Filtering to Obtain relevant Information(#24 in D2.4)
  - #43 Aggregating collected sensor data at a central point(#18 in D2.4)
  - #131 Support fuzzy or probability concepts for reasoning(#82 in D2.4)
  - #67 automatic analysis of cross enterprises product life cycle data(#40 in D2.4)
  - #78 system should provide location tracking of the stocks/livestocks(#50 in D2.4)
  - #79 location tracking should be implemented as independent app(#51 in D2.4)
  - #154 Aggregate data from various data bases and sources(#105 in D2.4)
  - #109 recognition of energy wasting behaviours(#72 in D2.4)

The non-functional requirements include:

- Security and Privacy
  - #82 Protection to sensitive information(#54 in D2.4)
  - #72 officials have a back door access to highly important information(#45 in D2.4)
- Performance
  - #140 Transparency of device performance(#91 in D2.4)
  - #138 Distributed Intelligence should not lead to resource-heavy systems(#89 in D2.4)

4.1 Functional

Some of these requirements are shared with other work packages as well as have been covered by LinkSmart middleware that is presented in section 5.

4.1.1 Transparent communication

Transparent communication requires a component that is able to deliver data and information that can be understood by another machine. This also requires a dissemination strategy that involves inter-enterprise communication as envisioned by the traceability scenario in ebbits. In the manufacturing scenario, the communication will also involve several manufacturing sites that are connected through virtual private network within the internet. The communication among several enterprises and manufacturing sites involves different systems and data formats that are influenced proprietary formats, culture and localization.

Heterogeneity of the communication protocols is covered by the use of standard WS-I web services[1] in an service oriented architecture of LinkSmart. The integration of physical world sensor and

[1] to http://www.ws-i.org

actuators into the ebbits platform will also be covered in WP-8. WP-5 is then responsible to filter and transform the data that comes out from the network layer and going into the application. The transformation and filtering will be initially done in the multi sensor fusion components aiming at delivering a higher quality of information. The sensor fusion components will achieve sensor readings with higher degree of confidence. These sensor readings can be disseminated directly to interested application for offline analysis purposes.

This cluster of requirements recapitulates the requirement: #27, #44, #45, #93, #130

4.1.2 Sensor Data Fusion

Decision making process must be supported by the decision makers' awareness of the current situation. Situation awareness can be augmented by providing users with meaningful information from the existing data. This transformation process from sensor data into meaningful information is known as sensor data fusion.

Data fusion also influences the means how the sensed data is transformed into useful information for the users in order to increase the situation awareness. On the other hand, data fusion will also be used by the context aware applications to infer the action needed in a context of the current situation.

In the requirement elicitation processes, the users mentioned that they want to optimize the energy consumption within the production area and improve accuracy and time needed for the traceability of food. Thus in production area, data such as the energy consumption, operational processes, production material, as well as In food traceability, data such as ingredients of livestock's feed, medical history of the livestock, meat quality, breeding data, and regulations must be fused and presented to the users to support them in taking decisions.

Another requirement that is a classic problem of sensor fusion application is an indoor location tracking. In the manufacturing scenario, Comau revealed that the locations of goods being used are not yet always known as they are not tracked electronically. This problem is also found in farms, as pigs are let loose in a farm building, the farmers cannot identify the pigs individually anymore. Although they use markers, the marks can dissolve over time therefore the farmers would like to have an electronic tracking to monitor the position of specific pig.

This cluster of requirements constitutes the requirement: #35, #36, #43, #50, #66, #67, #78, #79, #92, #109, #131, #141, #154.

4.1.3 Context Awareness

The users that we interviewed stated that information should be aggregated and personalized according to the users or user profiles since each stakeholder may require different kind of information as well as different level of detail. This is highly desired to avoid users being overwhelmed by irrelevant data.

Although, the presentation of the information to the users is out of scope of WP-5 since this would be covered by the human computer interface components such as industrial HMI module, ERP System, etc. However, the preparation of the information from the raw data is very much relevant for the goal of WP-5. Therefore the components in WP-5 must be able to aggregate data from distributed sources and prepare the require information. Data preparation in different level of the network can distribute the workload to different nodes and avoid a single point of failure.

The Self-management requirements can take advantage of the context awareness to react to any mal functions that are detected. Self-management aims at minimizing maintenance costs as much as possible by letting the systems take care of themselves. Most of the calibration and maintenance are now done manually by technicians that consumes some time which in turn will costs some losses in the production. Self-management includes self-configuration, self-adaptation, self-diagnosis, and self-protection. This requires a continuous monitoring for self-diagnosing the system conditions and state, in order to detect errors log device events.

This cluster of requirements constitutes the requirement: #47, #49, #75 #81, #103, #134, #139, #157.

4.1.4 Information Logging

Data that is produced by sensors in production domain must be logged for different reasons - for instance for process optimization, data are analyzed offline and when problem is identified, data is analyzed to trace the source of the problem and the affected products. So far massive amounts of sensor data are logged in a database with a static entity model. This approach raises problem when data must be aggregated with another system, therefore we will enhance the information logging technique using semantic technology. This will allow any data structure mismatch be semantically resolved.

This cluster of requirements constitutes the requirement: #39, #64, #159.

4.1.5 Control Management

The current situation of control management in manufacturing and farms is automated, and the users can control and configure the automated processes. The users would like to keep automated processes with minimum maintenance effort as described in 4.1.3 and also have a possibility to configure the control remotely through wireless devices.

This cluster of requirements constitutes the requirement: #36.

4.2 Non functional

4.2.1 Security

Since data is shared among enterprise, everyone should be able to protect their data and able to control what they want to share. The users must be able to define policy and rules in order to grant and restrict access to their data. They also need to be sure that the security mechanism is reliable and guaranteed because otherwise they will have doubts to share their data.

This cluster of requirements constitutes the requirement: #72, #82, #135.

4.2.2 Performance

In terms of performance, the users requested that the system should be responsive and able to handle a large number of data in an acceptable amount of time. The users would also like to have a constant feedback when the system is processing something, so they can be sure that it is working.

This cluster of requirements constitutes the requirement: #138, #140.

4.3 Stakeholder Analysis

For the definition of requirements traditionally the user was taken as a reference. One could argue what exactly the term user includes but it is obvious that not only the requirements of the user have to be analyzed. The software has to be developed, run, administered, maintained, and monitored; in parallel, it needs to abide to certain standards or regulations. Each of these aspects is of interest to different people which not necessarily are using the system at all. We refer to these groups of people as stakeholders. We use the following definition (taken from IEEE 1471):

***Stakeholder**: An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.*

From ebbits scenarios and use cases, we identified the following Stakeholders:

- **Operators**

  Operators are people whose job is to operate the machining and equipments on the production lines, farms, shop floors. Their main activity is to start up and shut off the shop floor, supervise any automation components and report to the technicians if there is malfunction, and supervise any events that could violate the safety protocols.

- **Technicians**

The technicians are people who maintain the running systems in manufacturing sites, production lines, and farms. They are responsible to monitor and repair any malfunctions that happen on site. Their main activities include calibrating system parameters, design and deliver diagnostic logging mechanisms as well as and analyze the logged information.

- **End users of business applications**

  This stakeholder type includes people who are working in the management levels and use enterprise resource planning software on their daily activities.

- **Software Developer and Integrators**

  These are people that develop software (on ebbits) to integrate data and information of any entities on the shop floor and farms into manufacturing execution systems and enterprise resource planning systems.

- **Regulatory bodies**

  These are people who make regulations nationwide, EU-wide for industry procedures and safety in order to protect the end consumers. Any new regulation could influence the running operational procedures within the enterprise.

- **Consortium members**

  The consortium members consist of several types, firstly the researchers within ebbits consortium who would like to explore state-of-the-art technology for ebbits purposes as well as for their exploitations. Secondly, the domain partners who want to exploit ebbits for their business.

5. Overview of LinkSmart Functional View

In this overview, only the functional view of the LinkSmart middleware and the WP5 related components are discussed so that the missing functionalities that we may need in WP5 can be identified. Further views will be discussed in the next iterations of this deliverable as different views of ebbits architecture will be defined. Figure 5 shows the functional view of LinkSmart middleware and its relation to the physical communication layer and applications built on top of LinkSmart middleware. The concept of middleware in distributed systems is often taken to mean "the software layer that lies between the operating system and the applications on each side of the system" (Krakowiak 2003). Another characterization in terms of the ISO OSI stack (Day and Zimmermann 1983) is that middleware provides protocols that run on top of the transport layer and provide services to the application layer (Tanenbaum 2008).

LinkSmart as a middleware framework (Formerly known as Hydra (Eisenhauer, Rosengren et al. 2009)) defines an abstraction layer on top of heterogeneous communication protocols. It provides services to application developers, hiding the complexity of underlying device specifics. In LinkSmart, service interfaces are decoupled from the network protocol. For instance, LinkSmart utilizes OSGi Service Platform for local calls between Java-based modules and for remote calls SOAP over various network protocols such as HTTP, UDP, and Bluetooth.

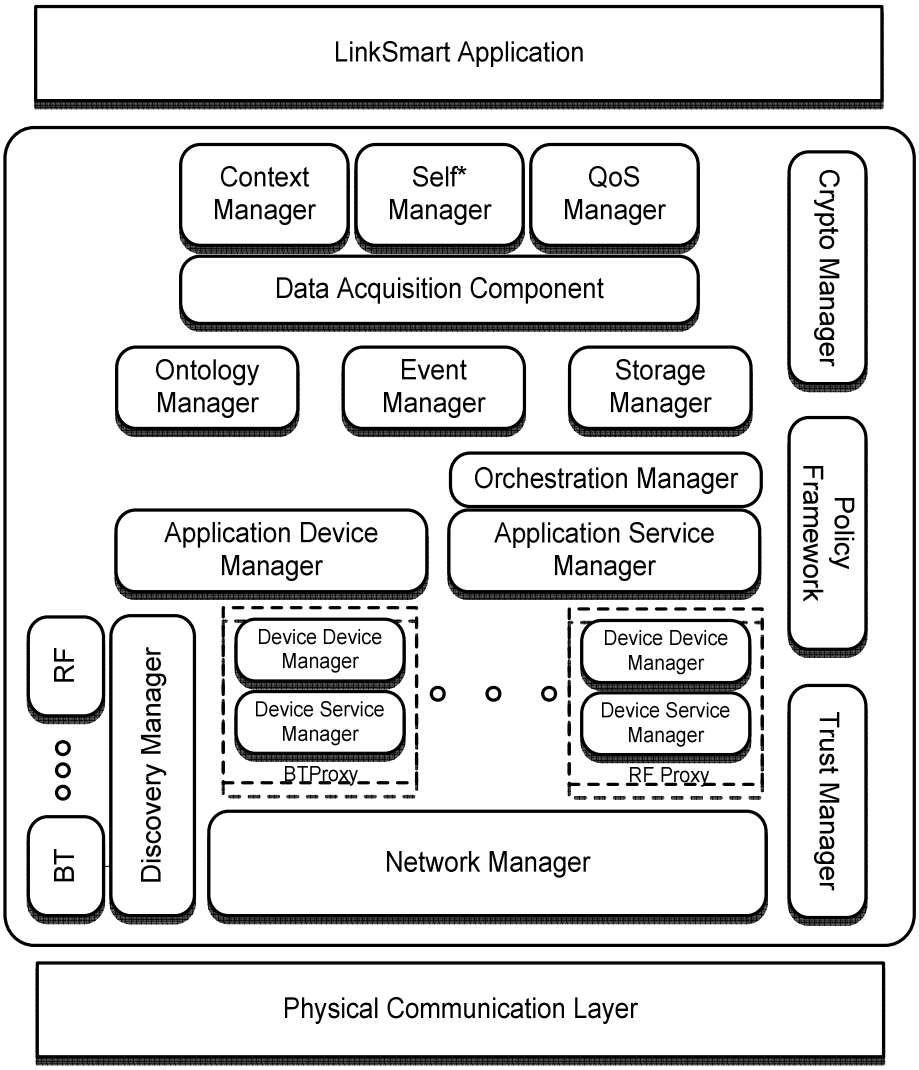

Figure 5: LinkSmart middleware architecture

Components inside the dotted square comprise the middleware. This figure clearly visualizes that LinkSmart is located between the physical communication layer and the application layer. What we

meant by physical layer in this sense is any means of communication protocols that can be abstracted by LinkSmart.

The middleware architecture follows strictly a service-oriented and component-based design adhering to the principles of loose coupling and separation of concerns. Figure 5 shows that LinkSmart consists of several components called managers. Each manager encapsulates a set of operations and data that realize a well-defined functionality. Thus, LinkSmart offers a large collection of reusable basic software components to application developers. While certain managers are mandatory, other ones may be used depending on the application's requirements. LinkSmart's component-based approach facilitates the development of scalable applications by plugging in certain functionality only when it needed. For example, the Network Manager is a mandatory component in any LinkSmart application but Context or Self* Managers are optional.

The LinkSmart SoA is implemented with WS-I (WS-I 2006) conformed Web Services based on either Java or .Net providing interoperability among different systems and platforms. Java based components make use of the OSGi[2] service platform as it represents a comprehensive framework for the development of modular and extensible applications.

Besides such architectural considerations, LinkSmart introduces the distinction of device developers and application developers allowing developers to best apply their expertise to specific tasks in pervasive application development. A device developer is responsible for connecting any kind of networked device to the LinkSmart middleware, exposing its functionalities as LinkSmart conformant services (see Section 5.2). Once integrated, the application developer can then transparently employ this device in his LinkSmart application (see Section 5.3.2).

5.1 Infrastructure

5.1.1 Overlay P2P connection

LinkSmart facilitates communication among devices via a P2P overlay network that is based on JXTA[3]. The basic LinkSmart component enabling network communication is the Network Manager. It is the incoming and outgoing point of information in a LinkSmart network. The network manager is not a centralized component in the network topology, in contrast it is deployed on any capable device (see Section 5.2.2). It implements SOAP Tunneling(Lardies et al. 2009) as Web Service transport mechanism, which allows LinkSmart devices to communicate securely even through firewalls or NATs. SOAP Tunneling decouples web services from physical addresses such as IP address. To the client application, all requested services seem to run local, though the network manager handles to reroute service invocations to the real physical address, where the actual service runs.

Inside a LinkSmart network, unique LinkSmart IDs (HIDs) identify all devices and services. Each Network Manager keeps a hash map mapping all available HIDs to their respective physical address. In order to deal with services that occur after disappearing due to changing locations or connection errors this hash map needs regularly an update.. All communication that happens between devices has to go through the respective Network Managers and SOAP Tunnels to reach its destination.

Applying such a comprehensive communication approach brings with it several advantages: A pervasive distributed network infrastructure allows for efficient resource sharing, fault tolerance when nodes break down, and ubiquitous access to the network.

The benefit of using JTXA is gaining platform independency and interoperability among different network protocols. Further, it provides support for state-of-the-art security concepts in P2P networks like authentication, authorization and integrity.

5.1.2 Semantic Services

One of the key components in the LinkSmart middleware is the device ontology. This ontology stores all information and knowledge regarding devices and device classes. The Device Ontology models

[2] http://www.osgi.org/
[3] https://jxta.dev.java.net/

specific features of devices such as: hard- and software capabilities of a device, device's services and events, device's state-machines, device's security capabilities, Quality-of-Service properties, and device's discovery information. After devices have been discovered, the relevant information can be obtained from the Discovery Manager. The Discovery Manager uses UPnP profile[4] to advertise device description as well as their events. Alternatively, Discovery Manager also uses SAWSDL[5] and WSDL[6] for describing web services that do not relate to any device.

One of the basic features of knowledge models comprises the ability to answer the queries. The Ontology Manager provides specific querying functionality accessible from specific Web-Service methods. The queries must be formulated in SPARQL[7]. The Ontology Manager also implements a general interface, which enables developers to retrieve the result for any SPARQL query.

5.1.3 Security

SOAP messages exchanged between network managers are encrypted using an asymmetric security functionality that requires keys and certificates. These are stored in a *java keystore*[8] of each Network Manager and protected by passwords. Before a secure message is going to be exchanged, the certificates between these two managers have to be shared among the entities. Then, in order to verify a certificate or key of a Network Manager, the module uses *aliases,* which correspond to the HIDs of Network Managers that the messages are sent to and retrieved from. When a message needs to be verified or encrypted, the sender (encryption) or receiver (verification) uses the public key of a specified manager. Figure 6 shows how the secure data is sent via the involved managers:

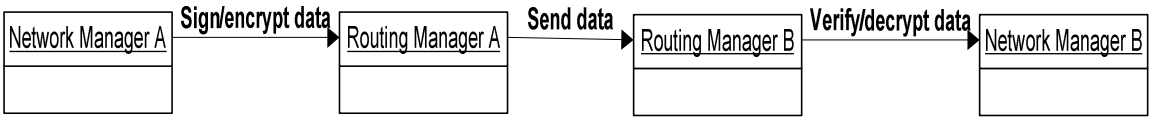

Figure 6: LinkSmart's secured message transmission

LinkSmart also provides a set of cryptographic algorithms that can be used by application developers to secure the context information going through the middleware (Hoffmann et al. 2007).

5.1.4 Distributed Storage

LinkSmart allows components to store data remotely. The Storage Manager is responsible for providing the interface that allows devices and application store and retrieve data and files in volatile as well as persistence storage. The Storage Manager provides different storage quality and with different performance as well as security requirements. For instance, Intermediate data might be stored without additional redundancy. In contrast, sensitive data might be stored inside a reliable storage array or has to be distributed redundantly over multiple disks. Accessing remote storage devices from a resource-constrained device introduces increased response times. Hence, the Storage Manager has to be able to compensate the storage variety. The Storage Manager also provides different levels of security in order to secure sensitive data. For instance, small data can be encrypted before it is stored. On the other hand, encrypting large data might slow down the system performance. Thus, this kind of data will be scattered over multiple physical storage, while their security is guaranteed by applying internal HW-Seeds in the used hash-functions.

5.2 Sensor & actuator abstraction

5.2.1 Device classification

Before an application developer can start building a system of interconnected heterogeneous devices with LinkSmart, s/he must have service level access to these devices. Typically sensors and actuators neither are capable of offering extensively interoperable service interfaces nor do they

[4] http://upnp.org/sdcps-and-certification/standards/sdcps/
[5] http://www.w3.org/2002/ws/sawsdl/
[6] http://www.w3.org/TR/wsdl
[7] http://www.w3.org/TR/rdf-sparql-query/
[8] This class represents a storage facility for cryptographic keys and certificates.

usually communicate over commonly used protocols. This is where the device developer comes into play. He is responsible for developing a software proxy that acts as a bridge between the device's communication protocol and the LinkSmart network. Such a proxy's job is twofold: Downward, it has to understand the device's communication technology and the format of the data exchanged. Upward, towards the LinkSmart network, it has to provide the translated device functionalities as Web Services.

LinkSmart as a generic middleware aims at allowing developers to integrate a wide range of different devices, from sensors over mobile phones to powerful computers. To provide a better overview over devices and their capabilities LinkSmart introduces device classifications concerning a device's network capabilities and the possibility to deploy parts of the LinkSmart middleware on that device. Resulting from this classification and the goal to deploy LinkSmart on as many device classes as possible, we also describe an approach to enable lightweight web services for less powerful devices.

The LinkSmart network architecture is based on IP networks with the communication scheme based on Web Service calls. If a device's communication protocol does not implement the IP layer, it will need means to be integrated in the LinkSmart network. The way this is done depends on the device's capability to host LinkSmart components. For this LinkSmart identifies the following device classes:

D0 devices are not able to host the minimally required subset of the LinkSmart middleware and do not support IP communication. Thus, they need a proxy running on a more powerful device to manage the communication and data transfer of the D0 device and expose its functionality as a Web Service. D0 devices are typically legacy devices with very limited power in terms of processor and memory using communication protocols like Bluetooth[9], ZigBee[10], IrDA[11] or Serial RS-232 among others. Sensors and actuators are D0 devices.

D1 devices cannot host the LinkSmart middleware but do implement IP communication and are suitable for running embedded Web Services. PDAs and mobile phones are examples of D1 devices.

D2 devices can host the LinkSmart middleware but do not implement IP communication. Thus, communication needs to be bridged by a device that is capable of IP. Some PDAs are examples of D2 devices.

D3 devices are able to host the LinkSmart middleware and provide IP support. Examples of D3 devices are powerful mobile phones, personal computer or laptops.

D4 devices are D3 devices that host proxies for D0 and D1 devices.

[9] https://www.bluetooth.org
[10] http://www.zigbee.org
[11] http://www.irda.org/

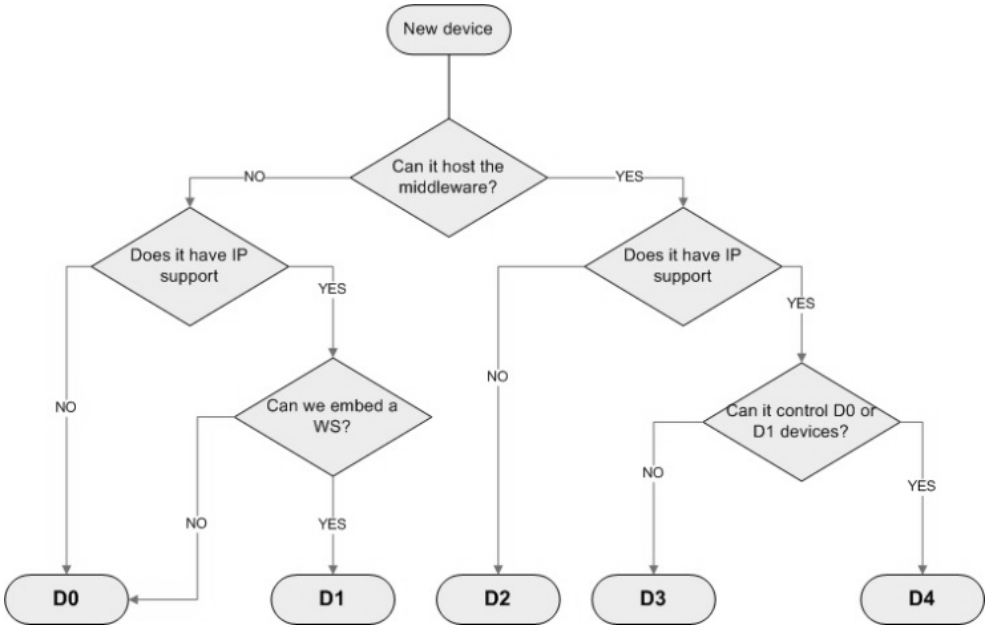

Figure 7: Device Classification Decision Flow Chart

Figure 7 shows a decision flowchart to help developers decide how they should develop a LinkSmart device. First, we have to check whether the device can host the LinkSmart middleware or not. If the device is not powerful enough, it is a D0 or a D1 device. If the device can host a web service and has IP communication capabilities, it is a D1 device. Otherwise, it is a D0 device. On the other hand, if the device can host the middleware, we have to check if the device in question supports IP communication. If the answer is negative, we have a D2 device. If the answer is positive and the device can control D0 and D1 devices in the system, we have a D4. Otherwise, it is a D3 device.

5.2.2 Lightweight web service for D1

Devices from class D1 lack the ability to host the LinkSmart middleware inside OSGi or .NET environments. Nevertheless, such devices are programmable and support IP-based communication, hence they are basically able to run embedded Web Services. LinkSmart introduces ServiceCompiler (Hansen et al. 2008), a Java-based web service compiler that generates web service code for different target platforms and protocols. Currently ServiceCompiler supports J2SE and J2ME platforms and implements SOAP communication over TCP, UDP or Bluetooth. ServiceCompiler integrates itself perfectly in LinkSmart's model driven development approach, as it utilizes semantic information provided by the device ontology to generate the respective services. Besides Web Services that allow a device to communicate inside a LinkSmart network, ServiceCompiler also generates UPnP service code, enabling a device to be discovered inside the network.

ServiceCompiler eases the task of device developer to integrate devices into LinkSmart network.

5.2.3 Sensor & actuator resource management

Resource management is one of the most important factors for wireless sensor nodes. LinkSmart assumes that processor and memory resources will be maintained by an operating system running on the nodes. However, LinkSmart supports energy management. Each LinkSmart device provides an energy service, which allows applications to receive information on energy consumption and to enforce energy usage policies (e.g.: scheduling when the device should be put on sleep, wake up and standby). In order to coordinate global energy consumption, the energy policy monitor interprets energy policies and executes a set of services depending on the respective policy. Devices can be selected by explicitly referring to name/id or by selecting criteria expressed over their energy profiles and other device descriptions in the device ontology. Various run-time consumption restrictions can be applied to the rules, for instance a specification of thresholds for overall consumption per group or for subsets of devices, and the actions taken such as disabling devices.

5.2.4 Data Acquisition

Data acquisition in LinkSmart is responsible to serve several purposes. For instance, delivering sensed data and its characteristic such as unit and precision, hardware and software resource data, platform information, user profiles and preferences, security data and state status.

Information from sensor nodes can either be pushed to the application or polled by the application. LinkSmart provides both of the methods via the Event Manager and the Data Acquisition Component. The application can use one method exclusively or combine both methods depending on the application context. For instance, if the application requires collecting samples every interval of time, the Data Acquisition Component should be used to poll the information out of the sensor network. When an application only has an interest to be informed about specific events that it registered on, while it does not occur very often, e.g. for energy saving reasons, it should only use event-based method.

Since sensed data can be unreliable caused by environment condition, LinkSmart allows developers to define a plausibility check through regular expressions. Plausibility check discards abnormal values when defined by the developers. For instance, if developers defined that the possible value for a room temperature is between -15 to 100 degrees Celsius, when a sensor delivers a value of 200 degrees, it will be discarded.

5.3 Model Driven Application Development

5.3.1 Sensor & actuator discovery.

Discovering available devices in the network is important during development and at runtime. Nevertheless, heterogeneous discovery protocols in an application are hard to maintain. Thus, in order to have a uniform discovery mechanism that is transparent to the application, LinkSmart uses two-level discovery. The first level discovers devices and services in the local network that are only detectable by their specific discovery protocols (e.g.: Bluetooth, ZigBee, WS-Discovery). After the first level is completed, LinkSmart tries to assign a classification to each device by matching the information that has been obtained from their native discovery protocol with information stored in the Ontology Manager. When a closest class of a device has been found, LinkSmart generates an UPnP proxy for the corresponding device. The second level of the discovery protocol tries to find any UPnP device in the local network, which carries a LinkSmart signature. When necessary, Discovery Managers in the same LinkSmart network communicate with each other to exchange a list of devices, thus devices connected to other gateways, regardless of their physical locations, can still be detected by any application connected to the same LinkSmart network. We also consider using WS-Discovery for the second level discovery in the future since it has become a standard for discovering Web Services.

5.3.2 Application Development

An advantage of LinkSmart middleware is that the managers were designed to be able to work in distributed manner. LinkSmart components themselves operate inside a service oriented architecture system that uses Web Services to communicate to among them. The managers are also able to communicate directly to each other in an OSGi environment, which eliminates the communication overhead caused by SOAP messages. This allows components to be either locally as well as in a distributed manner. For instance, the Quality-of-Service Manager is deployable in the same OSGi environment as the application, or can run on an independent server running on the network that is accessible via web services.

LinkSmart takes advantage of semantic web services, which makes it possible for application developers to address devices in an abstract manner, which decouples the application logic from any specific device. This allows devices being replaced during runtime. As depicted in

Figure 8, the middleware provides an application programming interface for the application developers that is called *Application Service Manager* whose main functions are: (i) to provide a service query interface for the application, (ii) to discover requested services, and (iii) to execute orchestrated services. The *Application Service Manager* uses an *Ontology Manager* to get semantic

information about services that it has found and uses the *Application Device Manager* to access services on these devices. The *Application Device Manager* retrieves services from devices on the network, by gathering information about devices that have been found on gateways or sinks by from *Discovery Manager. Discovery Manager* maintains this information in a *Device Application Catalogue*. Discovering a device can be initiated from an application by executing services provided by the *Application Service Manager*, which is then being propagated by the *Application Device Manager* to the nearest gateway. The nearest gateway forwards the request throughout the overlay network. In addition, an application may invoke the QoS Manager for performing a matching over a set of services that fulfill specific requirements to a certain extent.

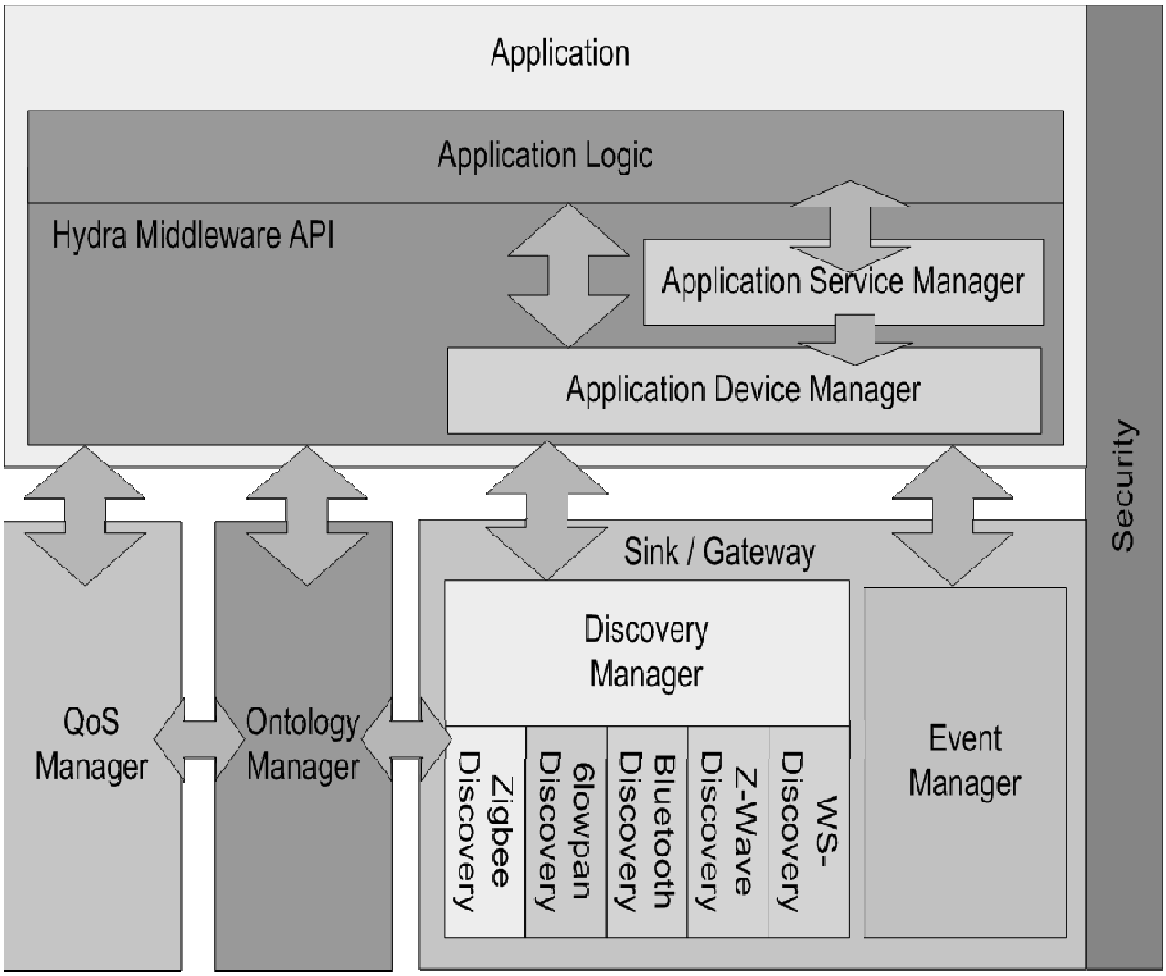

Figure 8: API for Application Development.

To assure the quality of information that goes to the application, LinkSmart provides a Quality of Service (QoS) Manager that allows application developers to select services based on specific QoS parameters. These QoS parameters are extensible and modeled in an ontology, for instance: network related parameters (e.g.: bandwidth, latency), resource related parameters (e.g.: power consumption, energy level, memory, processor), multimedia relevant parameters (e.g.: sound level, resolution, contrast, color). After the device developers have defined the QoS parameters in LinkSmart ontology, an application is able to query the QoS Manager for services that fulfill certain parameters to a certain extent.

5.3.3 Context Awareness in LinkSmart

The subject 'context awareness' is very broad, which means that it can be handled or defined in almost anyway. In order to simplify the use but not the potential for the LinkSmart developer/user the idea of context awareness and all its side meanings are joined in only a few components which can be used in a very powerful way. From the software integrators and developers the goal they want to achieve with a Context Awareness Framework is to make an application to be responsive to

any context changes. This fosters an intelligent behavior that the business application end users desired (e.g.: device behavior changes depend on the user's access right).

The basic approach to that idea is an object-oriented (albeit with a key-value pair model inside each object) configuration of contexts inside a rule engine, using inserted rules (as part of the specification of a context) to perform the reasoning and interpretation, which is extensible and easy to use for the person who defines these rules. As an example, the following pseudo-rule can be considered as simple and provides an "intelligent" outcome which can be interpreted by the context-aware application:

```
WHEN
      Movement Detected in Room A
      Room A is Dark
THEN
      Turn on Light in Room A
```

As previously mentioned, the rule shown above demonstrates the pseudo-code version of the rule. The actual rules are defined as objects that are interpreted by the Context Manager and processed into the rule language suitable for the rule engine used - Drools.

The example rule uses two pieces of data that can be sensed from the environment of *Room A*, by appropriate sensors. These are a light sensor and motion sensor(s). The data from these sensors is acquired using the Data Acquisition Component that reports the sensed data to the Context Manager, where it is modeled such that it can be reasoned over. The rule itself may specify the actual value of the sensed light level, or the actual contextualization of the sensed light levels as either "Light" or "Dark" could be handled in other rules. The firing of this rule, with the *when* conditions met, causes the light in *Room A* to turn on. This could be achieved by either the light service being called directly by the rule, or by a context-aware application being made aware of the situation, so that it may act on it, and turn on the light.

As discussed previously, the components which make use of this created rule are the key components inside the LinkSmart Context Awareness Framework: the Context Manager and the Data Acquisition Component. The application provides Context Specifications to Context Manager, contain the definition of the context being specified, as well as associated rules and subscriptions for data (if the context being specified represents a Device).

The main functionalities of the Context Awareness Framework are data retrieval, context reasoning and the execution of context-sensitive actions. This is the so called "intelligent" part of the Context Awareness Framework. It also provides functionalities beyond that such as an interface for accessing and querying (historic) context data for purposes arising while dealing with data inside applications or at a later stage. Therefore the Context Awareness Framework makes use of other LinkSmart components, like the Storage Manager or the Ontology Manager. Since context-awareness mostly deals with sensors which are rather limited in their capabilities, like e.g. a temperature sensors, it is sometimes useful to retrieve additional attributes of such a device, e.g. the position relatively to another object or absolute. The Ontology Manager provides methods to attach this kind of values to an ontology object which can be retrieved when needed.

Another benefit of the Ontology Manager is the use of assigning computational values to human understandable values, e.g. the GPS data which describes the position of a user's home and the position of the user's office. This is very helpful for the creation of the rule, in the case the user wants to define a rule which involves the position of a user or his smart phone. Such pseudo-rules could then be:

```
WHEN
            Phone near home
THEN
            Set phone profile to 'home'

WHEN
```

```
                Phone near office
THEN
                Set phone profile to 'work'
```

The Ontology Manager also provides the mechanism to define the term 'near home' or 'near office', which can be set by the developer and is sufficient for the whole lifetime of the LinkSmart application.

Figure 9 shows the overall component arrangement with respect to the Context Awareness Framework. As described above the Context Manager makes use of other already established LinkSmart components. It is clear that not all are mentioned in the text, due to the fact that the use of them is clear and described in other deliverables.

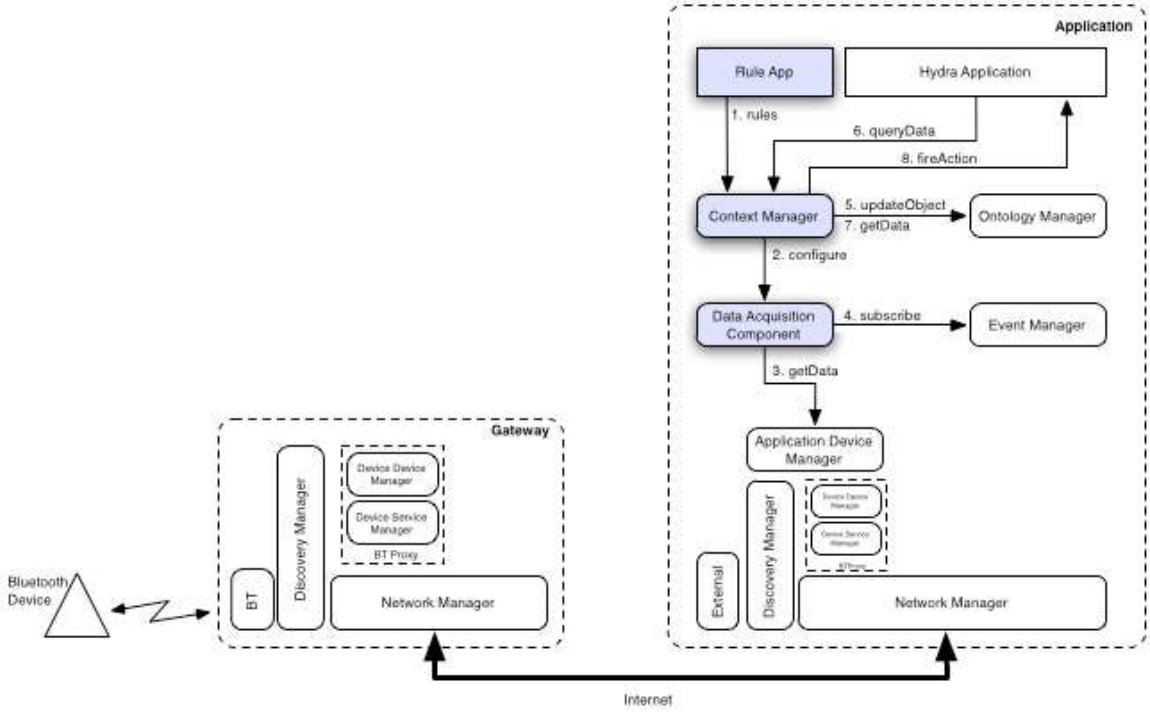

Figure 9: Context Awareness Integration to the LinkSmart Middleware

Figure 10 shows the data process of a simplified context awareness example, with the configuration part and data processing. There are only a couple of components involved in this procedure, for example the Storage Manager has been left out to simplify it.

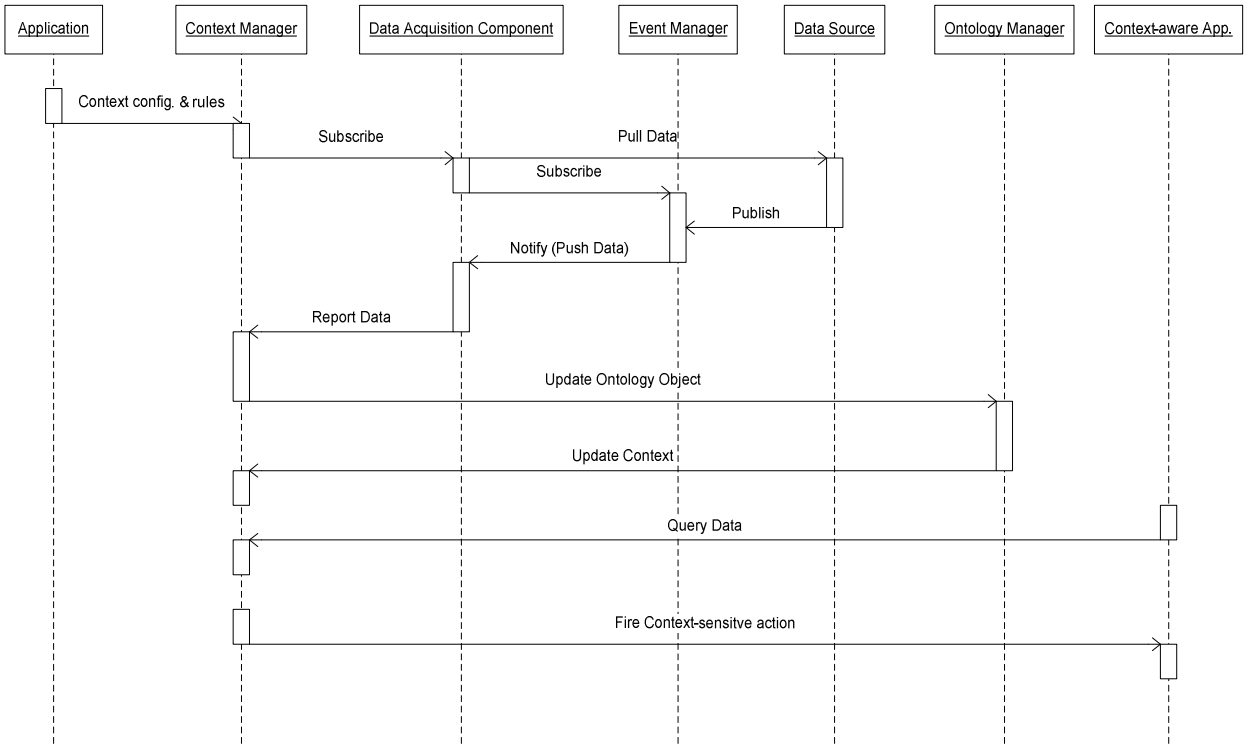

Figure 10: Sequence Diagram of the Context Awareness Framework

This is a more detailed description of the whole context processing procedure:

1. The Application send Context Specifications to the Context Manager
2. The Context Manager models contexts, and subscribes for required data from the Data Acquisition Component
3. The Data Acquisition Component initializes process for 'pulling' data from the data source at a set frequency
4. The Data Acquisition Component subscribes to the Event Manager for Events published to it by the data source
5. The Data Acquisition Component reports acquired data to the Context Manager where it is reasoned upon
6. The Context Manager may update the Ontology with the new values as interpreted by reasoning over new data. Not shown in figure is also the possibility to store context data inside using the Storage Manager
7. A application can query for context information
8. Output of rules in Context Manager as context-sensitive action(s) that are carried out with the involvement of other LinkSmart components, e.g. sensors, devices, applications.

As described above the Context Awareness Framework can be used in any situation where data by LinkSmart devices and sensors is needed to fulfill a certain task, e.g. the Quality of Service Manager and the self-management capabilities of the LinkSmart middleware. These two components base their functionality on changing situations which means on changing data. The quality of service depends on the attributes and capabilities of devices which might change due to a changing situation. An example illustrates these dependencies:

The user watches a film on TV and does not want to interrupt the viewing while he moves around in his house. After he has been in the living room he goes to the fridge in the kitchen, the media device in the kitchen has only limited resources (screen resolution or computational power), which means the quality of the TV broadcast has to be limited by the Quality of Service Manager. After he has been in the kitchen he decides to have a work out session in his gym and the TV program has to be adjusted to his step.

5.4 Summary and Conclusion

We reviewed the LinkSmart components that could be used for ebbits purposes. The basic infrastructure components such as network manager, security, and semantic services can be directly applied for ebbits, as they provide a generic abstraction to heterogeneous network protocols and elevate the low level communication into a secured service oriented architecture (SOA). SOA is adopted widely in business solutions and thus provides an interoperable communication within and inter enterprise. This fulfills the requirements of: "Transparent Communication" and partially "Security and Privacy".

Several components are not suitable to ebbits requirements and therefore they must be either extended, or completely rebuilt. For instance, storage manager only provides remote file system, which is not suitable for ebbits since ebbits must store a massive amount of distributed data, thus we will replace the storage manager using database management systems and a persistence abstraction layer such as Hibernate or ADO.NET Entity Framework.

The LinkSmart semantic infrastructure only describes devices semantically. However, the support for semantic description of events, which is of significant importance for ebbits, is missing in the LinkSmart middleware. This will require a modification on the event manager side as well as on the ontology manager side. The advantage of having events that are described semantically is the relationship between events and entities which can be inferred automatically by a software agent.

The context manager in ebbits only provides context reasoning based on rules that are coupled with physical sensors. This approach makes the system rather inflexible, as in real world there always exist exceptions which make the rule modeling complicated. Secondly, tightly coupling rules with sensors does not allow the context model to be eused. This is very unfortunate, since context modeling is a costly activity that must be done by experts.

6. Distributed and Centralized Intelligent Service Architecture

6.1 Functional view

The functional view of software architecture defines the architectural elements that deliver the system's functionality. The view documents the system's functional structure that demonstrates how the system will perform the functions required of it. According to Rozanski and Woods (2005), the functional structure model of the Functional View typically contains functional elements, interfaces, connectors and external entities:

- Functional Elements constitute well-defined parts of the runtime system that have particular responsibilities and expose well-defined interfaces that allow them to be connected to other elements. A functional element can be a software component, an application package, a data store, or even a complete system.
- Interfaces are specifications, defining how the functions of an element can be accessed by other elements. An interface is defined by the inputs, outputs, and semantics of each operation offered and the nature of the interaction needed to invoke the operation.
- External Entities can represent other systems, software programs, hardware devices, or any other entity the system communicates with.

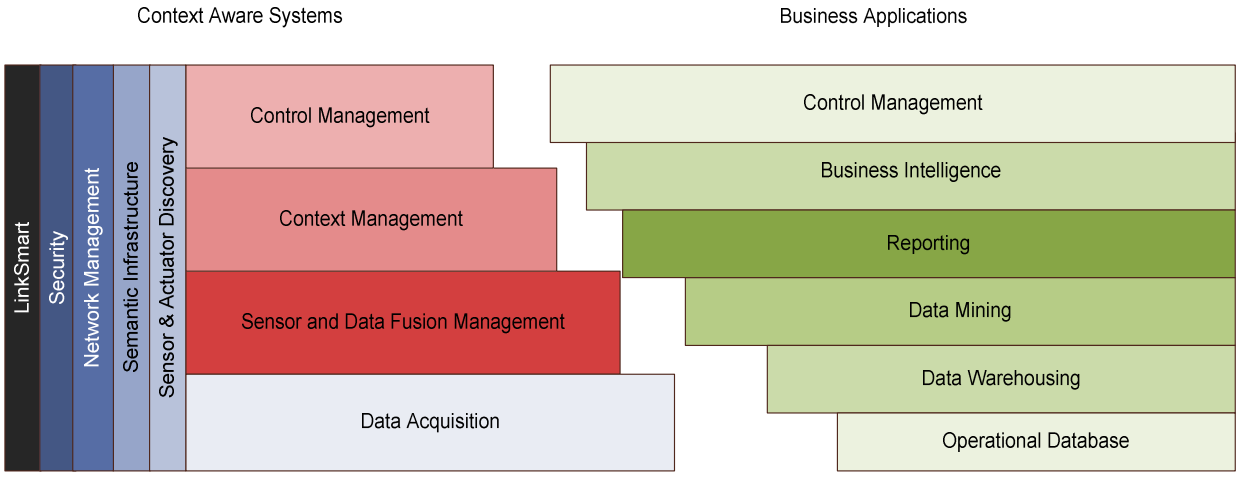

Figure 11: The Functions of Work Package 5 (red) and their relationship to LinkSmart middleware (blue) and enterprise applications (green).

Applied for the intelligent centralized and distributed services in ebbits, the functional view defines the three main functional capabilities that are to gather data intelligently from sensors and other input modalities, transform these data into context and adapt accordingly to the context in order to achieve business goals. As depicted in Figure 11, the components in work package 5 (depicted with red) are related to different level of business layers (depicted on the right side in green) and will be build on top of LinkSmart middleware (depicted in blue), thus the main components in the work package 5 must be designed as loosely coupled as possible not only to serve their vertical relationships but as well as their horizontal relationships to the existing business applications.

The components within WP5 will be developed based on LinkSmart Middleware (formerly called Hydra). LinkSmart provides security, device discovery, semantic infrastructures, and data acquisition component. Security in ebbits provides an encryption on the messages being exchanged. Moreover LinkSmart Security also allows services to be restricted based on certain policies. Network management of LinkSmart provides a p2p connection that works behind firewall seamlessly. Semantic Infrastructure allows application domain be modeled semantically so that relationships can

be inferred easily. Device discovery enable ebbits to discover devices based on their semantic description. This offers a great flexibility, since the users are not coupled with any specific devices, but rather to the semantic description of the devices (e.g.: capabilities, locations, quality, etc). Retrieving information from sensors is done by data acquisition component that provides push and pull methods these allows freedom for the application to choose how to sample data.

The enhancements that WP5 will introduce are firstly, the components that will be developed in the Task 5.2. Multi-Sensor Data Fusion. These components that are responsible to decide on data acquisition strategy manage and process raw sensor data into higher quality of information. These data can be processed on-line for any operational purposes (e.g.: to make decisions autonomously based on the context) as well as for off-line analytics purposes (e.g.: Business intelligence reporting). Data warehousing techniques are used to dump the operational data into another data repository in which data is transformed and analyzed without affecting the integrity of the original data. This relates closely to sensor data management in task 5.2 as data correlations and analytics are the fundamental features needed to support decision making process. Multi sensor data fusion provides means to fuse raw data into meaningful information for the users. Fusing data into information involves various techniques including filter, aggregation, correlation, pattern recognition, and estimation. These approaches are similar to data mining techniques. Reporting components in business application could take advantage by retrieving information from context management layer to report the context of each case in order to provide better causality information. (e.g.: why the energy consumption has increased?). A relation of business application and control management is that business rules defined in the business applications can be reused by control management to act to the current context.

6.1.1 Multi Sensor and Data Fusion Management

As discussed in D5.1.1. The first generic model was introduced by a data fusion working group of Joint Directors of Laboratories (JDL), a joint effort within the U.S. department of defense has been used as guidelines for developing multi sensor and data fusion solutions in many domains including manufacturing and goods production. However it was initially designed with vocabulary for defense systems. Moreover, it does not describe any architectural design that suffices as a reference for development process in ebbits (e.g.: it does not describe any discovery and data acquisition processes). Thus in this deliverable we will describe the viewpoints that are necessary to be a guideline for the project partners in developing the multi sensor and data fusion components. Figure 9 depicts a slight modification to JDL model. We changed the wording of each component's name from the original JDL model for clarity purposes as these terms are more generic compare to the original terms that were taken from defense domain.

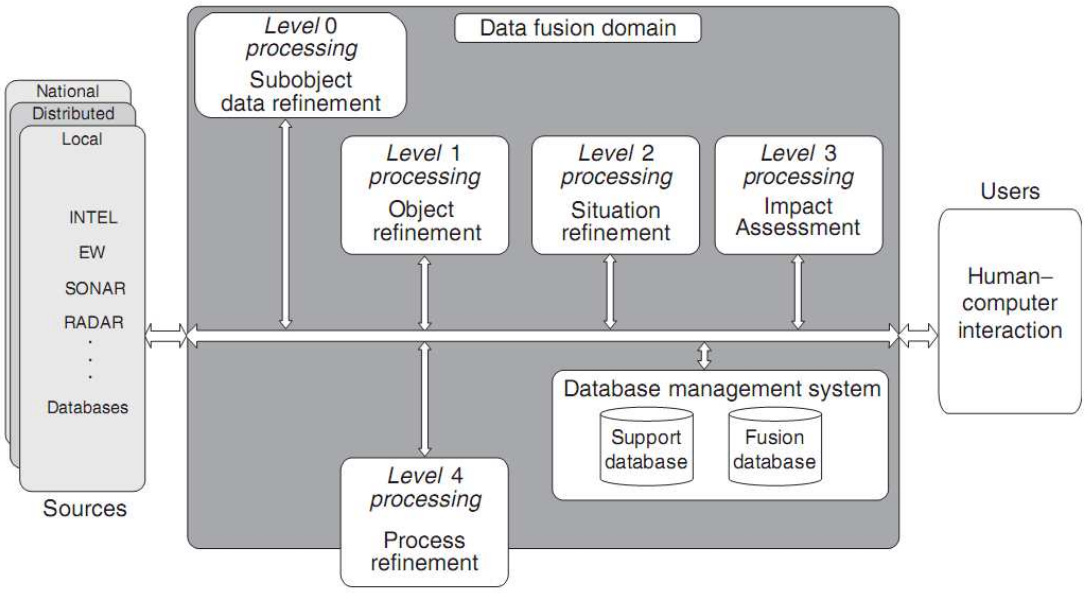

Figure 12: JDL Model for Sensor fusion(Liggins, Hall et al. 2009)

JDL model introduced a fusion process that includes 5 processing levels, a database, and a human computer interaction. The sources receive input from various sensors, a priori knowledge, and human. The database is responsible to maintain the data needed by processes. HCI allows human operators to provide query, input knowledge etc. We eliminate HCI components, as the sensor fusion processes in ebbits will primarily be used in conjunction with context and control management. Nonetheless, as the original JDL model, this model can also be used to support decision making process by human operators (e.g.: Business intelligence applications).

The processes in this model consist of 5 levels:

- **Subobject data refinement** aims at obtaining initial information about the each characteristic under observation without dealing with correlation to the object being observed. In this process, initial processing focuses on the signal acquired from individual sensor that includes cleaning signal from noises, smoothing, and extracting features. This stage can also consist of signal conversions and mapping. E.g.: analogue digital converter, quantization, image feature extractions. In this process, inference processes do not require assumptions about the presence or characteristics of entities possessing such observable features. This process is often performed by individual sensors, or with the product of individual sensors. However, when communication overhead and processing power is enough, this process can also include feature extraction and measurement using multi-sensor. Image fusion normally involves extracting features across multiple images, often from multiple sources.
- **Object refinement** was originally conceived as encompassing the most prominent and most highly-developed applications of data fusion: detection, identification, location, and tracking of individual physical objects (aircraft, ships, land vehicles, etc). Most techniques involve combining observations of a target of interest to estimate the states of interest. It focuses on combining sensor data from different sensors to estimate objects being observed based on its characteristics such as position, velocity. This stage aims at correlating entities and their individual characteristics e.g.: each step in manufacturing process consumes electricity and water.
- **Situation refinement** tries to describe relationship among the current entities and their environment which also includes clustering and relation analysis. This step inspects situation from holistic point of view to infer situations, states, occurring events, and interaction among entities and their environment. Methods for representing relationships and for inferring entity states on the basis of relationships include graphical methods (e.g., Bayesian and other belief networks). Situation assessment involves the following functions:
    - Inferring relationships and dependencies among entities
    - Recognizing/classifying situations based on the involved entities, attributes, and relationships.
    - Infer the effect of the objects interaction e.g.: new states and attributes of objects are introduced.
- **Impact refinement** definition was refined by JDL as "the estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants (e.g., assessing susceptibilities and vulnerabilities to estimated/predicted threat actions, given one's own planned actions)". This process tries to assess impacts of the situations as well as project the current situation to the future to draw inference about possible future impacts. Impact assessments includes analyze of opportunities, risks, vulnerabilities, and strengths. It involves combining multiple sources of information to estimate counterfactual outcome. It conducts a cost analysis given the current information.
- **Process refinement** deals with monitoring of the data fusion performance in order to improve the processes. This part works together with control and resource management in order to change the sensing processes (e.g.: by dispatching more or less sensors). This process combines information to estimate a system's measures of performance based on a desired set of system states and responses. This process may include sensor calibration and

alignment errors, track purity and fragmentation, etc based on the measures of errors and performance.

- **Database** in fusion domain is separated into database for support system and for fusion processes.

6.1.2 Context Management

The current LinkSmart context engine is not suitable for ebbits purposes since it is tightly coupled with drools[12] rule engine. The rule engine does not provide flexibility to learn and evolve over time autonomously and thus the rules must be maintained all the time. Ebbits requires a flexible system that needs a minimum intervention from human operators to reduce the maintenance costs as well as to keep the up time of the production machines as high as possible in order to meet the targeted throughput. Therefore in ebbits we will design a new context engine that is more flexible, and has high cohesion and low coupling to special technology, allowing the tailored components to be assembled on top of the framework to satisfy the ebbits requirements. The flexibility of a framework and its functionality and usability is always a trade off. The more flexible the framework is, the more difficult it would be to set up until a usable running system is achieved. Therefore this framework focuses primarily on the manufacturing and food production domain.

Context management has been developed variously over the last decade that includes simple approach such as attribute-value pairs to a more complex approach such as ontology modelling. Context management systems, according to a recent survey (Bettini, Brdiczka et al. 2010), are developed to gather, manage, evaluate and disseminate context information. Several challenges that context management system often face include (i) heterogeneous information sources (for instance, multi modal sensory, database, and user profiles), (ii) relationships and dependencies which are quite important to infer the appropriate responds to given the context information, and (iii) timeliness of the situations as context information might need to access past states and future states therefore, the historical information must be captured and take into account when inferring the present context.

Several works have discussed the disadvantage to model contextual using information from physical sensor directly, since this low level information is vulnerable to changes and uncertainty(Ye, Coyle et al. 2009). A higher level of context abstraction that uses situations, have been explored and proposed (Gellersen, Schmidt et al. 2002; Dobson and Ye 2006).

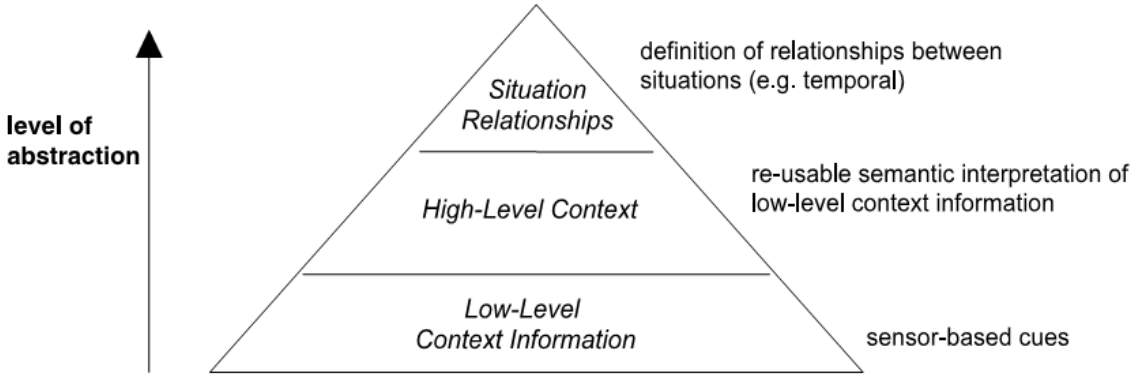

Figure 13: Different level of context abstraction (Bettini, Brdiczka et al. 2010)

Modeling context information in the higher abstraction decouples the context from the specific data acquisition processes and the sensor data itself. High level context information can be derived from the past, present and future situations of the environment and the entities in it. Situational information as explained in 6.1.1, are delivered by the sensor fusion component. This approach enhances the capability of context management in LinkSmart to be more flexible and extensible since high level context model can be reused and exchanged between systems.

[12] http://www.jboss.org/drools

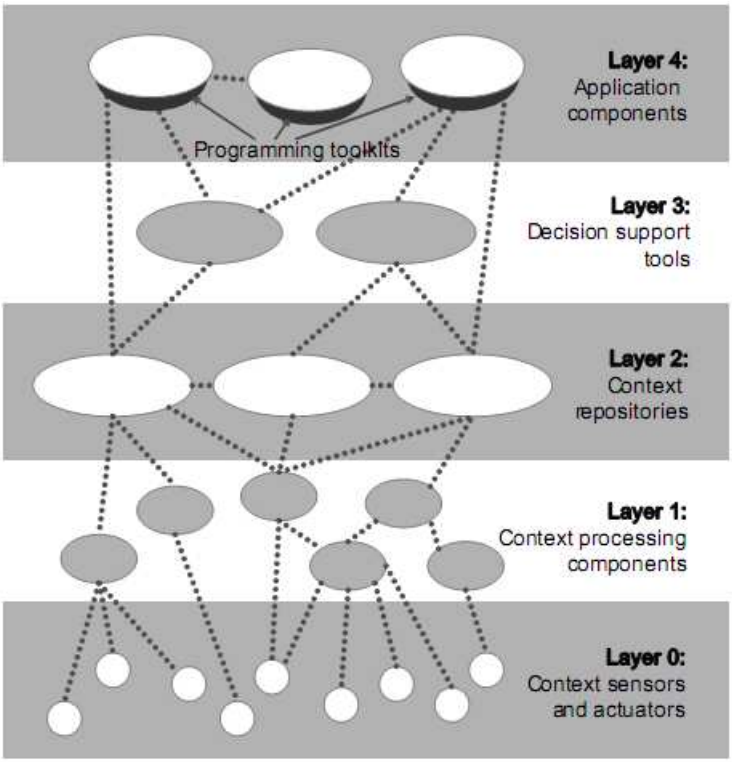

Figure 14: Context Aware Layers (Henricksen, Indulska et al. 2005)

Henricksen introduced five layers abstractions. Layer 0 resembles the hierarchical sensor network and fusion layer. Layer 1 translates the sensed data into beliefs of the actual situations (note that sensor readings always contain noise and are not 100% accurate.). Layer 2 manages the context model in a repository and serves context aware applications by providing means to query to the context models. Decision support tools support context consumer to decide on the corresponding actions and adaptations according to the context information. Programming toolkits provides support to the interactions of the application components with other components of the context-aware system.

Context management in ebbits is responsible to administer the context models of entities allowing applications to adapt to the situation changes. This offers a great flexibility and maintenance as required by the users, specifically the requirements of personalized information views, and self-* capability of the system.

Context aware framework in ebbits provides interface for the expert developers to define context models that represent context of the entities, saving them into repository, and providing a way to improve the models, and more importantly, it provides a channel to reuse and exchange the models among the applications in the network which also allows entities to move from an application to another.

We identify the functionality of context management system usually consists of several components (depicted in Figure 15):

Context Reasoner is responsible to correlate events and states of the current, past, and possible future situations with the corresponding entities and derive the context of the entities based on the information it obtains from the environment through sensors and previous knowledge.

Context modeler is responsible to interface with developers who define the context models of entities and situations. This can be implemented as a language such as xml, ontology, attribute-value model, as well as graphical user interface.

Feedback system is responsible to facilitate the improvement of the model by incorporating the feedback by expert users . The main goal of this component is firstly to improve the context models base on the feedback of the experts.

Knowledge Base is responsible to save context models into a repository. This layer is also useful for abstracting specific data persistence technology and give possibilities for developer to save the models in any persistence form e.g.: database, semantic web, semantic store.

Context Dissemination is responsible to disseminate the context information to the applications that are interested in the context of a situation in order to adapt its behavior.

Sensing Management is responsible to manage sensors and data acquisition processes, as well as conducting sensor fusion in order to detect events, entities, and current situations.

Control Management is responsible to manage actuation process in order to adapt the condition of the environment.

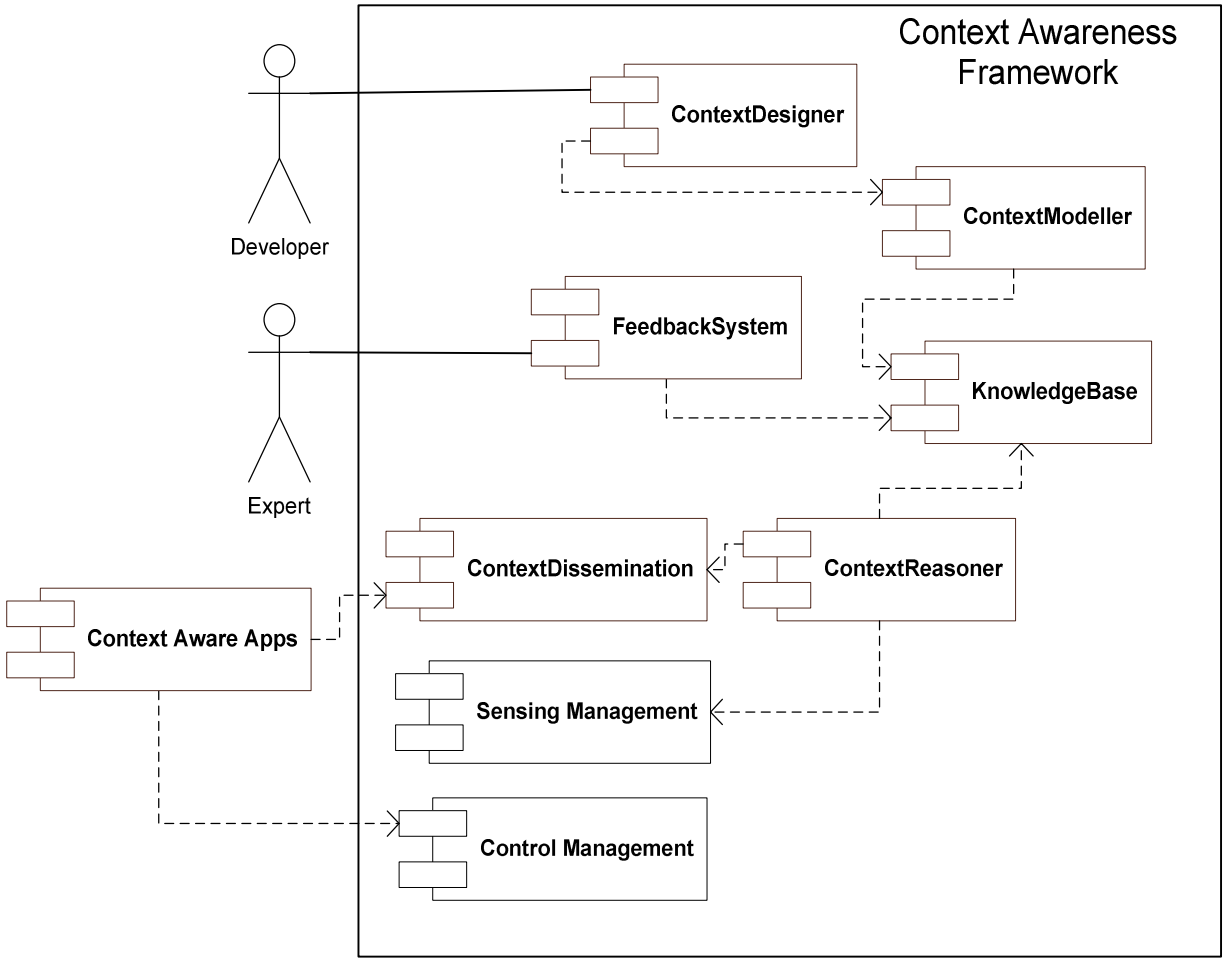

Figure 15: Context Awareness Framework in General

6.1.3 Control Management

Industrial control architecture

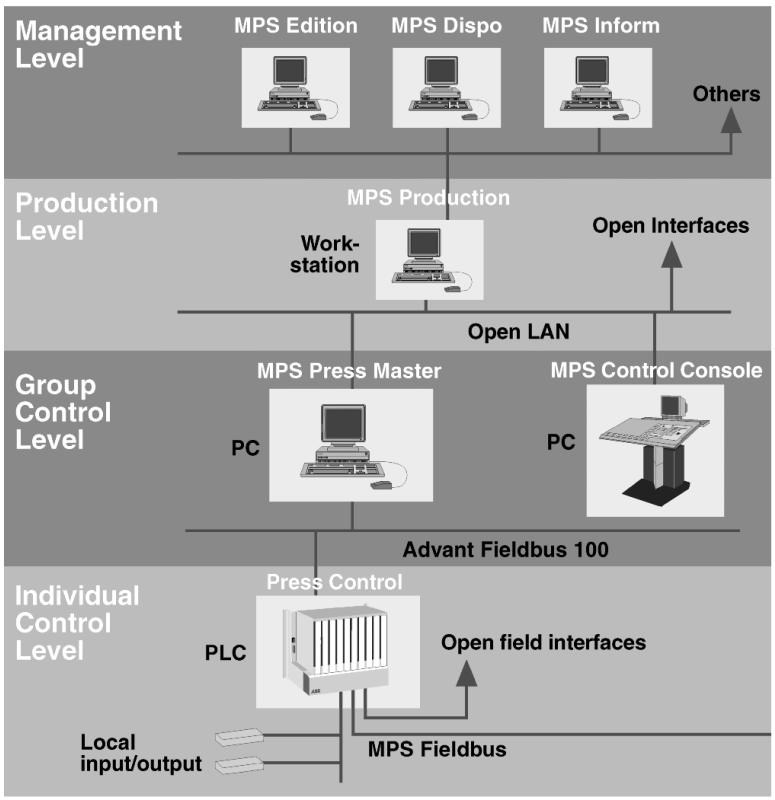

Figure 16: industrial layered architecture(Kirrmann 2011)

A general industrial control and process automation architecture abstraction can be seen in Figure 16 that is composed of four layers. The lowest layer deals with real time control mechanism such as moving robotic arms. This layer is dominated by embedded real time controllers that are known as programmable logic controllers (PLCs). PLCs communicate with each other through industrial bus systems such as Fieldbus[13], ControlNet[14], Profibus[15], and Modbus[16]. The PLCs are coordinated by Distributed Control System (DCS) or supervisory control and data acquisition (SCADA) which utilize PCs in the group control level. This level, is responsible mainly to coordinate without controlling the processes in real time. In the production level, manufacturing execution system (MES) and PCs are used to do planning and decisions of the production plans. The communication in this level is dominated by TCP/IP network. The decisions and production plans are then disseminated into the lower layers automatically through the network as well as manually through the operators. Management level deals mainly with office automation technologies such as enterprise resource planning system and analytic tools such as business intelligence. The integration between ERP and MES up to now is still a big problem, since there are technological and standardization gaps between two legacy systems. For the green field integration, ISA SP-95 defines standard interfaces for new MES and ERP systems(Scholten 2007).

Recent survey (Samad, McLaughlin et al. 2007) has revealed that the process controls that in the past have not been using discreet and event based control architecture, nowadays have become generally used as consequence of the emergence of the hybrid control that in this context encompasses regulatory, discrete, batch, logic, and sequence control. Many solutions have also shift from closed and proprietary into more open system as PC based supervision systems emerge.

[13] http://www.fieldbus.org/
[14] http://www.odva.org/Home/ODVATECHNOLOGIES/ControlNet/tabid/244/lng/en-US/language/en-US/Default.aspx
[15] http://www.profibus.com/
[16] http://www.modbus.org/

Fieldbus has brought a more complex distributed architecture. Sensor and actuators become more powerful and able to do processing locally for instance transmitters have compression and scaling algorithms built in, and actuators can include processors on which control calculations can be executed. Consequently, distributed processing introduces the potential for negative impact on overall system latency and jitter.

Model-predictive control that was only common on the supervisory level or group control has begun to emerge into individual control level. The implication of this trend will require user interfaces designed for end users working on this level such as technicians, unit operators, and supervisors.

Web and internet has also influenced the developments of the industrial control system. For instance internet offers costumer to subscribe to an online reporting tools (e.g.: Honeywell's Loop Scout service[17]) that collect process automation data, analyse it on the server, and report the result to the customer.

Samad et al. forecasted the term collaborative process automation systems (CPASs) for the next stage in the evolution of the distributed control system which requires a tight integration of information synchronization and real-time, contextual information exchanged directly between applications.

6.2 Deployment View

6.2.1 Service Oriented Architecture

The Service Oriented Architecture (SoA) represents an architectural style where the primary concept is the use of loosely coupled, implementation-neutral services supporting a business process as building blocks. Service consumers use the service by means of its published interface-based service description without dependence on implementation, location or technology. The process building of combining and sequencing services to provide more complex services is known as orchestration.

A SoA solution is built of a set of services orchestrated by clients or middleware to realize an end-to-end (business) process. The openness of the architectural style also allows for ad-hoc service consumers and flexible and dynamically re-configurable processes. The World Wide Web Consortium (W3C) defines SoA as "A set of components which can be invoked, and whose interface descriptions can be published and discovered". No universally agreed definition is available, but the term is generally considered to imply that application functionality is provided and consumed as sets of services which can be published, discovered and accessed and are loosely coupled as well as implementation and technology neutral.

SoA encourages loose coupling among the interacting software systems. A service is used only via the published service description and the service consumer does not address a specific implementation or deployed instance of the service. Changes to the implementation do not affect the service consumer and the service consumer can change the instance of the service that is used (changing location or implementation of the service, e.g. when two service providers offer the same service) without modifying the client application.

By abstracting the service from the implementation, the developer will not need to consider which technique was used to implement the service. Parallel implementations of the service may be available, and the actual version used is transparent to the consumer.

The use of standardized protocols for publishing, discovering and accessing services allows the service to be provided on any platform that can implement these protocols. In orchestrating a SoA solution, services that are (internally) implemented with different languages, architectural styles and on platforms from different vendors, can be used together transparently.

Any technology that can be used to implement loosely coupled, implementation independent services could be used to realize SOA. However, most discussions and actual implementations of SoA use Web Service technologies as the way of publishing, discovering and accessing a service. Web Service technologies include SOAP and XML for exchanging messages containing structured and typed information to access services, to publish and describe a service and UDDI for dynamically

[17] www.loop-scout.com

finding and invoking web services. On top of these now well-established protocols, a host of new protocols have been developed to support orchestration of services and describe the semantics of services e.g. OWL-S builds on OWL to define a core set of mark-up language constructs for describing the properties and capabilities of Web services, WS-Coordination provides a method of defining and supporting workflows and business processes. WS-Coordination is an extensible framework for providing protocols that coordinate the actions of individual web services in distributed applications to provide a business process defined in BPEL. WSRF (Web Services Resource Framework) defines an open framework for modelling and accessing stateful resources using Web services.

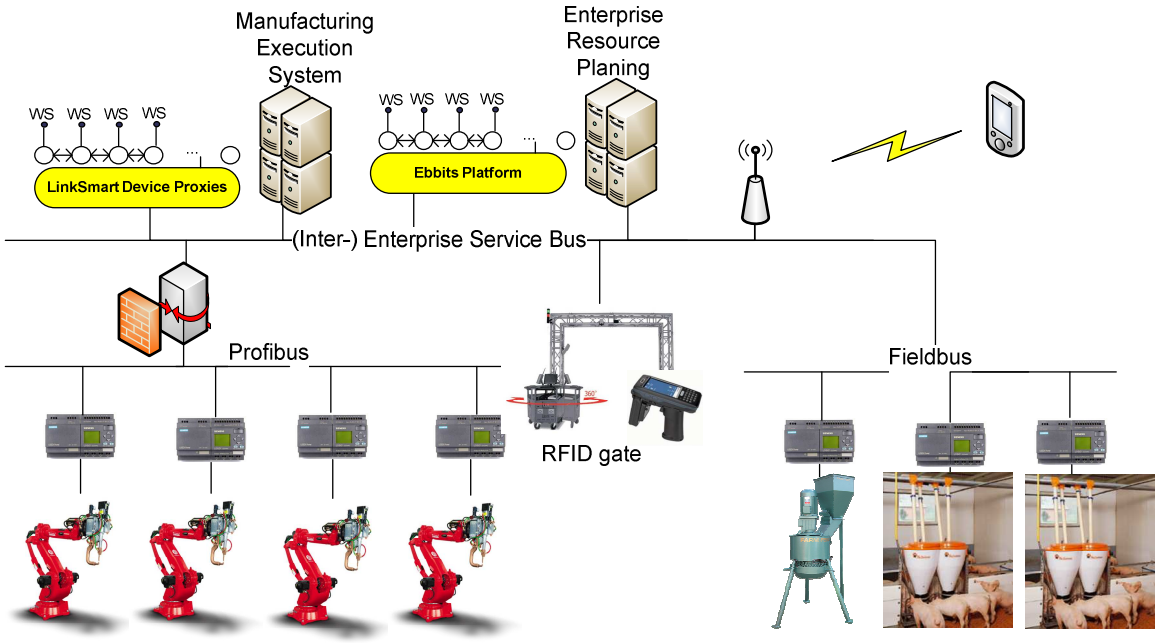

Figure 17: Service Oriented Architecture ebbits

Typical distributed industrial system architecture contains sensors, controller (PLCs), and mechanic components as actuators. Local architecture suggests that the controlled equipments are deployed within close proximity and the scope of each controller is limited to a small sub-system. PLCs are typically capable of accepting inputs from a supervisory controller (e.g.: DCS / SCADA) to initiate or terminate locally-controlled automatic sequences, or to adjust control set points, but the control action itself is determined in the local PLCs. The required operator interfaces and displays are also local. This provides a significant advantage for an operator troubleshooting a problem with the system, but requires the operator to move around the facility to monitor systems or respond to system contingencies. In a distributed control system, controllers are provided locally to systems or groups of equipment, but networked to one or more operator stations in a central location through a digital communication circuit. Control action for each system or subsystem takes place in the local controller, but the central operator station has a complete visibility of the status of all systems and the input and output data in each controller, as well as the ability to intervene in the control logic of the local controllers if necessary. The communications among PLCs involve a real-time and wired communication through industrial Bus as explained in 4.1.5.

In ebbits, we propose that communication to devices is facilitated by software proxies that operate in SOA environment. The software proxies offer web service interface that are accessible through the TCP/IP network as well as from internet. The proxy is responsible to provide communication from devices to MES, and ERP system. This means that the proxies must handle the communication between real-time operation of embedded systems and non-real time operation that happens on the SOA environment. In order to keep the embedded components always synchronized regardless of the jitter and delays of communication from non real-time systems, all control actions through the proxies must be offered as high level services that represent synchronized actions of embedded

components. For instance, a function that a service offer is called setProductionSpeed(int x) where the setProductionSpeed will execute smaller commands to control individual embedded devices. Since the direct actuation of the mechanical components must be done in the real-time environment, these small commands must be buffered in the real time system until all corresponding commands are accepted, before any execution process is allowed.

There have been many standards to integrate industrial controller with PCs such as OLE for Process Control (OPC), which stands for Object Linking and Embedding (OLE) for Process Control, is the original name for a standards specification developed in 1996. The recent specification of OPC enables web service as the interface which is named OPC Unified Architecture (OPC UA). We propose to use this standard for connecting the device proxies into the LinkSmart network. However, OPC standard is only widely used for PLCs based controllers which is not always the case for the farms equipments. Thus, for the farms equipment, the proxies must be built to handle heterogeneous proprietary protocols.

6.3 Summary and Conclusion

According to the initial requirements, intelligent centralized and distributed services in ebbits should define the three main functional capabilities including: (i) to gather data intelligently from sensors and other input modalities, (ii) to transform these data into context and (iii) adapt accordingly to the context in order to achieve business goals. We propose to follow the JDL model for processing data acquired from multiple sensors. The JDL model recommends a fusion process that includes 5 processing levels that process raw signals from multiple sources resulting in a situation assessment. The accumulated situation over time can be used to infer the context based on the context models defined by experts. We learned from the LinkSmart Context Framework that modeling context information in the higher abstraction decouples the context from the specific data acquisition processes and the sensor data itself. High level context information can be derived from the past, present and future situations of the environment and the entities in it. Situational information (explained in Section 6.1.1) is delivered by the sensor fusion component. This approach enhances the capability of context management in LinkSmart to be more flexible and extensible since the high level context model can be reused and exchanged between systems.

Industrial control and process automation architecture abstraction is normally composed of four layers. The lowest layer deals with embedded real time controllers that are known as programmable logic controllers (PLCs). In the second layer, the PLCs are coordinated by a Distributed Control System (DCS) or a supervisory control and data acquisition (SCADA) which utilize PCs. In the production layer, manufacturing execution system (MES) and PCs are used to do planning and decisions of production plans. And the highest layer is the office automation where normally enterprise resource planning system and service oriented architecture are used. Service oriented architecture provides a standardized communication for enterprise applications. These layers are not fully electronically integrated. Some of the data is still manually transferred using paper documents. When integrating these layers electronically, the low level communication that involves a real time embedded environment must be handled with consideration of communication jitter and delays that SOA caused. Therefore we propose a proxy based solution and a gateway in the real-time environment that is able to keep the synchronization of mechanical components. Integrating the manufacturing equipment to the LinkSmart proxies can be done through OPC technology, however for farm equipments we still have to deal with various proprietary protocols.

7. Bibliography

Krakowiak, S. *ObjectWeb*. 2003 8 April 2010]; Available from: http://middleware.objectweb.org/.

Day, J.D. and H. Zimmermann, *The OSI reference model.* Proceedings of the IEEE, 1983. **71**(12): p. 1334-1340.

Tanenbaum, A.S.V.S., M, *Distributed Systems. Principles and Paradigms*. 2nd ed. 2008: Prentice Hall International.

WS-I, Basic Profile Version 1.1. 2006; Available from: http://www.ws-i.org/Profiles/BasicProfile-1.1.html

Lardies, F.M., et al., Deploying Pervasive Web Services over a P2P Overlay, in Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises. 2009, IEEE Computer Society. p. 240-245.

Hoffmann, M., et al. Towards Semantic Resolution of Security in Ambient Environments. in International Conference on Ambient Intelligence Developments. 2007. Sophia-Antipolis, France: Springer Paris.

Hansen, K.M., et al., Flexible Generation of Pervasive Web Services Using OSGi Declarative Services and OWL Ontologies, in Proceedings of the 2008 15th Asia-Pacific Software Engineering Conference. 2008, IEEE Computer Society. p. 135-142.

Bettini, C., O. Brdiczka, et al. (2010). "A survey of context modelling and reasoning techniques." Pervasive and Mobile Computing **6**(2): 161-180.

Dobson, S. and J. Ye (2006). Using fibrations for situation identification, Citeseer.

ebbits-consortium (2010a). D5.1 Concept and Technologies in Intelligent Service Structures, Ebbits Consortium.

ebbits-consortium (2010b). D2.4 Initial Requirements Report, Ebbits Consortium.

ebbits-consortium (2010c). D2.1 Scenarios for Usage of the ebbits Platform, Ebbits Consortium.

Eisenhauer, M., P. Rosengren, et al. (2009). A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems. Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on.

Gellersen, H., A. Schmidt, et al. (2002). "Multi-sensor context-awareness in mobile devices and smart artifacts." Mobile Networks and Applications **7**(5): 341-351.

Henricksen, K., J. Indulska, et al. (2005). "Middleware for distributed context-aware systems." On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: 846-863.

Kirrmann, P. D. H. (2011). Control System Architecture ABB Research Center, Baden, Switzerland.

Liggins, M. E., D. L. Hall, et al. (2009). Handbook of Multisensor Data fusion, Theory and Practice. Boca Raton, FL, CRC Press.

Rozanski, N. and E. Woods (2005). Software systems architecture: working with stakeholders using viewpoints and perspectives, Addison-Wesley Professional.

Samad, T., P. McLaughlin, et al. (2007). "System architecture for process automation: Review and trends." Journal of Process Control **17**(3): 191-201.

Scholten, B. (2007). The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing, Isa.

Ye, J., L. Coyle, et al. (2009). Using situation lattices in sensor analysis, IEEE.

8. Table of Figures

Appendix A. Complete Requirements of WP-5

ID	Summary	Priority	Rationale	Fit Criteria	Source
27	Product-related information should be represented in a machine-readable format	3	Automatic processing requires that machines can understand and process information	Machines can process information of a product automatically.	TNM scenario workshop in Copenhagen
35	Hazardous Environmental Monitoring of Manufacturing Plant	1	Currently the environment of a plant provide is not monitored properly. However, this is quite important to guarantee the safety of an operator.	The safety of the operator is improved by 20% on the basis of environmental input information.	During ebbits manufacturing workshop (19th Oct, 20010) COMAU employee (Roberto) raised this issue.
36	Controlling of machines/stations in manufacturing plant remotely	3	To optimize production process.	Relevant stations that operate automatically can be started/stopped via remote calls.	During ebbits manufacturing workshop (19th Oct, 20010) COMAU employee (Fulvio) raised this issue.
39	Retrieve manufacturing data history of any relevant event during production	1	If production defects are recognized, it is helpful to look at the production process history in order to find out what caused the defects.	Any manufacturing relevant (pressure, energy consumption, temperature, humidity, time etc) data is retrievable.	During ebbits manufacturing workshop (19th Oct, 20010) COMAU employee raised this issue.
43	Aggregating collected sensor data at a central point	1	The aggregation of collected data is important for analyzing the data.	A framework is provided that aggregates collected sensor data at a central point of an application.	TNM said that they currently can obtain different sensor data, though the aggregation is missing.
44	Farmers are able to retrieve optimized models from research	4	Farmers are willing to share data if they could get something in return such as models to optimize feeding process.	Farmers can get optimized models electronically.	TNM workshop
45	System can feed the farms data to research	4	Most of the farming models are developed by research organizations, universities etc.	Researchers are able to get their hands on life data on farms.	TNM workshop (Thomas)
47	Resilience and adaptable to environment condition changes	2	environmental changes such as lighting, temperature affect the results of manufacturing process. so far machines are tuned manually by technicians. adapting to environmental condition can lead to reducing energy consumption e.g.:reduce	machines can adapt its parameters adapting to environmental changes.	During ebbits manufacturing scenario workshop in Oct 2010 this is issue had been raised by a COMAU

			heater temperature when it's warm outside.		employee.
49	Access to energy-related information from production machines needs to be provided.	2	Energy-related information is measured by some of the operational machines (e.g. in the production plant), but it is not distributed into a network.	If any machine provides access to energy-related information, ebbits distributes this information to all interested parties.	COMAU scenario workshop (10/19/2010).
750	Filtering to Obtain relevant Information	3	Too much information overwhelm farmers while making decisions.	Farmers are able to view the relevant information out of the whole.	TNM said that they need to provide only relevant information to farmers. Farmers have different views on relevant information
64	Loggiing of Quality related informaation of each Manufacturing Part	1	Quality is very important inside an assembly line as it is the essential parameter used for force tests or lack tests.Furthermore, if failures are detected lately when a car is already in the market, but shows some lack, the production history can be traced to find the devil in the detail.	Quality related information is logged inside a proper carrier medium.	During ebbits manufacturing workshop (19th Oct, 20010) COMAU employee (Fulvio) raised this issue.
66	correlate problems found with production batches	2	when the source of problem have been isolated, producers must know which products/batches are affected.	production batches affected by problems can be identified.	TNM Workshop
67	automatic analysis of cross enterprises product life cycle data	3	searching production problem from end costumer complaints need to track back data from several enterprises and logistic.	analyzing data cross enterprises can be done online and automatically.	TNM Workshop Copenhagen
72	officials have a back door access to highly important information	4	officials want to avoid enterprises commit information / documents forgery	offcials have an access to certain information	TNM Workshop in copenhagen
75	system should aware of what which livestocks are in the building	3	pigs in different phases have different requirements of climate, insulation, feed, vitamins, etc	system can adjust itself according to what's inside the building.	TNM Workshop in copenhagen
78	system should provide location tracking of the stocks/livestocks	1	users sometimes lost track where the goods /animals are.	users can identified where the stocks / livestocks are	TNM Workshop Copenhagen, Comau Workshop Turino
79	location tracking should be implemented as independent app	2	decoupling from existing system	tracking system is implemented independently	TNM Workshop Copenhagen

81	System should show Energy Cost for different granularity of production processes	1	energy cost at different levels is needed to do benchmarking of operational processes.	each automated process, machine is able to show energy cost	TNM Workshop in Copenhagen, Comau Workshop in Turino
82	Protection to sensitive information	1	some sensitive information endanger company existence.	system provides access restrictions to sensitive information	TNM Workshop, COMAU Workshop
91	filter/fusion information for each operational process	3	each process needs different resolution of information	processes only get information needed	Comau Workshop in Turino
92	early maintenance notification when needed	4	early maintenance prevent permanent damage to the robots, ensure the reliability of robots	users/technicians are notified if robots need maintenance	COMAU Workshop Turino
93	bring data from fieldbus network to ethernet network	3	analytics is done by ERP program on a computer that work on TCP/IP.	analytics software can analyse data from manufacturing robots	Comau Workshop Turino
103	automatic calibration	1	calibration is still done manually it is error prone, and takes time.	75% of existing manual calibration is done automatically.	Comao Workshop in Turino
109	recognition of energy wasting behaviors	4	help decision makers to optimize energy usage	decision makers are alerted when energy wasting takes place	Comau Workshop
130	Item identification system should provide open interfaces to other systems	3	Identification of pigs is done with RFID tags at their ears and with antennas in corridors that recognize pigs passing by. The system should not be connected to a specific system, but rather provide open interfaces that can be exploited by any system.	Any system can easily access the item identification system.	TNM user workshop in Copenhagen
131	Support fuzzy or probability concepts for reasoning	4	there is no reasoning algorithm that is able to solve any kind of cases	Fuzzy concepts should be supported through e.g. probabilistic models.	Hydra open requirements
134	Ability to self-adaptation	2	A knowledge model enables the middleware to contain a representation of itself and manipulate its state during its execution. This feature should serve as the basis for self-adaptation of the middleware (e.g. reconfiguration of resource usage, triggering the component-based services).	Middleware is able to adapt its configuratiton in 60% of identified cases requiring reconfiguration.	Hydra open requirements
135	Protection of System Integrity	2	In order to prevent an inexperienced user to cause malfunctions by changing system configurations, the middleware should monitor, analyse and, if necessary,	Middleware provides mechanisms to monitor system integrity and to react in the case of failures.	Hydra open requirements

			prevent or give notifications about faulty changes.		
138	Distributed Intelligence should not lead to resource-heavy systems	1	We have a need for "intelligence" (Semantics, reflection etc.). We have a need for supporting embedded systems. This should not conflict	Minimum hardware requirements (which must be supported by all target hardware) are defined and all hardware that meets the specifications is guaranteed to work with hydra.	Hydra open requirements
139	Support runtime reconfiguration	3	To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable at runtime.	Services and devices can be connected during runtime.	Hydra open requirements
140	Transparentness of device performance	3	The middleware should contain services that allow monitoring on what devices are doing. This includes monitoring response time, device load (e.g., CPU), and message interchanges per second.	Devices provide monitoring services.	Hydra open requirements
141	Report errors in devices	2	Devices should be able to report errors.	Devices provide services for reporting errors.	Hydra open requirements
154	Aggregate data from various data bases and sources	3	Information will be stored in several places, but needs to be combined in some place and assigned to the actual product or entity.	A data aggregation component is available.	TNM user workshop in Copenhagen
155	Synchronisation of Acquired Data is necessary	3	Data synchronization might be necessary, because data will be acquired automatically, manually, semi-manually with different timestamps.	A data synchonization component performes a timestamp-based synchronisation of a data set.	TNM user workshop in Copenhagen
157	Different Views on the Data is necessary	3	We need services that provide different views on the data cloud by combining data from different sources.	Data can be filtered and sorted based on an arbitrary set of parameters.	TNM user workshop in Copenhagen
159	End-users need to be able to managment their distributed data	3	Farmers want to manage their distributed data, because today they have no full control of data.	End-users can easily manage data from distributed sources.	TNM user workshop in Copenhagen