Enabling the business-based
Internet of Things and Services

(FP7 257852)

D2.8.3 Change request and re-engineering report 3

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: 2014-03-05

Document control page

Document file:	D2 8 3 Change request and re-engineering report 3 V1.0.doc
Document version:	1.1
Document owner:	L. Christiansen (IN-JET)
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	Lasse Christiansen (IN-JET)	2013-08-09	ToC
0.2	Yves Martin (SAP AG)	2013-09-06	WP6 sections
0.3	Ferry Pramudianto (FIT)	2013-09-17	
0.4	Jan Hreno (TUC)	2013-09-20	
0.5	Paolo Brizzi (ISMB)	2013-09-20	
0.6	Matts Ahlsén (CNET)	2013-09-24	WP7 additions
0.7	Lasse Christiansen (In-JET)	2013-10-17	Revision after review
1.0	Lasse Christiansen (IN-JET)	2013-10-24	Final version submitted to the European Commission
1.1	Lasse Christiansen (IN-JET)	2014-03-05	Corrected typo errors after comments from the Commission

Internal review history:

Reviewed by	Date	Summary of comments
P. Rosengren (CNET)	2013-10-17	Approved with comments.
C. Pastrone (ISMB)	2013-10-02	Approved with comments.

Index:

1. Executive summary

This deliverable reports the results from the third iteration cycle of the work done in *Subtask 2.3.1 Lessons Learned collection and analysis* and contains a complete list of Lessons Learned during this 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. The next cycles of the project is actual the last year of the project.

Also included in this deliverable are the results of *Subtask 2.3.5 Requirements re-engineering*. The resulting new and updated requirements arise from the analysis of Lessons Learned and from the continuous technology, regulatory standards and market watch.

The document includes the content of what was originally planned as two separate deliverables, i.e., *D2.7.3 Lessons Learned and results of usability evaluation 3* and *D2.8.3 Change request and reengineering report 3*, combined under the name of the latter. This measure was taken to reduce duplication and redundancy.

The ensuing document feeds into deliverable *D2.9.3 Updated requirements report 3.*

1.1 Lessons Learned

Section 4 contains all Lessons Learned in cycle 3 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 Section 5.

The Lessons Learned have been collected and reported per work package. Details can be seen in the table below in Section 1.2.

In the second cycle no Lessons Learned have been collected in WP1, WP2, WP3, WP9, WP11 and WP12.

In total, the second iteration cycle yielded 22 Lessons Learned.

1.2 Requirements Engineering

Section 5 describes the requirement engineering work performed in the third iteration cycle. This has resulted in the creation of 3 new requirements, modification of 2 requirements and deletion of 3 requirements, relative to the complete list reported in *D2.9.2 Updated requirements report 2*. As expected, these changes arise mainly from the efforts of the technical work packages.

Work package	Lessons Learned	New requirements	Updated requirements	Deleted requirements
WP4	5	0	0	0
WP5	5	0	0	0
WP6	2	2	0	0
WP7	2	0	0	0
WP8	8	1	2	3
WP10	0	0	1	0
Total	22	3	3	3

2. Introduction

This deliverable reports outcomes of the third iteration cycle of Task *T2.3 Evolutionary requirements refinement*, more specifically the results of *Subtask 2.3.1 Lessons Learned collection and analysis* and *Subtask 2.3.5 Requirements re-engineering*.

The document includes the content of what was originally planned as two separate deliverables, i.e., *D2.7.3 Lessons Learned and results of usability evaluation 3* and *D2.8.3 Change request and reengineering report 3*, combined under the name of the latter. This measure was taken to reduce duplication and redundancy.

2.1 Purpose, context and scope of this deliverable

This document contains a complete list of Lessons Learned during the third iteration cycle of the ebbits project, organised per work package. These Lessons have been extracted from the joint repository in the GForge Wiki.

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 has resulted in the creation of new requirements and updating and deleting of existing requirements. Additional changes to the requirements arise from the continuous technology, regulatory standards and market watch and from verification and validation results.

The content of this document feeds into deliverable *D2.9.3 Updated requirements report 3*.

3. Research and Development Methodology

The ebbits project has adopted a human-centred, iterative development process following the guidelines of ISO 9241-210 Human-centred design for interactive systems[1]. A description of the methodology, the software engineering process, the iterative approach, the reengineering of requirements and the ebbits application of Lessons Learned can be found in deliverable *D2.7.1 Lessons Learned and results of usability evaluation 1*.

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

4. Lessons Learned in the Third Cycle

This section contains all Lessons Learned in cycle 3 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. This process results in the identification of a series of improvement opportunities and into the need for new, changed and deleted requirements. The changes in requirements are reported Section 5.

A total of 22 Lessons Learned has been reported in the third iteration cycle.

4.1 Lessons Learned in WP3

The work undertaken in WP3 involves Enterprise frameworks for life cycle management. TUK is the WP leader and no Lessons Learned have been collected and validated from this WP.

4.2 Lessons Learned in WP4

The work undertaken in WP4 relates to Semantic knowledge infrastructure. SAP is the WP leader and 2 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
SAP LL WP4-1	Application developers cannot define the mapping of devices to domain entities.	Toolset support is required that allows administrators of a local ebbits instance to map the available devices to the domain model defined by the application.	EBBITS-386
SAP LL WP4-2	All (and whole) ontologies are stored as local copies in every ebbits component regardless whether some information is useful or not for the business interest of the component owner.	This results in difficulties to distribute/update ontologies, causing potential inconsistencies and scalability issues. It also leads to wasted resources as well as inefficient ontology reasoning.	
TUK-LL-WP4-3	We had only a static information about the birth, slaughter dates of animals from farms, so we had to prepare a simulator generating such a data. In the future, real sensors will provide these data together with interfaces to the existing SAP systems of farms.	Food traceability model was implemented but no feed genetic info was provided, so the food gained mainly animal health and life info,	EBBITS-363

		additionally the info from food transport is logged.	
TUK-LL-WP4-4	Our models were linked to appropriate concepts of the IOT-A ontology	Our device ontology should be aligned with existing models to avoid future interoperability issues	EBBITS-253
TUK-LL-WP4-5	Food traceability chain uses model of services for matching of appropriate step services to gain the relevant information for food	Digital thing can be almost anything based on domain, so we had to implement the service matchmaking to get appropriate web services for different digital things	EBBITS-274

4.2.1 Analysis of Lessons Learned

- SAP LL WP4-1: During an internal conference of WP4, partners agreed on the following: Applications based on ebbits should be deployable in different environments. The applications will require, e.g., a certain type of information, but it will not know which devices will be available in the installation environment to provide this information. The application developer can only define a generic domain model, but when deployed, the domain entities contained therein need to be mapped to the devices that are available. This should be configurable at runtime by an administrator. The idea was to provide a tool that allows to do this.
- SAP LL WP4-2: During the development for the manufacturing scenario, we realized it resulted in unnecessary redundancy and performance inefficiency to host a whole copy of entire ontology models by each individual ebbits component. Besides duplicated local ontology storage leads also to problems with distribution/updates of ontology models. Therefore, when we took the ontology server to the cloud, we tried to solve this problem by storing ontologies at central places, and local components just host a thin client layer to access/query only relevant parts of ontologies.
- TUK-LL-WP4-3 During the preparation of the food traceability scenario we have learned that we have not enough data from a direct access to the farm CRMs or sensors monitoring animals. So we have prepared a simulation SW, which generates data in the exactly same form as a previously received static example from the farm. Thus we can simulate continuous data streams coming from food lifecycle.
- TUK-LL-WP4-4 IOT-A models developed within the IOT-A project were identified as a potentially expandable and partially reusable within the ebbits. As our models used similar concepts to describe devices, services, quality of services, we have decided to link our models to IOT-A models, so it can be later easier to identify which of our data is related to any possible IOT-A implementations in the same domain
- TUK-LL-WP4-5 The ebbits semantic model contains a digital think concept. Any system dealing with the real thing represented by the digital thing concept in the ebbits platform should be able to provide a service by which the data about the digital thing can be

accessed. However, it is not known in advance, where exactly and how this service is implemented in different systems. So we have modelled in our model a property of the digital thing service, where its instance clearly specifies how and where to access the digital thing services.

4.3 Lessons Learned in WP5

The work in WP5 involves Centralised and distributed intelligence. FIT is the WP leader and 5 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
FIT LL WP5-1	Defining rules with drools requires some concepts from the ontology; this is error prone w.r.t. mistyping the concept names	Defining rules for context aware applications must be supported by type safe feature	
FIT LL WP5-2	Not all ebbits developer users have experience in programming language such as JAVA.	Users should be able to define rules with simple visual language	
FIT LL WP5-3	Querying ontology every time sensor data are acquired could affect ebbits performance.	We need to pre-cache the partial knowledge from the ontology that are often used as JAVA objects.	
FIT LL WP5-4	Drools and Ontology add too much complexity for prototype developments	Experienced developers would like to work with programming language that they familiar with e.g. Java & .NET	EBBITS-403, EBBITS-397
FIT LL WP5-5	Developing prototypes include partly repetitive works	Programming work related to data acquisition and provisioning could be automated by some code generation tool	EBBITS-404

4.3.1 Analysis of Lessons Learned

Five lessons learned were identified:

- FIT LL WP5-1: During the ebbits Month 24 Demonstrator preparation, an informal interview to the developers was conducted. Most of the developers found it was difficult to define context rules using drools and without type safe features e.g., provided by tools for other programming language like Java and C# provide.
- FIT LL WP5-2: the users of ebbits from COMAU do not have computer science background and they do not have any experiences with programming language as java. They would

prefer graphical language as they are used to ladder logic language (IEC 61131). Therefore EBBITS-403, EBBITS-397 were added (See FIT-LL WP5-4)

- FIT LL WP5-3: During the ebbits Month 24 Demonstrator preparation we learned that accessing ontology is much slower than normal database, therefore some knowledge from the ontology that are used quite often should be cached in the memory.
- FIT LL WP5-4: During the ebbits Month 24 Demonstrator preparation we learned that drools adds unnecessary complexity for ebbits use cases and the experienced programmers would prefer defining rules using programming language that they are familiar with such as java. Therefore EBBITS-403, EBBITS-397 were added to build visual programming tool that generate the links between sensor stream to the domain objects. The visual model should encapsulate the complexity of accessing sensors and expose them as java objects. These requirements are also relevant to (FIT LL WP5-2)
- FIT LL WP5-5: During the ebbits Month 24 Demonstrator preparation some of the programming tasks such as connecting and provisioning sensor data could be automated using sort of template code. Thus EBBITS-404 was added to generate java code automatically from a model definition.

4.4 Lessons Learned in WP6

The work in WP6 revolves around Mainstream business systems. SAP is the WP leader and 2 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
SAP LL WP6-1	Frequent changes and additions of the data model occur during the implementation of services for an ebbits use case. Each change requires a lot of additional but often also in another sense redundant code to write if no standard data exchange formats together with a connectivity framework are used.	A connectivity framework with standard data formats like Atom or OData should be used. This allows for changes in the data model to be nearly automatically reflected in the provided service.	EBBITS-402
SAP LL WP6-2	Managers, for example a plant manager in the manufacturing scenario, or users without the technical knowledge of Business Analysts want to get insights from Business Intelligence data quickly without the need for help from an IT department.	A "point and click" solution to manipulate, organize and consolidate data and answer business intelligence questions in a visual way should be available. This could also be utilized by users without technical knowledge about	EBBITS-401

		query languages like SQL.	

4.4.1 Analysis of Lessons Learned

Two lessons learned were identified:

- LL WP6-1: During the ebbits Month 24 Demonstrator preparation, an internal evaluation with a trial version of just the ABAP (programming language of SAP) stack of the SAP AG without any connectivity framework was conducted. It turned out to be too inflexible to accommodate the frequent changes in the data model during the preparation phase. Therefore, the decision was taken to use a connectivity framework with standard data exchange. This resulted in creation of requirement EBBITS-402.
- LL WP6-2: Farmers as in the ebbits traceability scenario do not have an IT department or business analysts at their disposal. On the other hand, managers in the car manufacturing scenario want to have answers to their questions quickly, without depending on IT. All these users still want to gain insights from business intelligence data, but they need a very easy-to-use solution. This resulted in creation of requirement EBBITS-401.

4.5 Lessons Learned in WP7

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

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
CNET-LL WP7-3-1	Traceability is a property that can be ascribed to any IoT enabled physical or virtual entity (i.e., a DigitalThing in the ebbits domain ontology)	Traceability should be reflected by architectural concepts in an IoT-Architecture	EBBITS-201 EBBITS-165
CNET-LL WP7-3-2	The ebbits architecture has been mapped to (parts of) the IoT-A project generic architecture, in order to increase our understanding of what can be considered generic and specific in the ebbits architecture.	Mapping the ebbits system to an existing (in this case the IoT-A) reference architecture, allows us to analyse ebbits concepts and designs from an external perspective. It facilitates comparison with other IoT systems and communication with stakeholders. We should also consider if specific ebbits solutions could influence the IoT framework or other similar	EBBITS-406

		frameworks.	

4.6 Analysis of Lessons Learned

- CNET-LL WP7-3-1. Several of the ebbits architectural components can be said to support the notion of traceability, such as the thing manager, service orchestration components and support for global identity (based on EPC). It can be argued that traceability is not a distinguishing feature of specific IoT architectures, since there are obvious applications that are not focused on traceability, but we believe it is a sufficiently significant feature to be included in a generic architecture like the one provided by ebbits.
- CNET-LL WP7-3-2. The IoT-A proposed architecture is a fairly recent development and is yet to get a wider dissemination and application. The ebbits project should consider what specific ebbits solutions can be used to influence the IoT framework (or other related IoT architectures).

4.7 Lessons Learned in WP8

The work in WP8 takes care of Physical world sensors and networks. ISMB is the WP leader and 8 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
FIT LL WP8-3-1	Users of IT systems often fail to understand or accept security policies. If an interface provides users with hints and explanations, they are more likely to follow regulations.	Security related user interfaces should have an e-learning aspect, teaching users to properly configure the system.	EBBITS-405
FIT LL WP8-3-2	Security, which enforces delivery order, might cause unwanted effects. For example when dividing a large message into smaller packets these may arrive in a different order at the recepient side. With a security enforcing order this large message is rejected.	Security, which enforces order, has to be clearly marked and documented.	EBBITS-405
ISMB LL WP8-3-3	In order to perform first tests in the easier way, the multi-radio feature, today, is managed as a PWAL sub-component. This implies that the multi-radio package is different for any PWAL installation, because requires knowledge about the specific operations performed and about the local network. Furthermore, the network manager is unaware of the available network interfaces.	The Multi-radio package has to become a sub-component (like a plug-in) of the LinkSmart Network Manager.	EBBITS-353
ISMB LL WP8-3-4	RFID systems interfere with each other due to electromagnetic mutual interference in the transmission/reception. This lesson has been experienced while developing a component for the	In an environment with many RFID systems working simultaneously it	EBBITS-345

	traceability scenario.	is necessary to manage readings timing.	
ISMB LL WP8-3-5	Water has high conductivity; it reflects and absorbs the electromagnetic signal. UHF tags are particularly susceptible to this problem. The usage of UHF tags with organic materials (made of water) can introduce relevant problems.	In the entire traceability chain the usage of certain technologies needs to take into account the physical nature of traced objects.	none
ISMB LL WP8-3-6	In the traceability scenario, for an efficient and effective control of the "cold chain", it is necessary that relevant information (temperature, time of transport, etc.) are available and accurate throughout the entire chain. The absence/missing of certain data may generate errors and problems. E.g., in the calculation of the product remaining shelf life, if something is missing the calculation goes wrong and shelf life is inconsistent (this could lead to wrong and dangerous information for the end-user)	If relevant information is missing, this needs to be taken into account while elaborating enhanced traceability features.	none
ISMB LL WP8-3-7	Auto-generated code was used for the development of PWAL; during the progress of the development, this has resulted in a great source of duplication. From this, it was understood that relying on auto-generated code simplifies the developer's life only initially, while it introduces complexity when the project enlarges. Also, often, this practice does not allow to fall into certain levels of code quality.	Even if the auto-generated code is not necessarily a bad thing, it could be avoided in some situations or a slightly different approach could be used. A possible strategy could be reducing the part of code that needs to be automatically generated and insert in the framework the redundant code.	None
ISMB LL WP8-3-8	The initial PWAL development code documentation, even if available, was not sufficient; this has taught us that a more in-depth documentation would help, especially in very critical phases (like the last complete PWAL code refactoring).	It is a good practice to document the code; more documentation is available, the better it is.	none

4.7.1 Analysis of Lessons Learned

Regarding software development experience issues, one lesson has been identified, regarding security and security in the process.

- LL WP8-3-1, LL WP8-3-2: Generally, it is desired that security which messages arrive in the same order as they have been sent in. However, when many messages are sent in close time, switches in order over IP are a general phenomenon. To avoid frustration such behaviour of the running security has to be well documented and developers should be well informed.

Other two lessons are related with actual code developments and programming methodologies:

- LL WP8-3-7: the usage of auto-generated code, even if it partly simplifies the developer work, introduces complexity and is prone to code duplication. This could impact negatively with the code quality. The auto-generation of code should be adjusted to avoid duplication of code, maybe moving the shared components into the core capabilities of the relevant framework.
- LL WP8-3-8: this is an obvious lesson, which any developer needs actually to know. The code documentation is particularly important, since the early stages of any development. The code documentation of the PWAL needs to be additionally improved.

Also, other three process related lessons, mainly related with the traceability scenario, were identified:

- LL WP8-3-4: the lesson was raised while preparing the M36 demo. In the same environment two RFID reader were working together, interfering with each other. The only way to address this problem is to manage reading timing (in a time division multiplexing way). This lesson could be connected to the requirement EBBITS-345 about a unique ebbits timing system.
- LL WP8-3-5: the choice of UHF technology for bulk beef identification can introduce problems because the organic tissues (mainly made of water) reflect and absorb the electromagnetic signal. The usage of certain technologies cannot disregard from physical nature of objects taken into account.
- LL WP8-3-6: In order to perform an effective "cold chain" monitoring it is necessary that relevant information (temperature, time of transport, etc.) is available on time and with the correct accuracy, in order to prevent potential healthy risky problems. The absence of certain data needs to be taken into account while performing some value added services like the dynamic remaining shelf-life evaluation.

Finally, one lesson affects the architectural level:

- LL WP8-3-3: in order to perform the multi-radio features a two steps approach was considered to finally achieve the integration those capabilities in ebbits: in the first one, the early implementation, multi-radio was not inserted into the network manager because too complex, and managed as a PWAL sub-component. This, even if correctly working, implies that the multi-radio packages need to be configured for each different PWAL instance; e.g. it could be necessary to configure available network interfaces and rules on when to use them. In the second instance, the multi-radio package needs to be integrated in the overall architecture, becoming a sub-component of the LinkSmart Network Manager.

4.8 Lessons Learned in WP9

The work in WP9 involves Platform integration and deployment. CNET is the WP leader and 2 Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected
ISMB LL WP9-3-1	Auto-generated code was used for the development of PWAL; during the progress of the development, this has resulted in a great source of duplication. From this, it	Even if the auto-generated code is not necessarily a bad thing, it could	None

	was understood that relying on auto-generated code simplifies the developer's life only initially, while it introduces complexity when the project enlarges. Also, often, this practice does not allow to fall into certain levels of code quality.	be avoided in some situations or a slightly different approach could be used. A possible strategy could be reducing the part of code that needs to be automatically generated and insert in the framework the redundant code.	
ISMB LL WP9-3-2	The initial PWAL development code documentation, even if available, was not sufficient; this has taught us that a more in-depth documentation would help, especially in very critical phases (like the last complete PWAL code refactoring).	It is a good practice to document the code; more documentation is available, the better it is.	none

4.8.1 Analysis of Lessons Learned

The two lessons are related with actual code developments and programming methodologies:

- LL WP9-3-1: the usage of auto-generated code, even if it partly simplifies the developer work, introduces complexity and is prone to code duplication. This could impact negatively with the code quality. The auto-generation of code should be adjusted to avoid duplication of code, maybe moving the shared components into the core capabilities of the relevant framework.
- LL WP9-3-2: this is an obvious lesson, which any developer needs actually to know. The code documentation is particularly important, since the early stages of any development. The code documentation of the PWAL needs to be additionally improved.

4.9 Lessons Learned in WP10

The work undertaken in WP10 involves End-to-end business applications. COMAU is the WP leader and no Lessons Learned have been collected and validated from this WP.

Org. No.	Experience and knowledge gained	Lesson Learned	Requirement affected

4.9.1 Analysis of Lessons Learned

4.10 Lessons Learned in WP11

A plan for demonstration has been developed. However, at the time of writing, only the Campus Party event in London has taken place and no lessons involving requirements have been learned.

4.11 Other Work Packages

Work packages WP1, WP2 and WP12 have collected no Lessons Learned in the third cycle, largely due to their non-technical nature.

5. Requirements Engineering in the Third Cycle

A total of 22 Lessons Learned has been reported, validated and analysed in Section 4. Relative to the set of requirements listed in 'D2.9.2 Updated requirements report 2' this has resulted in 3 new requirements, 3 changed requirements and 3 deleted requirements originating from different work packages as detailed below.

Work package	Lessons Learned	New requirements	Updated requirements	Deleted requirements
WP4	5	0	0	0
WP5	5	0	0	0
WP6	2	2	0	0
WP7	2	0	0	0
WP8	8	1	2	3
WP10	0	0	1	0
Total	22	3	3	3

5.1 Change request and reengineering originating from WP3

5.1.1 Lessons Learned

No new lessons learned were identified

5.1.2 New requirements

No new requirements were identified.

5.1.3 Updated requirements

No requirements were updated.

5.1.4 Deleted requirements

No requirements were deleted.

5.2 Change request and reengineering originating from WP4

5.2.1 Lessons Learned

5 lessons learned have been reported, validated and analysed. The analysis has resulted in the following changes to requirements.

5.2.2 New requirements

Key	Summary	Rationale
EBBITS-	Toolset support allows	Application developers cannot define the mapping of

386	administrators of a local ebbits instance to map the available devices to the domain model defined by the application.	devices to domain entities.

5.2.3 Updated requirements

No requirements were updated.

5.2.4 Deleted requirements

No requirements were deleted.

5.3 Change request and reengineering originating from WP5

5.3.1 Lessons Learned

5 Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the requirements.

5.3.2 New requirements

No new requirements were identified.

5.3.3 Updated requirements

No requirements were updated.

5.3.4 Deleted requirements

No requirements were deleted.

5.4 Change request and reengineering originating from WP6

The work in WP6 revolves around Mainstream business systems. SAP is the WP leader and 2 Lessons Learned have been collected and validated from this WP.

5.4.1 New requirements

This resulted in creation of requirement EBBITS-401 and EBBITS-402.

Key	Summary	Rationale
EBBITS-401	Enable self-service approach to BI information	Managers or users without the technical knowledge of Business Analysts want to get insights for enabling business-based IoT quickly without the need for help from an IT department.
EBBITS-402	Data exchange with Enterprise Systems in standard formats like Atom or OData	For the frequent changes and additions of the data model which are necessary for the different ebbits use cases, implementing services without a connectivity framework was not flexible enough. For each change this requires to write too much own code for standard functions. A connectivity framework with standard data formats is needed where changes in the data model are nearly automatically reflected in the provided service.

5.4.2 Updated requirements

No requirements were updated.

5.4.3 Deleted requirements

No requirements were deleted.

5.5 Change request and reengineering originating from WP7

5.5.1 Lessons Learned

2 Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the requirements.

5.5.2 New requirements

No new requirements were identified.

5.5.3 Updated requirements

No requirements were updated.

5.5.4 Deleted requirements

No requirements were deleted.

5.6 Change request and reengineering originating from WP8

5.6.1 Lessons Learned

In the third period, 8 Lessons Learned have been added so reported in this deliverable, such as validated and analysed. The analysis has resulted in the following requirements addition and modification. Furthermore, in the last period has been registered the deletion of three already exiting requirements. Other requirements has been modified accordingly to the ebbits requirements process management.

5.6.2 New requirements

Key	Summary	Rationale
EBBITS-405	Security has to be well documented	Security, which enforces order, has to be clearly marked and documented. Security related user interfaces should have an e-learning aspect, teaching users proper use. Generally it is desired that security enforces that messages arrive in the same order as they have been sent in. However, when many messages are sent in close time, switches in order over IP are a general phenomenon. To avoid frustration such behavior of the running security has to be well documented and developers should be well informed.

5.6.3 Updated requirements

The requirement EBBITS-353 has been update to better describe the problem, accordingly with the lesson LL WP8-3-4; the requirement EBBITS-345 has been slightly modified thanks to another lesson (LL WP8-3-4) raised in this period, which has confirmed the problem.

Key	Summary	Rationale
EBBITS-353	Multi-radio devices should be able to detect which LinkSmart Network Manager to connect/migrate to, according to the current network interface active	Devices with multi-radio capabilities should be able to switch interface, and therefore network, without compromising the connectivity to the LinkSmart layer. The multi-radio features should be handled at Network Manager level. Furthermore, when migrating to a new interface, the device should register itself to the proper Network Manager available in that network. At network manager, level should be possible to automatically manage multi-radio features, if any. Devices accessing ebbits by using non-corporate or external networks (e.g. 3GPP) should know, somehow automatically, which border network manager must connect to.
EBBITS-345	ebbits should implement a distributed time dissemination and synchronization service	Several application in ebbits relay directly or indirectly on accurate time-stamping of data and events, thus given the distributed nature of ebbits, a time dissemination and synchronization service is required within the platform. ebbits should provide a time dissemination and time synchronization service.

5.6.4 Deleted requirements

The requirements EBBITS-400 has been moved to the "resolved" status, thanks to the recent third year developments. Results that refer to this requirement will be presented at M36 demo. The requirements EBBITS-169 and EBBITS-238 have been closed as out-of-scope as it is infeasible for ebbits to put electronic ear-tags in all animals and to enforce new procedures in the industry.

5.7 Change request and reengineering originating from WP9

5.7.1 Lessons Learned

No Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the requirements.

5.7.2 New requirements

No new requirements were identified.

5.7.3 Updated requirements

No requirements were updated.

5.7.4 Deleted requirements

No requirements were deleted.

5.8 Change request and reengineering originating from WP10

5.8.1 Lessons Learned

No Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the requirements.

5.8.2 New requirements

No new requirements were identified.

5.8.3 Updated requirements

The requirement EBBITS-166 has been added to the M36 demo as both RFID and QR barcodes are used. It has also been linked to the innovation General purpose RFID reader interface (CIC 10).

5.8.4 Deleted requirements

No requirements were deleted.

5.9 Change request and reengineering originating from WP11

No lessons or requirements has been created or changed for WP11.

6. Validation Results

6.1 Summary of verification results

No verification results have been reported by the technical work packages.

6.2 Summary of validation results

No end users have so far been involved in validation testing.

6.3 Summary of results from usability testing

No usability testing has been done on the prototypes at this stage.

6.4 Summary of outcomes of field trials

No field trials have been performed in the third development cycle.

7. Impact Assessment

7.1 Impact on overall architecture

The architecture description has been updated with concepts and components specifically designed to support traceability. A first mapping of ebbits concepts and structures to the IoT-A reference architecture was done during this period. Further IoT-A compliance should be analysed in the further developments of the architecture.

7.2 Impact on architecture for Automotive Manufacturing

Improved access to BI information is sought for.

7.3 Impact on architecture for Food Traceability

There is a need for additional initial product data in order to improve the early annotation of the products/things to be traced throughout the traceability chain. Also improved access to sensor data during the cold chain would be beneficiary, e.g., in order to more accurately calculate shelf life.

7.4 Impact on individual work packages

Apart from the requirements re-engineering reported above, no additional impact results on individual work packages have been reported .

8. References

Use the following style for references.

(EC, 2007)	European Commission (2007). A lead market initiative for Europe. Brussels. COM(2007) 860 final.
(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.
(Chen et al 2007)	Chen, Y.C., Liu, C.H., Wang, C.C., Hsieh, M.F. (2007). "RFID and IPv6-enabled Ubiquitous Medication Error and Compliance Monitoring System", 9th International Conference on e-Health Networking, Application and Services, 2007, 19-22 June 2007 Page(s):105 - 108.