Enabling the business-based
Internet of Things and Services

(FP7 257852)

D2.7.1 Lessons Learned and results of usability evaluation 1

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 version: 1.1

Submission date: 3 October 2011

Document control page

Document file: D2.7.1 Lessons Learned and results of usability evaluation 1.doc
Document version: 1.1
Document owner: In-JeT ApS

Work package: WP2 – Requirements engineering and validation
Task: T2.3 – Evolutionary requirements refinement
Deliverable type: R

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	H. Udsen (IN-JET)	2011-07-20	TOC
0.2	H. Udsen (IN-JET)	2011-09-07	Lessons Learned (not validated) entered for several WPs
0.3	A. Al-Akkad (FIT)	2011-09-15	Additions to WP5 Lessons Learned and Usability evaluation results
0.4	A. Al-Akkad (FIT)	2011-09-16	Analysis of WP5 Lessons Learned included
0.5	A. Al-Akkad (FIT)	2011-09-16	Corrections to section 4.4.1
0.6	H. Udsen (IN-JET)	2011-09-18	Sections 2, 3, editing of other sections
0.62	Y. Martin (SAP)	2011-09-19	WP 6 section
0.7	F. Pramudianto, A. Al-Akkad (FIT)	2011-09-15	M12 demo prototypes included in Usability evaluation results
0.8	M. Caceres (ISMB)	2011-09-20	Analysis of Lessons Learned for WP8
0.9	M. Knechtel (SAP)	2011-09-21	Lessons Learned + Analysis for WP4
0.91	M. Ahlsén (CNET)	2011-09-21	WP7 update
0.92	P. Kool, M. Ahlsén (CNET)	2011-09-22	WP9 update
0.93	P. Brizzi, M. Caceres (ISMB)	2011-09-22	WP8 update
1.0	H. Udsen (IN-JET)	2011-09-26	Executive summary and editing
1.1	H. Udsen (IN-JET)	2011-10-03	Reviewer comments addressed
1.1			Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
P. Rosengren (CNET)	2011-10-02	Approved with comments
C. Pastrone (ISMB)	2011-09-29	Approved with comments

Index:

1. Executive Summary

This deliverable reports the first results of *Subtask 2.3.1 Lessons Learned collection and analysis* and contains a complete list of Lessons Learned during the first iteration cycle of the ebbits project, organised per work package.

The subsequent analysis of Lessons Learned has identified a number of relevant improvement opportunities for the specification of requirements for the next cycles of the iterative development process, feeding into the closely related deliverable *D2.8.1 Change request and re-engineering report 1*.

1.1 Research and development methodology

Section 3 provides an overview of the research and development methodology applied in ebbits, describing how the requirement engineering process follows the guidelines of ISO 9241-210 Human-centred design for interactive systems.

The future use of ebbits applications will be assessed in two different domains, Automotive Manufacturing and Food Traceability. A set of domain-specific Vision Scenarios are used to derive technical and business oriented usage scenarios, providing the foundation for an initial set of requirement specifications, which again contribute to the definition of the first architectural specifications. This drives the research and development work in application implementation and system integration, which will be conducted in four one-year iterative cycles, each resulting in a specific prototype application.

After the successful completion of a prototype cycle, each RTD work package will analyse and report their development results, RTD experiences, Lessons Learned in the development and integration work, including the latest developments in technology, regulatory affairs and markets.

The Lessons Learned concept is an important part of knowledge management in this iterative process, involving the systematic and continuous collection, indexing and dissemination of these Lessons, which will be undertaken in WP2. The six steps of this process are collection, verification, storage, dissemination, reuse and identification of improvement opportunity.

1.2 Lessons Learned

Section 4 contains all Lessons Learned in cycle 1 and the subsequent analysis, the outcome of which is the identification of a number of improvement opportunities. The ensuing changes in requirements are reported in the deliverable *D2.8.1 Change request and reengineering report 1*.

The Lessons Learned are reported per work package as follows:

WP2 has collected one Lesson Learned; no requirements were added, updated or deleted.

WP3 has collected three Lessons Learned; no requirements were added, updated or deleted.

WP4 has reported five Lessons Learned, resulting in the addition of five new requirements.

WP5 has reported eleven Lessons Learned, resulting in the addition of five new requirements.

WP6 has reported two Lessons Learned, with the subsequent updating of one requirement.

WP7 has collected seven Lessons Learned, resulting in the addition of six new requirements.

WP8 has reported seventeen Lessons Learned, resulting in the addition of twenty new requirements, while two requirements have been updated.

WP9 has collected three Lessons Learned; no requirements were added, updated or deleted.

No Lessons Learned have been collected so far in WP1, WP10, WP11 and WP12.

In total, the first iteration cycle yielded 49 Lessons Learned. This has resulted in the creation of 45 new requirements and modification of four requirements. No requirements have been deleted.

1.3 Usability evaluation

Four prototypes have been demonstrated, two for each application domain, after six and twelve months, respectively. The M6 prototypes were additional to the prototypes prescribed at the end of each annual cycle in the Description of Work, for the purpose of familiarising developer end users from different organisations with the test-bed environment and each other.

1.3.1 Month six demos

The M6 Automotive Manufacturing demo was driven by a common industrial task of spot welding through automatic robots in automated manufacturing plants. Monitoring power and water consumption is a first step for enabling process optimisation, accounting and analysis of key performance indexes (e.g. OEEE).

The M6 Food Traceability demo seeks to enhance the communication workflow of medical data between veterinarians and farmers. For the demo, the focus has been on tracing the medication of pigs to prevent medicated animals from being sent to slaughter by mistake and ensuring that it can be traced back in case of contamination.

For the two M6 demos the general conclusion is that the LinkSmart middleware developed in the Hydra project provided adequate support for application developers, exploiting two LinkSmart components: a Network Manager and an Event Manager.

Various advantages and disadvantages of building on the LinkSmart middleware have been reported, and for future work a rather generic and reusable approach will offer more flexibility.

1.3.2 Month 12 demos

For the M12 Automotive Manufacturing demo the initial architecture of the ebbits platform was evaluated by asking several developers to work on a typical application; some developers were asked to develop applications, others devices. The application monitors the physical state of a transformer and the energy consumption of a welding process.

The M12 Food Traceability demo was developed to monitor the state of health of pigs in a farm, such as their drinking and eating pattern and movement behaviour, simulating the feeding and drinking station through an event generator.

The M12 Demo Prototypes are explained in detail in *D5.4.1 Multi-sensory fusion and context awareness prototype*.

For the two M12 demos it was concluded that the event-driven approach used is easy to understand and that the context-aware paradigm supports the developers in clustering sensor information based on entities. However, application developers also mentioned that the number of events could grow significantly in complex systems. Therefore, a complex event-processing engine should be integrated into the ebbits platform. Various other problems and findings were reported, e.g. the unreliability of the LinkSmart Event Manager for handling a burst of events.

2. Introduction

This deliverable reports outcomes of Task *T2.3 Evolutionary requirements refinement* and more specifically the first results of *Subtask 2.3.1 Lessons Learned collection and analysis*.

2.1 Purpose, context and scope of this deliverable

This document contains a complete list of Lessons Learned during the first iteration cycle of the ebbits project, organised per work package. These Lessons have been extracted from the joint repository specifically established in the WP2 area of the Wiki part of GForge, the selected tracking and management tool for the project.

The Lessons Learned have subsequently been analysed to elicit relevant improvement opportunities for the specification of requirements for the next cycles of the iterative development process.

This analysis feeds into the closely related deliverable *D2.8.1 Change request and re-engineering report 1*, as does the results of the usability evaluation reported in Section 5 .

3. Research and Development Methodology

The ebbits project will develop architecture, technologies and processes, allowing businesses to integrate the Internet of Things into mainstream enterprise systems and support interoperable real-world, online end-to-end business applications. It will bridge 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. The platform will be based on a Service-oriented Architecture (SoA), transforming every device into a service.

The work in ebbits involves many complex issues, which will be tackled in twelve interrelated work packages.

In the subsections below, a brief description is given of the software engineering process, the iterative approach and the Lessons Learned methodology adopted for ebbits, outlining how these techniques all work together to support the requirements re-engineering process.

3.1 Software engineering process

The requirement engineering process in ebbits follows the principles of ISO 9241-210[1]. This standard provides guidance on human-centred design activities throughout the lifecycle of computer-based interactive systems. The engineering process is enhanced by the expertise of the multi-disciplinary team, which is essential for a human-centred design process. The solutions are implemented in iterations, balancing technology and user functions. Users (who may be end users or developer users) will be actively involved in the process.

3.2 Overview of the iterative approach

The starting point of the iterative design process in ebbits is a set of domain-specific Vision Scenarios delivering end user visions of the future use of ebbits applications in two different domains: Automotive Manufacturing and Food Traceability. Figure 1 presents an overview of the iterative approach.

[1] ISO 9241-210:2010-03 (E). Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems

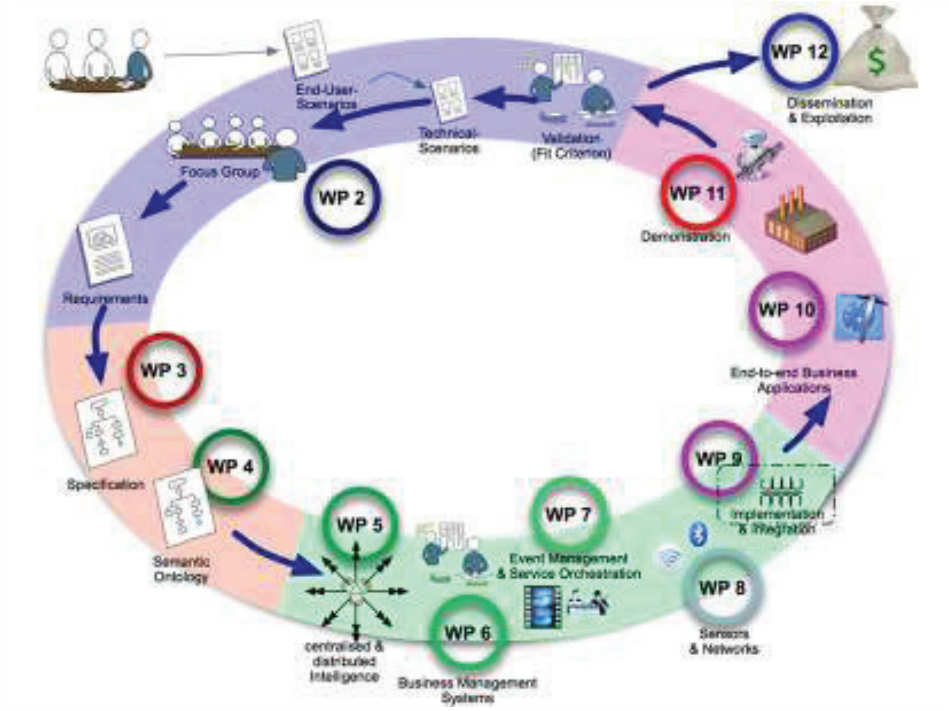

Figure 1 – Overview of the iterative process

These Vision Scenarios are used to derive technical and business oriented usage scenarios, providing the foundation for an initial set of requirement specifications reported in deliverable *D2.4 Initial requirements report*.

This initial set of requirements contributes to the definition of the first architectural specifications, which drive the research and development work in application implementation and system integration. Software prototypes will be demonstrated and validated in domain-specific settings, aiming to demonstrate the outcome of each cycle to developer users, end users, project partners, reviewers, etc.

The planned verification, validation and usability testing activities are described in deliverable *D2.6 Validation framework*. All results from validation and experiences gathered in the process will lead to refined technical and business-oriented scenarios, revised requirements specifications, updated architecture and new prototype specifications.

3.3 Re-engineering of requirements

The ebbits project is planned to evolve in four iterative development cycles, each resulting in a prototype application as follows:

- End of year 1: First prototype of the ebbits platform serving as proof-of-concept
- End of year 2: Prototype II based on the production optimisation scenario
- End of year 3: Prototype III incorporating the food traceability scenario and additional elements from the production optimisation scenario
- End of year 4: Fully operational prototype, combining all elements from the two domains into the final platform prototype and demonstrator

After the successful completion of a prototype cycle, each RTD work package will analyse and report their development results, RTD experiences, Lessons Learned in the development and integration work and other relevant knowledge gained during the development cycle. Moreover, knowledge gained from formal testing and system integration will be collected together with the latest

developments in technology, regulatory affairs and markets, which influence the outcome and exploitability of the project.

3.4 The ebbits approach to Lessons Learned

Lessons Learned are a principal component of a project culture committed to Knowledge Management. Lessons Learned help to support project goals in the RTD work by:

- Promoting recurrence of successful outcomes
- Precluding the recurrence of unsuccessful outcomes.

As part of the continuous improvement programme adopted by the ebbits Project Board, a systematic and continuous collection, indexing and dissemination of Lessons Learned will be undertaken in WP2.

This section will establish criteria for the Lessons Learned process and discuss how to turn Lessons Learned into Lessons Applied.

Lessons are learned during project RTD work, during testing and integration, as a part of the validation of project prototypes and from watching developments in technology, market and regulatory standards. Lessons can thus be learned throughout the project work. As such, Lessons Learned constitute both individual and organisational knowledge and understanding gained by experience, either negative (missed targets, solutions that do not work as expected, wrong choice of technology) as well as positive (easier implementation than expected, faster response time, more interoperable devices than expected).

In order to implement a workable Lessons Learned process, we need first to define what we understand with the term "lesson". We use the following characterisation for a lesson:

- It must be significant in terms of the project progress and ability to meet its goal
- It must be valid, i.e. the experience gained must be repeatable
- It must be applicable to the ebbits project
- It may contain or address pertinent info
- It may provide information of interest

Not all experiences will qualify as being Lessons Learned and it is important that reported Lessons Learned not merely restate existing information and existing experiences not related to ebbits work.

The ebbits Lesson Learned process has six steps:

- Collection
- Verification
- Storage
- Dissemination
- Reuse
- Identification of improvement opportunity

3.4.1 Collection

The collection process focuses on collecting Lessons Learned from many sources internal and external to the project. The collection will be undertaken in practically all work packages.

In particular, the RTD work undertaken in the technical work packages will provide a large amount of Lessons Learned, by virtue of the many researchers participating in this work and the many small and large experiences gained individually and as teams. The challenge here is to identify and properly describe the Lessons Learned and filter them according to significance, validity, and applicability to the ebbits project.

Testing and validation of the prototypes will also provide a range of experiences that can be classified as Lessons Learned. Finally, the supporting work undertaken in the form of

demonstrations, training and dissemination will, at least in principle, contribute Lessons Learned to the project.

3.4.2 Verification

Verifying the collected lessons according to established standards is the second step in the process. All Lessons Learned must be verified for correctness, significance, validity, and applicability. The verification will be performed by the WP2 team together with the Technical Manager and relevant WP leaders. The Technical Manager will decide to add and remove Lessons Learned as necessary.

Some of the criteria that may be used for verification are:

- Relationship with the project flow
- Relevance to the project outcome
- Significance in terms of quality parameters such as robustness, ease of use, functionality
- Research aids used
- Systemic process issues
- Credibility or reputation of the originator

3.4.3 Storage

The Lessons Learned will be entered into a dedicated area of the ebbits Wiki. The area has been created and is maintained by WP2. It contains a simple categorisation tagging for filtering purposes. For the sake of simplicity, a very simple template will be provided with no special structure or format needed. An example is shown in Figure 2.

The Lessons Learned repository will act as an organisational memory for experiences accumulated during the course of the project.

WP8 Lessons Learned in the first development cycle

Cat.	Org.	Experience and knowledge gained	Lesson learned
SWD	FIT	It is hard to make automatic build of LinkSmartMiddleware? as the automatic build solutions are difficult with OSGI	• Think about the automatic build of codes in advance and decide on a common tool • Watch development of ivy, maven
NET	FIT	Wireless communication is difficult in the scenarios. In famrs no good coverege in manufacturing too big EM pollution	• Wireless communication methods have to be investigated
PRO	ISMB	Applications need to access different physical-world parameters on PLCs with different periodicities, eventually changing dynamically over time.	The PWAL of a PLC should support the configuration of dynamic polling policies of devices involved in a manufacturing plant
PRO	ISMB	In many cases, applications need to be aware of some characteristics of the Physical World (e.g. the periodicity of detecting and exposing information data by a physical sensor, the duration of a measurement).	The PWAL should support dynamic polling policies of devices involved in a manufacturing plant.
RTD	ISMB	In order to perform some energy-related measurement, it is often needed to execute computations which highly depend on time precision (e.g. integrals of power over time).	Time-stamping at low level, with significant accuracy, might be needed in some use-cases
SWD	ISMB	Memory data blocks inside a PLC contain raw data representing Physical World information. By using suitable symbols, the above information can be clearly identified. Additional configuration is then needed to enrich their description with process-related meaning.	The PWAL must provide suitable methods to give users the possibility to enrich PLC data with additional meta-data needed by ebbits.

Figure 2 – Example of Lessons Learned posted in the Wiki repository

3.4.4 Dissemination

Obviously a very important part of the process is to inform other users in the feedback cycle. All project workers are encouraged to continuously consult the Lessons Learned repository, not only with the purpose of reporting, but also to continuously follow Lessons Learned by other project partners.

Once in every iteration cycle, the Lessons Learned will be documented as described in Section **Error! Reference source not found.**.

3.4.5 Reuse

Because Lessons Learned in one work package may affect other work packages, each ebbits partner is encouraged to take advantage of all the Lessons posted under the individual work packages. The WP leaders have a responsibility to consult the Lessons Learned repository regularly and at least before any major decision affecting the scientific work and project outcome is to be made. The WP leaders are obliged to take part of the engineering process of requirements, which is based on a timely assessment of the reported Lessons Learned.

3.4.6 Identification of improvement opportunity

The last step in the process is the identification of incremental and innovative improvements and additions to the initial set of requirement specifications for the project.

From the Lessons Learned, relevant new and/or updated requirements will be extracted. The identification, formulation and validation of requirements will be performed by a WP2 team together with the Technical Manager and relevant WP leaders. The team will update the GForge Issue Tracker with the extracted requirements and report the change requests for the first cycle in the deliverable *D2.8.1 Change request and re-engineering report 1*. See also Section **Error! Reference source not found.**.

4. Lessons Learned in the First Cycle

This Section contains all Lessons Learned in cycle 1 and the subsequent analysis. To facilitate referring to individual Lessons Learned they have been named LL followed by the relevant work package number and Lesson number (as they appear in the Wiki repository), e.g. LL WP3-1. Though reported for one work package, Lessons Learned may very well affect other work packages, and therefore it is important to keep track of all new entries the repository on a regular basis.

The outcome of this process is the identification of a series of improvement opportunities, some of which may result in the need for new, changed and deleted requirements. The changes in requirements are reported in the deliverable *D2.8.1 Change request and reengineering report 1*.

A total of 49 Lessons Learned has been reported in the first iteration cycle, resulting in the addition of 45 new requirements and update of 4 requirements. No requirements have been deleted.

4.1 Lessons Learned in WP2

WP2 is responsible for Requirements engineering and validation. IN-JET is the WP leader and 1 Lessons Learned has been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
IN-JET LL WP2-1	External experts typically have very full calendars and cannot be expected to attend Vision scenario workshops at short notice. This may conflict with the time constraints imposed at the start-up phase of a project	It may be unrealistic to expect external participation in Vision scenario workshops with early due dates for resulting deliverables

4.1.1 Analysis of Lessons Learned

LL WP2-1 is non-technical in nature. This Lesson is not directly applicable to the ebbits project, because is involves the initial planning of the project work. But the result of the Lesson, nonetheless, has been the decision to regard the User Partners as experts in their individual domains and rely on their knowledge for eliciting requirements and defining Vision scenarios. Though generally it adds to the visionary dimension to engage external experts, this approach was considered reasonable because the proposed business applications of ebbits are sufficiently anchored in the present.

This Lesson has not resulted in changes to the initial set of requirements.

4.2 Lessons Learned in WP3

The work undertaken in WP3 involves Enterprise frameworks for lifecycle management. TUK is the WP leader and 3 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
TUK LL WP3-1	In *D3.1 Enterprise use cases* description of general use cases was "too general" and should be more consistent with the modelling guidelines	The indicators for measuring overall efficiency of the ebbits application should be defined (in discussion with COMAU and TNM). Based on the discussion results, relevant processes and general use cases should be modified/elaborated in more details
TUK LL WP3-2	Potential general optimisation methodologies and methods have been identified, but their implementation would require additional overhead	ebbits platform should be able to expose all the information needed to support the adoption of commonly used optimisation methods. On this

	(including use of additional human resources)	basis new/modified optimisation methods can be proposed in the future
ISMB LL WP3-3	In the manufacturing scenario, although expensive manufactured items are provided with identifiers, several items are manufactured in batches	Having items manufactured in batches can have an impact on accounting energy-related operations (e.g. we have a batch of 100 components which costs 1 KWh to be manufactured: if only 20 of them are used in a final product, it is necessary to understand how to account the consumption to the final product)

4.2.1 Analysis of Lessons Learned

WP3 is not technical in nature. Thus LLs in this WP have not resulted in changes to the user requirements; rather they have impacted on some further project activities and deliverables (mostly within WP3).

According to LL WP3-1 description of general use cases was too general, which means that additional information is needed for more detailed description of these processes. This additional information can be obtained from description of processes reported in *D3.4 Business framework for online OEEE applications for production and energy optimisation* and in *D3.5 Business framework for online food traceability in lifecycle perspective*. Based on this, more detailed general use cases can be created.

LL WP3-2 implies that additional information is needed to decide on the optimisation method to be used in the manufacturing domain. This information is expected from the description of use cases in the above mentioned deliverable D3.4. Current analysis indicates that modification/combination of the existing metrics (as a basis for optimisation) will be used, rather than new metrics.

LL WP3-3 reports that some goods are manufactured in batches. In such cases, calculation of energy needed for manufacturing of these goods will be based on estimations (e.g. energy consumption for production of the whole batch divided by the number of units in the batch).

4.3 Lessons Learned in WP4

The work undertaken in WP4 relates to Semantic knowledge infrastructure. SAP is the WP leader and five Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
SAP LL WP4-1	Appropriate knowledge representation formalism is needed to avoid the high complexity in the knowledge manipulation process	The knowledge representation formalisms must be as small and easy as possible, but as expressive as necessary for our scenarios. The dialects of OWL-Lite came out as the best candidates
SAP LL WP4-2	Common ontology namespace is important	In order to be able to connect the several ontology modules defining the overall ebbits ontology and possibly also the external ontologies, an ontology identifier has to be specified. Ebbits ontologies will share namespace http://www.ebbits-project.eu/ontologies. In future, this identifier may enable inclusion of the ebbits knowledge bases into the external ontologies

TUK LL WP4-3	The strategy of ontology reflection of the physical devices used in HYDRA would probably lead to the information redundancy for the ebbits use cases	The Hydra semantic discovery strategy and the representation of the physical devices in the ontology must be adapted for ebbits purposes. The solution must be able to create unique mapping between physical devices and their semantic reflections. As large numbers of devices are expected in the ebbits use cases, the solution must be able to preclude redundant information
TUK LL WP4-4	The ebbits ontology manager should provide dedicated services for accessing specific knowledge	The use cases implemented in the M12 Automotive Manufacturing demo showed that the application developers must be able to access specific knowledge in a specific way. As presently knowledge is accessed using SPARQL queries, dedicated services would mean not having to learn new query languages and hence using the ontologies in a more suitable way
TUK LL WP4-5	The development of the semantic models must be driven by the real use cases instead of the theoretical assumptions to avoid the unnecessary complexity in the knowledge	The design of the semantic models based on the theoretical assumptions of the potential usage may lead to the complex models, which are never used in the practice. The development of the ontologies must be driven only by the real use cases with respect to the future usage. The semantic models should not contain the information, which is not used in the practice. This approach enables to hold the ontologies simple, easy to understand, easy to maintain and, of course, the computation complexity and the query answering response much faster

4.3.1 Analysis of Lessons Learned

The analysis of the five Lessons Learned has resulted in the definition of 5 new requirement; no requirements were modified or deleted.

LL WP4-1 reports the need for appropriate knowledge representation formalism. The search for suitable candidate led to the new requirement *[ebbits-464] OWL-lite will be used to model ontologies*.

LL WP4-2 concerns the importance of a common ontology namespace and led to the new requirement *[ebbits-465] Ontology namespace*.

LL WP4-3 reports the need for unique mapping between physical devices and their semantic reflections, which led to the new requirement *[ebbits-383] Devices should be annotated with id, type, name, location and current/historical data*.

LL WP4-4 requires of the ontology manager to provide dedicated services for accessing specific knowledge, resulting in the new requirement *[ebbits-466] Query the ontologies conveniently*.

LL WP4-5 reports that development of the semantic models must be driven by the real use cases instead of theoretical assumptions, leading to the new requirement *[ebbits-467] Only relevant parts in the ebbits ontologies*.

4.4 Lessons Learned in WP5

The work in WP5 involves Centralised and distributed intelligence. FIT is the WP leader and 11 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
FIT LL WP5-1	Sensor fusion algorithms vary greatly and cannot be generalised in a single module	We need a plug-in system that allows new algorithms (e.g. mathematical calculations) to be added dynamically

		at runtime
FIT LL WP5-2	Sensor fusion calculation module is needed by different components e.g. proxy devices and context manager	Sensor fusion calculation module should be decoupled from the context manager
FIT LL WP5-3	Due to deviations of the data sent by different sensors inferring context needs to take uncertainties of data into account	Soft-logic algorithms are needed to calculate the certainty of a context
FIT LL WP5-4	Application development can benefit from a central interface that could deliver the real time context information of the devices	Context manager needs to be implemented which should monitor context changes of devices and processes
FIT LL WP5-5	Rules are needed to describe relationships among sensor data and to infer context	A rule engine is needed for the sensor fusion and the context manager
FIT LL WP5-6	Dynamically loaded libraries (e.g. DLL, JAR, OSGI bundle) could contain malicious code	Dynamically loaded libraries must contain a valid signature in order to prevent security breaches in the system
FIT LL WP5-7	Libraries could contain functionality that should not be available to all kinds of applications (e.g. calculation of quality rating of meat should only be allowed for slaughterhouse application but not for consumer application)	The dynamic loading of libraries must be restricted through policies
FIT LL WP5-8	Sensor fusion modules need to access historical data	Event manager will need to store all events and provide a query interface to retrieve historical data
FIT LL WP5-9	Business rules are triggered by business events to perform actions e.g.: to notify enterprise applications when the process starts and finishes or how many resources have been consumed for the process	Context manager should monitor and analyze sensor events and as a result generate business events
FIT LL WP5-10	Performance of the context manager could be affected by too large number of sensor events	Strategy to support scalability needs to be developed
FIT LL WP5-11	In a dynamic environment entities (like mobile devices or sensors) constantly enter and leave the context boundary of an application	The context model needs to be extensible during runtime

4.4.1 Analysis of Lessons Learned

The analysis of the 11 Lessons Learned has resulted in the definition of 14 new requirements and the modification of one requirement; no requirements were deleted.

LL WP5-1 describes the need for a plug-in based approach that allows loading sensor fusion algorithms at runtime. This analysis results in the definition of one new requirement [ebbits-451].

In LL WP5-2 we agreed that the sensor fusion calculation module must be decoupled from the context manager. This also results in the definition of a new requirement [ebbits-452].

LL WP5-3 states that soft-logic algorithms are required to calculate the certainty of a given context. For instance, due to deviations in collected sensor data the context manager needs to take into account uncertainties in this collected data. For this purpose two new requirements [ebbits-131] and [ebbits-450] have been created.

LL WP5-4 reflects the implementation of the context manager while considering monitoring contextual changes of both devices and processes. This has resulted in two new requirements [ebbits-140] and [ebbits-154] being created. Subsequently the status of requirement [ebbits-154] has been set to duplicated, as [ebbits-43] already covers it.

In LL WP5-5 we experienced that the sensor fusion and context manager needs a reusable rule engine. This meant the addition of a new requirement [ebbits-453].

LL WP5-6 states that libraries that are loaded dynamically must contain a valid digital signature in order prevent security breaches in a system; the integrity of the sensor fusion and context manager depends also on the integrity of their dynamically loaded libraries. Hence, a new requirement [ebbits-135] has been created.

LL WP5-7 explains that the dynamic loading of libraries needs to be restricted by policies. For this purpose one new requirement [ebbits-458] has been defined.

LL WP5-8 points out that the Event Manager needs to store all sort of events and provide a query interface to retrieve historical data, as for instance the sensor fusion module requires to access such historical data. Therefore, requirement [ebbits-64] has been updated. This Lesson influences the work to be done in WP7 Event management and service orchestration.

LL WP5-9 outlines that the context manager needs to monitor and analyse sensor events and as a result trigger business events. This analysis results in the definition of two new requirements [ebbits-454] and [ebbits-455].

In LL WP5-10 it is reported that a strategy has to be developed that supports scalability. For instance, the performance of the context manager could be affected by an excessive number of sensor events. For this purpose, a new requirement has been defined [ebbits-456].

LL WP5-11 mainly reflects that the context model must be designed to be extensible at runtime. This analysis results in the creation of two new requirements [ebbits-134] and [ebbits-139]. Note: [ebbits-139] is dependent on [ebbits-134].

4.5 Lessons Learned in WP6

The work in WP6 revolves around Mainstream business systems. SAP is the WP leader and two Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
SAP LL WP6-1	A potential user of the ebbits platform in the traceability scenario, e.g. the farmer, will not only be working at his desk	A mobile access (at least a read access) to the data in the ERP system would be desirable
SAP LL WP6-2	There are a lot of different mobile devices with various platforms available on the market right now	In order to support different mobile platforms, it could be convenient to access the required information by using web browsers

4.5.1 Analysis of Lessons Learned

The analysis of the two Lessons Learned has resulted in the modification of one requirement; no requirements were added or deleted.

LL WP6-1 specifies the need for mobile access, which should be at least read access, to the high-level data in the ERP system. This analysis resulted in a modification of [ebbits-99].

LL WP6-2 is related to LL WP6-1, stating that in order to be able to use a variety of mobile platforms the mobile access should be browser-based if possible. This also involved the modification of [ebbits-99].

4.6 Lessons Learned in WP7

The work undertaken in WP7 deals with Event management and service orchestration. CNET is the WP leader and seven Lessons Learned have been collected and validated from this WP.

Org.	Experience and knowledge gained	Lesson Learned

No.		
ISMB LL WP7-1	An action executed by the system may be dependent on more than one event, and some of them could have occurred in the past. It would be useful to maintain an event history onboard ebbits nodes	The proposed architecture needs to support a persistent events storage on some of the more event-consuming nodes
CNET LL WP7-2	The platform must be able to handle a large number/high frequency of parallel event streams. Large=? High =?	Need lower layer filtering and fusion (WP5), as well as higher level (WP7) aggregation and semantic enrichment of events
CNET LL WP7-3	Depending on the network reliability, events may be delivered out of order and in parallel streams, and may potentially be lost due to failures	The platform must guarantee reliable delivery of events. A proper ordering of events must be provided. It should be possible for applications and event processing components to query the properties of a specific network installation, so as to determine a level of reliability
CNET LL WP7-4	The initial tests indicates that the current design does not scale properly	The current event system based on the publish/subscribe principle has limitations in terms of scalability. Alternatives should be considered
CNET LL WP7-5	It must be possible to interpret events in the context of the different layers in the architecture (from PWAL to BRAL =Business Rules Adaptation Layer)	The event core ontology (or simplified/compiled subsets thereof) should be accessible by all event processing components
CNET LL WP7-7	The "data management" parts of ebbits are to be further analysed specifically with respect to eventing	Need for coordination of issues related to vocabularies, ontologies, knowledge representation and data/events persistency (WP3,4,7)
CNET LL WP7-7	The current systems design work lacks a proper overall coordination. Despite requirements and "lessons learned" compilations, the process does not fully capture and generalize the experience gained into reusable knowledge	As part of the design and implementation work, the project should compile a set of design guide-lines and principles for event-based and service-oriented architectures. It should have a strong IoT focus

4.6.1 Analysis of Lessons Learned

The analysis of the seven Lessons Learned has resulted in the definition of six new requirements; no requirements were modified or deleted.

LL WP7-1 states that actions may depend on multiple possibly historic events. This introduces two new requirements *[ebbits-461] Dependencies on past events possible* and *[ebbits-218] An event history should be maintained*.

LL WP7-2 and WP7-4 address scalability and propose multi-layer filtering and aggregation. Two new requirements *[ebbits-462] Scalable event processing* and *[ebbits-463] Semantic event processing* have been added. LL WP7-4 is linked to LL WP8-9, see below.

LL WP7-3 addresses reliable delivery of events. This is supported by the new requirement *[ebbits-217] Events must be possible to order in the actual event sequence*.

LL WP7-5 states the need to interpret events on several layers in the platform architecture. This is addressed by the new requirement *[ebbits-463] Semantic event processing*.

LL WP7-6. Event management should be based on well-defined semantics, i.e. a vocabulary encoded in a rich semantic model, such as an ontology. The ebbits platform also deals with "data" management, in terms of modelling, storage and exchange of information. This LL states the need

to coordinate the selection and design of vocabularies and models across the work packages dealing with event and data management. It is the basis for the new requirements *[ebbits-463] Semantic event processing* and *[ebbits-220] Event model based on common vocabulary*, which more explicitly express the need for an event ontology to be based on a vocabulary related to the overall ebbits data and process models. This particularly involves WP3, WP4, WP6 and WP7.

LL WP7-7 is a remark on the need for improved coordination and collaboration in the design process in order to maximise knowledge development and sharing. It does not imply any new requirements at this stage.

4.7 Lessons Learned in WP8

The work in WP8 takes care of Physical world sensors and networks. ISMB is the WP leader and 17 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
FIT LL WP8-1	It is hard to make automatic builds of LinkSmart Middleware as the automatic build solutions are difficult with OSGI	Think about the automatic build of codes in advance and decide on a common tool. Watch development of ivy, maven. This is also relevant for WP9
FIT LL WP8-2	Wireless communication is difficult in the scenarios considered in the project. For instance, in farms, there could be poor radio coverage, while in manufacturing EM pollution could be present	Wireless communication technologies have to be analysed with respect to the considered scenario
ISMB LL WP8-3	Applications need to access different physical-world parameters on PLCs with different periodicities, eventually changing dynamically over time	The PWAL of a PLC should support the configuration of dynamic polling policies of devices involved in a manufacturing plant
ISMB LL WP8-4	In many cases, applications need to be aware of some characteristics of the Physical World (e.g. the periodicity of detecting and exposing information data by a physical sensor, the duration of a measurement)	The PWAL should support dynamic polling policies of devices involved in a manufacturing plant
ISMB LL WP8-5	In order to perform some energy-related measurement, it is often needed to execute computations which highly depend on time precision (e.g. integrals of power over time)	Time-stamping at low level, with significant accuracy, might be needed in some use-cases
ISMB LL WP8-6	Memory data blocks inside a PLC contain raw data representing Physical World information. By using suitable symbols, the above information can be clearly identified. Additional configuration is then needed to enrich their description with process-related meaning	The PWAL must provide suitable methods to give users the possibility to enrich PLC data with additional meta-data needed by ebbits framework
ISMB LL WP8-7	The name of the PLC symbols is usually decided by PLC developers using a custom approach	The PWAL should include mechanisms to associate custom symbols to ontologies selected within the ebbits environment
ISMB LL WP8-8	Memory writing processes in PLCs could trigger sharp program interruption if the data inserted is not correct or adequate	The PWAL should be aware of writing on the PLC memory appropriate data types and value ranges regarding the

		different variables
ISMB LL WP8-9	In some use case, multiple (e.g. 10, 20, or even 50) parameters change at the same time onboard the PLC (e.g. due to some synchronous polling of wired sensors). This causes the generation of a large number of events at the same time, which, on very large-scale deployments, can originate scalability problems	It is needed to explore methods allowing aggregation/disaggregation of events to enhance scalability, without compromising the current mode of operations
ISMB LL WP8-10	When the PWAL tries to read or write some critical variable onboard PLCs via OPC, sometimes weird values are returned due to the PLC program using those variables when OPC server accesses them	A semaphore-based policy must be agreed with PLC developers, to ensure that all variables exposed to ebbits are safe
ISMB LL WP8-11	When the remote LinkSmart environment is restarted while developing, many variables in the local environment (e.g. the link to the Event Manager) stop properly working	Similar events should automatically generate some explicit exception in order to enhance code safety
ISMB LL WP8-12	Devices accessing the ebbits network could use different network managers depending on the available network interface	Devices accessing ebbits by using non-corporate or external networks (3GPP) should be able to detect which border network manager they must connect to
ISMB LL WP8-13	Data storage needs within multi-radio could suddenly and quickly grow in case network coverage is not good and large amounts of data need to be transferred	Some mechanism should be designed and developed in order to maintain the data storage as low as possible e.g., by implementing application specific policies
ISMB LL WP8-14	In multi-radio devices, the network selection should be based not only on the availability and the current status of the network resources but also on application requirements	The Multi-radio Manager should provide the mechanisms capable of acquiring the necessary information and using it for selecting the most proper network interface
ISMB LL WP8-15	The presence of interference in a specific environment might significantly degrade the performance of a 6LoWPAN	Frequency agility features should be included in 6LoWPAN networks being integrated into ebbits platform. This would improve overall system reliability
ISMB LL WP8-16	Multi-radio could need additional features, besides network selection, in order to provide network reliability and efficiency to the ebbits framework	The use of data storage and delay tolerance networking capabilities combined with multi-radio could be beneficial for getting more reliable data transmission
ISMB LL WP8-17	A multi-radio device, while representing the same physical device, could have different low level addresses (e.g. IP address) according to the selected network interface	Mechanisms (e.g. cryptographic tickets) could be implemented in order to avoid the re-generation of new HIDs in multi-radio devices when connecting to new network managers

4.7.1 Analysis of Lessons Learned

The analysis of the 17 lessons learnt has led to the definition of 20 new requirements and the modification of 2 existing requirements. No requirements have been deleted.

LL WP8-1 and LL WP8-11 describe some LinkSmart specific issues, like the lack of automatic builds due to some technical limitations of OSGI or proper handling of restarting remote environments by local instances. These issues define two new requirements, *[ebbits-468] LinkSmart should support automatic builds* and *[ebbits-469] Local LinkSmart instances should properly handle local variables when remote environments are restarted*. LL WP8-1 is also relevant for WP9.

LL WP8-2 exposes the risks of using wireless communications in both scenarios, which should be taken into account when choosing the wireless technologies and designing deployment architecture.

LL WP8-3 and LL WP8-4 address the issue of different polling policies that should be supported by the PWAL, leading to the definition of a new requirement *[ebbits-470] PWAL should support reconfigurable dynamic polling policies*.

LL WP8-5 highlights the necessity of accurate time-stamping at data acquisition level for different applications; therefore four new requirements have been defined: *[ebbits-155] Synchronization of acquired data is necessary, [ebbits-219] Event history size and/or time span should be configurable, [ebbits-471] ebbits should implement a distributed time-dissemination and synchronization service* and *[ebbits-472] PWAL should support accurate time-stamping of data acquainted*.

LL WP8-6 and LL WP8-7 discuss some issues faced in the identification and enrichment of raw data from the physical world (in particular a PLC) through the PWAL, thus resulting in the following new requirements: *[ebbits-473] PWAL should expose suitable methods in order to enrich raw data* and *[ebbits-474] PWAL should be able to match PLC symbols with ebbits ontologies*.

Moreover, the existing requirement *[ebbits-383] Devices should be annotated with id, type, name, location and current/historical data (status, work in progress, consumables levels, quality record, energy consumption, energy profile, planned/unplanned intervention/maintenance, fault info, etc.)* has to be modified.

LL WP8-8 and LL WP8-10 present some potential errors in the writing process on the PLC, thus necessitating two new scheduling and error control requirements: *[ebbits-475] PWAL should adopt a lock and semaphore-based and policy to the access of PLC memory* and *[ebbits-476] PWAL should implement an error control strategy to assert correct data type and values written to the PLC*.

LL WP8-9 points out scalability issues in events generated from the PWAL, coped with by two new requirements: *[ebbits-477] PWAL should implement a heterogeneous multi-data aggregation in single events* and *[ebbits-478] PWAL should expose basic feature extraction and sensor fusion functionalities (e.g., moving average, decimation, filtering, etc) in order to minimize scalability issues*. This Lesson is linked to LL WP7-4 above.

LL WP8-12 and LL WP8-17 outline some potential LinkSmart Network Manager association problems when using multi-radio. Such issues will be tracked with the following new requirements: *[ebbits-479] Multi-radio devices should be able to detect which LinkSmart Network Manager to connect/migrate to, according to the current network interface active* and *[ebbits-480] Multi-radio devices should avoid re-generation of HIDs when migrating to a different LinkSmart Network Manager*.

In addition, two existing requirements were identified for extensions in their definitions: *[ebbits-51] The network infrastructure needs to have self-configuration capabilities* and *[ebbits-53] New products should be networked with mainstream enterprise systems easily and cost-efficiently*.

LL WP8-13 and LL WP8-16 report some network reliability features that should be supported by multi-radio devices, features covered with two new requirements: *[ebbits-481] Multi-radio devices should be use local data caching and delay tolerance networking* and *[ebbits-482] Multi-radio devices with local data caching should implement suitable application specific data-expiration policies in order to prevent cache overflows*.

LL WP8-14 details some policies for the multi-radio network selection strategy, summarised in two new requirements: *[ebbits-483] Multi-radio devices should be able to gather information about their network interfaces needed for the selection policies* and *[ebbits-484] Multi-radio devices should select the most proper network interface according to the application requirements*.

LL WP8-15 explains the benefits of frequency agility in 6LoWPAN networks, which could be introduced as a new requirement: *[ebbits-485] 6LoWPAN networks should include frequency agility features in order to enhance the overall system reliability*.

4.8 Lessons Learned in WP9

The work in WP9 takes care of Platform integration, test and deployment. CNET is the WP leader and the following Lessons Learned have been collected and validated at the end of the first iteration from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned
CNET LL WP9-1	Too little/to late integration work, has a negative impact on the work process and the results	Every major platform release/milestone, should be foregone by minimum two IRL integration workshops
CNET LL WP9-2	LinkSmart OSGi development environment is error prone with regards to the configuration parameters. For instance it is possible to lose all security, Network Manager and Event manager settings when rebuilding the system	Convince LinkSmart to set up configurations in a text based file instead
CNET LL WP9-3	Integration work hampered by that all partners involved in the integration did not turn up to the integration meetings.	All partners with code to integrate should be attending the integration meetings.

4.8.1 Analysis of Lessons Learned

The analysis has not led to any new or modified requirements for the platform or its architecture per se. However, it is recommended that these LLs result in updates to the procedures in the Test and Integration Plan (D9.1), with respect to integration procedures.

LL WP9-1 and LL WP9-3 imply that the practical systems integration work has to be more carefully planned, executed and adhered to. LL WP9-2 requires LinkSmart providers to take the necessary technical measures.

4.9 Other work packages

In the first iteration cycle no Lessons Learned were reported from WP1, WP10, WP11 and WP12, mainly due to their nature or the timing of the work to be done.

5. Results of Usability Evaluation

This Section outlines usability results of the prototypes that were demonstrated at the Review meetings in month six (M6) and month twelve (M12) of the ebbits project. At this early stage only usability of developer tools has been evaluated, as experienced from an application developer perspective.

5.1 M6 Demo Prototypes

The M6 prototypes were additional to the prototypes prescribed at the end of each annual cycle in the Description of Work.

The goals of these M6 demo were:

- To provide an overview of ebbits and test-bed environment to be evolved further during the project
- To get the technical team to work together
- To involve users at an early stage

Both the Automotive Manufacturing and the Food Traceability M6 prototypes were built on top of LinkSmart middleware, developed in the Hydra[2] project.

5.1.1 Automotive Manufacturing M6 Demo Prototype

The first Automotive Manufacturing demo was driven by a common industrial task of spot welding through automatic robots in automated manufacturing plants. Spot welding requires a large amount of power and water in order to cool the tips, transformer and triacs. Monitoring power and water consumption is a first step for enabling process optimisation, accounting and analysis of key performance indexes (e.g. OEEE). The vision of ebbits is to interconnect robots in the plant to spatially differently located manufacturing execution systems (MES). An MES is fed with sensor data in the plant, enabling it to actuate the robots as required. For the sake of simplicity, the welding robot is emulated by a simple water loop system controlled by a commercial off-the-shelf (COTS) water pump installed at the premises of ISMB (Torino, Italy). The consumption is monitored through a COTS water meter and a COTS electricity meter (Smart Plug) that is also used to emulate a simple switch On/Off control. The user can interact with the devices located in Torino through a GUI in Birlinghoven, Germany.

[2] Hydra EU Project, http://www.hydramiddleware.eu/

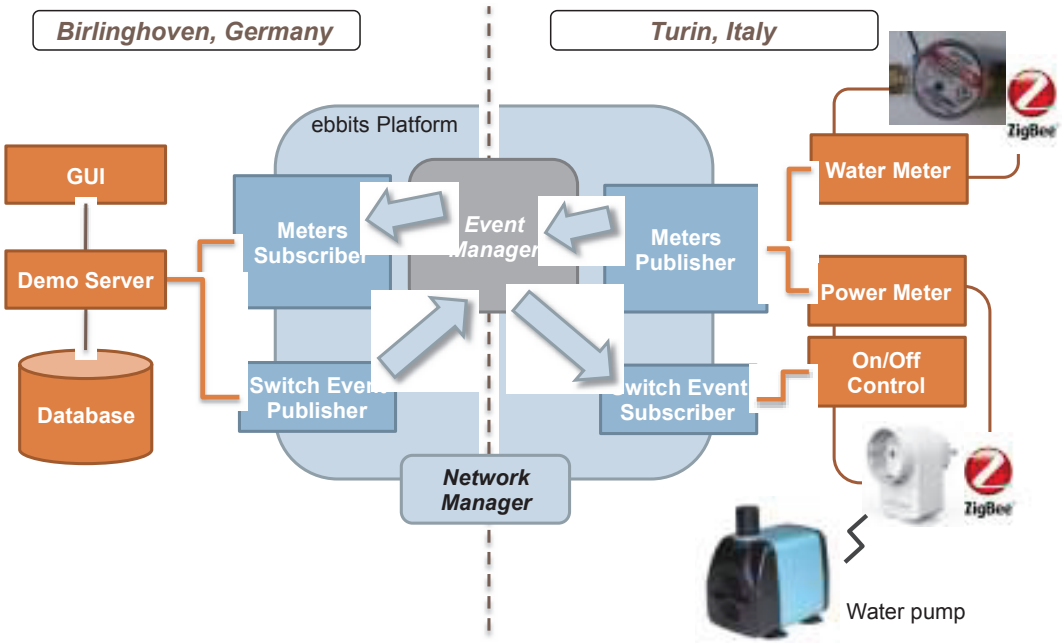

Figure 3 – Automotive Manufacturing M6 Prototypical Demo

In the manufacturing scenario two kinds of events exist:

- Energy event, i.e. regular measurement of the instant and total consumption (of the water and electricity).
- A switch event, i.e. either the pump is switched 'On' or 'Off'.

5.1.2 Food Traceability M6 Demo Prototype

The first Food Traceability demo seeks to enhance the communication workflow of medical data between veterinarians and farmers. The argument is that in order to meet a growing demand for food safety, an acceptable level of transparency must be provided over the entire lifecycle of a product. For the demo, the focus has been on tracing the medication of pigs to prevent medicated animals from being sent to slaughter by mistake and ensuring that it can be traced back in case of contamination. The developed system comprises a medical application deployed on a personal digital assistant (PDA) that feeds data through the LinkSmart middleware into an enterprise resource system (ERP), thus enabling farmers to monitor the state of health of their pigs via a web interface. In practice, it means that a vet examines a sick pig, identified by an RFID tag. After medicating the pig, the vet documents the treatment using a PDA. Finally, the data is synchronised with the farm ERP system by means of LinkSmart middleware. The farmer can then monitor the state of health by using a dedicated web application.

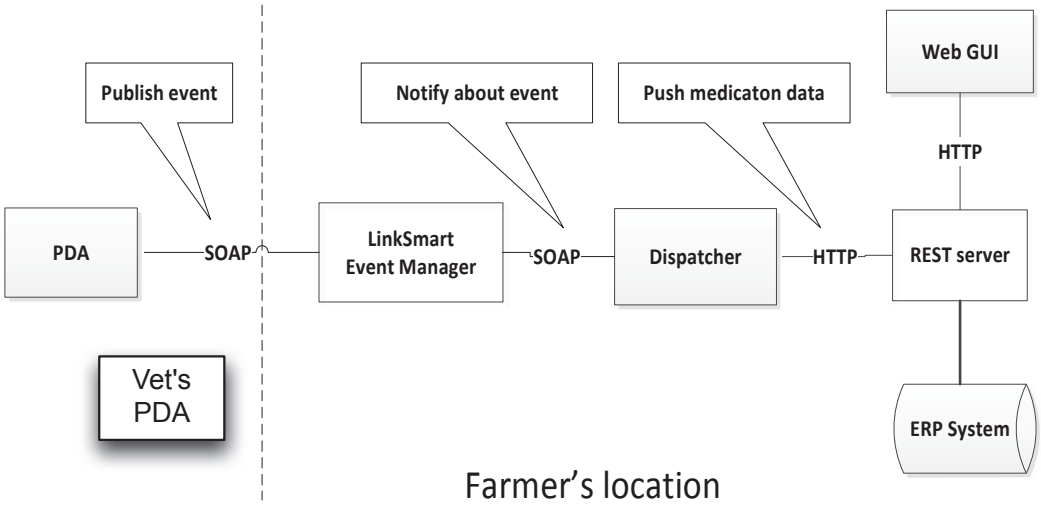

Figure 4 – Food Traceability M6 Prototypical Demo

5.1.3 Conclusion

In general, the LinkSmart middleware provided adequate support for application developers while implementing the two prototypes. Essentially, two LinkSmart components have been exploited: a Network Manager and an Event Manager. In the following we summarise pros and cons experienced when leveraging LinkSmart.

The ebbits project strives to enable Internet of Things (IoT) for business-enabled services. In order to approach this vision a sophisticated network routing component is required. To accomplish this, the Network Manager resides at the bottom of the LinkSmart middleware architecture, providing Web Service based device access independently of the underlying communication technology (ZigBee, Bluetooth etc.). Further, the LinkSmart Network Manager facilitates the detection of services in an overlay P2P network based on queries that expect persistent identifiers as an input. With regard to the manufacturing prototype demo, this was important because the Event Manager Service was deployed at the premises of ISMB in Torino, Italy and the manufacturing GUI deployed at the premises of Fraunhofer FIT in Birlinghoven, Germany (see Figure 3). As the Network Manager implements P2P communication through SOAP Tunnelling to allow communication beyond firewalls or NATs (Milagro et al 2008), we discovered a slight, but acceptable delay. More importantly, we had to disable security, as otherwise we randomly faced an unhandled exception and delay of approx. 40 seconds when executing a LinkSmart Web Service call.

The LinkSmart Event Manager deals with the complexity of publishing and subscribing to events. In general, with respect to software maintenance there is a great benefit from the loose coupling between data source and destination(s) provided by the LinkSmart Event Manager. For example, because the starting and stopping of the pump was realised through publishing of events, using another COTS pump would require minimal changes, as it would only be necessary to adapt the device specific SDK code. However, for future work in all likelihood more complex business events are subject to be handled. For this purpose, the LinkSmart Event Manager has to be extended for semantics, besides encapsulating simple key-value pairs in a topic.

In both prototypical demonstrations a Web GUI based on AdobeFlex[3] was provided. The data being visualised was kept in a database that is accessible via a REST[4] interface. Both application domain prototypes required pushing values to the REST-based database server being exposed through the GUI. For this purpose a generic OSGi utility bundle has emerged, which was used for both demos. However, the manufacturing demo also required the opposite direction of communication, i.e. to

[3] Adobe Flex, http://www.adobe.com/products/flex/
[4] Representational State Transfer

invoke the Event Manager Service to publish events in order to start the water pump. In order to fulfil this feature, a dedicated application specific LinkSmart OSGi bundle has been developed that runs on the same device as the manufacturing GUI. This bundle listens to REST calls and maps this into a LinkSmart event publish call that is routed to the Event Manager being deployed at another device. For future work, a rather generic and reusable approach will offer more flexibility. Of course, the application logic of the GUI could have invoked the Event Manager Service running on a different device also through a SOAP Web Service call applying SOAP Tunnelling, but at this stage the developer of the GUI had no knowledge of consuming LinkSmart enabled SOAP Web Services.

5.2 M12 Demo Prototypes

The M12 Demo Prototypes are explained in detail in *D5.4.1 Multi-sensory fusion and context awareness prototype*.

5.2.1 Automotive Manufacturing M12 Demo Prototype

To summarise, we evaluated the initial architecture of the ebbits platform by asking several developers to develop a typical application in the Automotive Manufacturing domain. They were given a task to develop an application that monitors the physical state of a transformer (Trafo) and the energy consumption of a welding process. The group of developers was divided into application developers and device developers. The device developers were asked to integrate a PLC and a few sensors by creating device proxies. The monitoring sensors consisted of three thermometer sensors. The first sensor measured the water temperature at the beginning of the cooling circuit (Tin), the second sensor measured the water temperature near the transformer (Tmid) and the third sensor measured the water temperature (Tout) at the end of the circuit. Moreover, two energy sensors measuring voltage (V) and current (I) and a water meter (W) were installed.

The sensors were connected to a Siemens S7-400 series PLC which in turn was connected to an Ethernet network. Additionally, there was an OPC server communicating with the PLC. The device proxy used an OPC client library to communicate with the OPC server that accessed the PLCs with Siemens native protocol. The PLC contains a program written using ladder logic and assembler that pulled the data from the physical sensors and wrote the values into memory. The OPC server provided an interface for reading this memory from an OPC client.

Following an event-driven approach (Mani Chandy 2006), the device proxy pulled the sensor values (Tin, Tout, V, I, and ProcessStatus) from the OPC server every 500ms and published the results to the event manager whenever the value had changed relative to predefined thresholds.

The device and application developers then together implemented the context module. The context model was designed and stored in a semantic store. Furthermore, several rules to reason high-level context for instance were defined. To calculate the amount of energy used to produce a body part, the context manager starts summing up the electricity in watts (current times volt) and the amount of water when it receives a "processStart" event and stops if it receives a "processStop" event. The ProcessStatus event is set in the PLC if the RFID tag of an item is read on the conveyor.

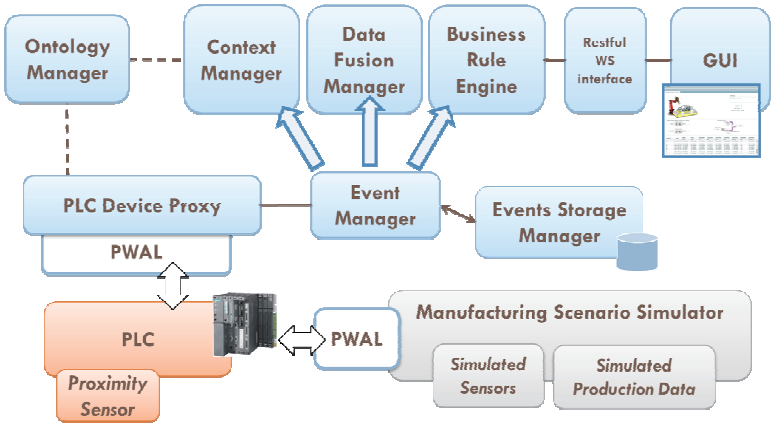

Figure 5 Architecture of M12 Manufacturing Demo

5.2.2 Food Traceability M12 Demo Prototype

The traceability scenario was developed to monitor the state of health of pigs in a farm, such as their drinking and eating pattern and movement behaviour. For the traceability scenario we simulated the feeding and drinking station through an event generator. The device developers however must still develop device proxies to communicate with the event generator. Also, a context model is developed that is stored in the ontology manager. The data fusion manager aggregates the events from feeding and drinking station as well as pigs' movement per day. The context manager to infer the health of the pigs uses this information. The health context is calculated by the end of the day, and an event containing this information is raised. The relationship between events coming from the device proxies is modelled in ontology and stored in the ontology manager. The business rule engine captures the 'state of health' event and according to the rule defined, it forwards the information to the corresponding business application, in this case it was a dedicated GUI for farmers.

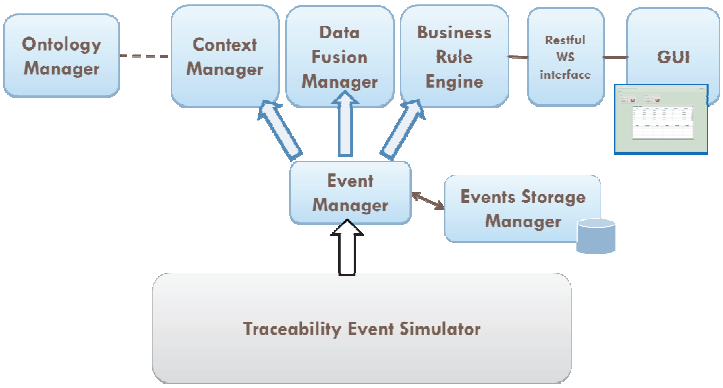

Figure 6 Architecture of the M12 Traceability Demo

The context manager also propagates the context of the pigs to the pens. For instance, if sick pigs are in a pen, the "problem" context of the pen changes to the number of sick pigs. Moreover the context manager keeps track of the location of each pig, and based on this it determines the number of pigs in each pen.

5.2.3 Conclusion

The feedback we obtained from developers reflects that the event-driven approach we used is easy to understand and that the context-aware paradigm supports them in clustering sensor information based on entities. However, application developers also mentioned that the number of events could grow significantly in complex systems. Therefore, a complex event-processing engine should be integrated into the ebbits platform. Based on this, events can be filtered and aggregated. Secondly, developers had problems estimating the relations between context and entities as events are raised for each context change. They would prefer to be able to subscribe to a collection of events belonging to one entity. They also had problem with the arrival time, since time synchronisation has not been implemented yet. Further, developers seem to not entirely understand how and when to query any information from the ontology, because they are not familiar with SPARQL[5] query language. Furthermore, they also would like to have a fully functional rule engine integrated in the middleware.

Another problem that they faced was the reliability of the event manager of the LinkSmart middleware; the handling of a burst of events is not good enough. Some events were dropped, and

[5] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

this caused unexpected behaviour if the event is quite important. Hence, for future work this issue must be tackled, as both application domains ask for high demands and strict constraints.

6. References

(Mani Chandy 2006) K. Mani Chandy Event-Driven Applications: Costs, Benefits and Design Approaches, *California Institute of Technology*, 2006.

(Milagro et al 2008) Milagro, F., Antolin, P., Kool, P., Rosengren, P., Ahlsén M. (2008). SOAP tunnel through a P2P network of physical devices, Internet of Things Workshop, Sophia Antopolis.