Enabling the business-based
Internet of Things and Services

(FP7 257852)

D2.8.1 Change request and reengineering report 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: 4 October 2011

Document control page

Document file: D2.8.1 Change request and reengineering report 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	A. Al-Akkad (FIT)	2011-09-16	Input to 4.3 provided (new, deleted and updated requirements).
0.3	Y. Martin (SAP)	2011-09-19	Input to 4.3 provided (new, deleted and updated requirements).
0.4	M. Caceres (ISMB)	2011-09-20	Input to 4.6 provided (new, deleted and updated requirements)
0.5	M. Knechtel (SAP)	2011-09-21	Input to 4.2 from WP4 provided (new, deleted and updated requirements)
0.6	M. Ahlsén (CNET)	2011-09-21	WP7 update
0.61	P. Kool, M. Ahlsén (CNET)	2011-09-22	WP9 update
0.7	P. Brizzi, M. Caceres (ISMB)	2011-09-22	WP4 & WP8 update
0.8	A. Al-Akkad (FIT), F. Pramudianto (FIT), A. Schneider (FIT), H. Udsen (IN-JET)	2011-09-28	Sections 1, 2, 5.2 and 5.3 + editing
1.0	R. Tomasi (ISMB), M. Ahlsén (CNET)	2011-09-29	Sections 5.1 and 6
1.1	H. Udsen (IN-JET)	2011-10-04	Reviewer comments addressed, editing
			Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
R. Tomasi (ISMB), P. Cultrona (COMAU)	2011-09-29	Approved with minor revisions
P. Rosengren (CNET)	2011-10-03	Approved with comments

Index:

1. Executive Summary

This deliverable reports the first results of *Subtask 2.3.5 Requirements re-engineering* and contains a list of new and updated requirements elicited during the first iteration cycle of the ebbits project.

These new and updated requirements arise from the analysis of Lessons Learned reported in deliverable *D2.7.1 Lessons Learned and results of usability evaluation 1*, from verification and validation results and from the continuous technology, regulatory standards and market watch.

The content of the document feeds into deliverable *D2.9.1 Updated requirements report 1*.

1.1 Requirement Engineering

The requirement engineering work performed in the first iteration cycle has resulted in the creation of 81 new requirements and the modification of four existing requirements (some of them more than once). As expected, these changes to the initial set of requirements (as collated in deliverable *D2.4 Initial requirements report*) arise mainly from the efforts of the technical work packages as follows:

Work Package 4 has created 40 new requirements and revised one requirement.

Work Package 5 has created 14 new requirements and revised one requirement.

Work Package 6 has revised one requirement.

Work Package 7 has created 6 new requirements and revised one requirement.

Work Package 8 has created 18 new requirements and revised three requirements.

At this early stage of the development process it is to be expected that the requirements engineering process is dominated by discoveries resulting in new requirements, as an increasingly deeper understanding of the future outcomes of the development work is acquired.

1.2 Validation results

1.2.1 Verification activities

Overall requirements and system testing:

In the first validation cycle the overall architecture and several critical components were still undergoing major design or development activities, and it was therefore decided to limit the scope of verification to just two selected use cases, one for each domain. The scope of verification will be broadened in next iterations.

This activity resulted in the definition of two prototype architectures which have been used as test benches to drive the System Testing process.

System design and Integration testing:

Derived from overall requirements, at System Design Level partners have defined a set of system requirements, representative of the behaviour of the corresponding scenarios.

The results of Integration testing can be summarised as follows:

- All different components have been integrated and tested; under normal conditions, the components have responded according to the internal requirements, both for in-range and out-of-range values.
- During integration tests, a number of issues have been identified on some underlying LinkSmart components. Such issues have been transformed into Lesson Learned and then into specific formal requirements.

Module design and Unit testing:

The main Unit testing results have been the verification that each single module responded correctly to both in-range and out-of-range values, thus pre-validating each single component for the integration testing phase.

1.2.2 Results from usability testing:

Four prototypes have been demonstrated, two for each application domain, after six and twelve months, respectively. Usability has been assessed from a developer user's point of view.

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.

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.

No **end user validation activities** or **field trials** have taken place at this stage of the project.

1.3 Impact assessment

The efforts of this first round of requirements engineering have resulted in an improved precision of the technical requirements, as well as in the addition of a more focused set of scenario related requirements. The former has been achieved through an improved understanding by practical use of the baseline technology, the latter being attributed to the elaboration of the set of use cases during the period.

2. Introduction

This deliverable reports outcomes of Task *T2.3 Evolutionary requirements refinement* and more specifically the first results of *Subtask 2.3.5 Requirements re-engineering*.

2.1 Purpose, context and scope of this deliverable

This document contains a list of new and updated requirements elicited during the first iteration cycle of the ebbits project, listed per work package. These new and revised requirements have been extracted from the continuous technology, regulatory standards and market watch, from verification and validation results and from the improvement opportunities arising from the analysis of Lessons Learned (LLs) reported in deliverable *D2.7.1 Lessons Learned and results of usability evaluation 1*.

The content of this document feeds into deliverable *D2.9.1 Updated requirements report 1*.

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. Requirement Engineering in the First Cycle

A total of 49 Lessons Learned has been reported, validated and analysed in deliverable *D2.7.1 Lessons Learned and results of usability evaluation 1*. Relative to the initial set of requirements listed in *D2.4 Initial requirements report* this has resulted in 45 new requirements and 4 changed requirements originating from different work packages as detailed below. In addition, the analysis reported by WP4 in *D4.3 Coverage and scope definition of a semantic knowledge model* yielded 36 new requirements. No requirements have been deleted in the first iteration cycle.

At this early stage of the development process it is to be expected that the requirements engineering process is dominated by discoveries resulting in new requirements, as an increasingly deeper understanding of the future outcomes of the development work is acquired.

4.1 Change request and reengineering originating from WP4

4.1.1 Lessons Learned

In WP4, five Lessons Learned have been reported, validated and analysed. The following changes have been made to the initial set of requirements.

4.1.2 New requirements

The analysis of the Lessons Learned has resulted in the definition of four new requirements.

- **[ebbits-464] OWL-lite will be used to model ontologies.** The selection for the knowledge representation formalism in D4.2 has been made to allow high performance processing. This requirement is derived from LL WP4-1.
- **[ebbits-465] Ontology namespace.** In D4.2 it has been learned that a common namespace is important for the collaborative creation of ebbits ontologies. This requirement is derived from LL WP4-2.
- **[ebbits-466] Query the ontologies conveniently.** In D4.2 it has been learned that convenient mechanisms to query ontologies facilitate using the ebbits middleware to develop applications. This requirement is derived from LL WP4-4.
- **[ebbits-467] Only relevant parts in the ebbits ontologies.** In D4.3 and during the M12 demonstrator implementation it has been learned that we cannot model each and every aspect but only those items relevant for ebbits. This requirement is derived from LL WP4-5.

The analysis provided in deliverable *D4.3 Coverage and scope definition of a semantic knowledge model* has resulted in the definition of a further 36 new requirements, listed below.

- **[ebbits-379] All stakeholders should be annotated with unique Id, type, name and relevant info.** It is important to recognise the users interacting with the system and their privileges and restrictions. This requirement was derived from the use cases from deliverables D2.1 Scenarios for Usage of the ebbits Platform and D3.1 Enterprise Use Cases.
- **[ebbits-380] Monitored/sensed data should be contextualized (timestamp, geotag, type, etc).** It is important to know when and where data was sensed/monitored. This requirement was derived from the use cases 6.2.1.1 Monitoring and sorting data 6.3.1.1 Production optimisation.
- **[ebbits-381] Monitored/sensed data should be annotated (semantically) in local server/repo/store.** Information relationships should be available as soon as data enters the ebbits system. This requirement was derived from the use cases 6.2.1.1 Monitoring and sorting data 6.3.1.1 Production optimisation.
- **[ebbits-382] Alerts should be contextualized (timestamp, geotag, type, message, warning level, etc).** Generated messages and alerts need to be traceable and provide rich

information about the event detected. This requirement was derived from the use case 6.2.1.3 Sending alert to Mario.

- **[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).** Another added value that ebbits could introduce in enterprise domains is efficiency tracking, which requires a monitoring and log of several metrics in devices/tools/machinery and resources in general. This requirement was derived from the use cases 5.2.1.1 Initial assessment of energy consumption 5.2.1.4 Intervention on the device 5.2.4.1 Filling in a form about an accident.
- **[ebbits-384] Production should be annotated or modelled in order to calculate OEE and get number of orders/elements/products requested/delivered/in-progress/faulty.** Real-time traceability of produced goods/services is achieved by properly annotating their status and metrics during the manufacturing process. This requirement was derived from the use cases 5.2.3.1 Decision about energy reduction 5.2.4.2 Collect the production data.
- **[ebbits-385] Logistic should be annotated or modelled in order to get information about element and consumables (present, in transit from supplier, ordered, etc).** The ebbits platform could provide also a system for efficient management of consumables and logistic aspects needed in (not only) manufacturing domains. This requirement was derived from the use case 5.2.4.2 Collect the production data.
- **[ebbits-386] Feedstuff should be annotated with origin, genetics, treatment, storage conditions and transport/delivery info (batch number, silo id, amount, timestamp).** A detailed annotation of feedstuff is required to the reasoning processes devised. This requirement was derived from the use cases 8.2.1.1 Information about delivered feed 8.2.1.3 Feedstuff delivery.
- **[ebbits-387] Animals should be annotated with RFID tag, weight, genetics, birth date, and current/historical (time-stamped) data (growth/weight, location/movements, consumed feed, water, weaning, insemination, heat during pregnancy, born piglets, anomalies, vaccines).** Proper identification of animals and logging the most relevant information about their lives is vital for the traceability and quality control proposed in ebbits. This requirement was derived from the use cases 8.2.2.1 Implementation and activation of RFID tag 8.2.2.3 Monitoring of feeding and identification of deviations in behaviour of animals 8.2.2.5 Prediction of delivery to slaughterhouse 8.2.3.1 Animal transportation to slaughterhouse 6.2.3.2 Decision about insemination & 6.2.3.3 Check heat 6.2.3.5 Vaccinate 6.2.3.6 Wean piglets 6.2.3.7 Medical intervention 6.2.3.8 Selling pigs to slaughterhouse.
- **[ebbits-388] Meat packages should be annotated with ID of animal (for trace).** Meat traceability is one of the main added values of the ebbits platform in the agricultural domain. This requirement was derived from the use case 8.2.4.2 Identification of potentially infected meat packages.
- **[ebbits-389] Farm's soil/fields should be annotated with location, laboratory analysis info (date, sample field source, lab id/name, results, etc), current/historical data (types of crops, grain maturity, soil nutrients, quality of products grown, etc).** Different reasoning applications devised in ebbits for tracking the soil efficiency require a detailed annotation and logging of farms' soil. This requirement was derived from the use cases 6.2.1.1 Analyse nutrients in soil 6.2.1.2 Analysis of the history of the land use 6.2.1.5 Decision about harvesting time.
- **[ebbits-390] Farm's repository should store information about harvesting equipment, man power, etc.** The ebbits platform would provide also some functionalities for the management of resources needed for harvesting, thus they need to be included in the knowledge model. This requirement was derived from the use case 6.2.1.5 Decision about harvesting time.

- **[ebbits-391] Sow farm production should be annotated or modelled in order to allow tracking the number of piglets, pigs at fertile age, pigs ready to slaughter, maximum capacity, etc.** By proper reasoning and processing, the ebbits platform can exploit the knowledge in the network and extract aggregated information required in real time. This requirement was derived from the use case 6.2.3.1 Check conditions for production.
- **[ebbits-392] Halves of slaughtered pigs should be annotated with id, id of slaughtered pig, weight, fat thickness, date of slaughter, price, etc.** Traceability of meat requires proper tracking of pigs since birth to stores, thus the information after its slaughter is very relevant. This requirement was derived from the use case 6.2.4.1 Specification of payment to the farm.
- **[ebbits-393] Order should be annotated with type/ID of good/service, amount, price, dates(issue, expiry, delivery, etc).** Ebbits platform can be exploited also for generic enterprise processes, like account management. This requirement was derived from the use case Order goods.
- **[ebbits-394] Invoices should be annotated with supplier's info (name, id, bank account, contacts, etc), goods or services info (type/id, amount, price, dates, etc).** Ebbits platform can be exploited also for generic enterprise processes, like account management. This requirement was derived from the use case Send invoice & Record data to the accounting system.
- **[ebbits-395] Payment reports should be annotated with responsible id/name, bank, order number, status, etc.** ebbits platform can be exploited also for generic enterprise processes, like account management. This requirement was derived from the use case Send payment.
- **[ebbits-396] Generic Information should be annotated with requester, sender, content (price, capacity, dates), etc.** Information exchanged between stakeholders could be exploited for some reasoning, thus it is convenient to model it semantically. This requirement was derived from the use case Request for information. *Fit criteria:* Knowledge model for information exchange.
- **[ebbits-397] Stakeholders should be stored in local catalogues or external directories (advisory company, chamber of commerce, etc) and accessed by the different subsystems inside and outside ebbits.** In order to apply access control lists/policies, all stakeholders must be identified. This requirement was derived from the use cases from deliverables D2.1 Scenarios for Usage of the ebbits Platform and D3.1 Enterprise Use Cases.
- **[ebbits-398] Manufacturing Monitor System should have write access to local server/repo/store.** The reasoning processes devised in ebbits require access to knowledge/information found in local stores. This requirement was derived from the use case 6.2.1.1 Monitoring and sorting data. *Fit criteria:* Access rules/policy granted for MMS.
- **[ebbits-399] Manufacturing System for Analysis should have read/write access to local server/repo/store.** Analysis/reasoning is based on local monitored data, and reports are sent back to local server/repo/store. This requirement was derived from the use case 6.2.1.1 Monitoring and sorting data.
- **[ebbits-400] Reports should have a list of allowed readers/subscribers.** Aggregated data, reports, alerts, etc, should be available only to stakeholders interested in them. This requirement was derived from the use case 6.2.1.2 Checking the status of the manufacturing plant.
- **[ebbits-401] ebbits platform should have a list of alerts and subscribers.** The different monitored processes in ebbits should generate alerts and send them to the interested subsystems or stakeholders. This requirement was derived from the use case 6.2.1.3 Sending alert to Mario.

- **[ebbits-402] Manufacturing Monitor System must have read access to internal and external environment data.** The reasoning processes devised by ebbits for the manufacturing domain require environmental monitoring. This requirement was derived from the use case 6.3.1.1 Production optimisation.
- **[ebbits-403] Maintenance Manager and operators should have access to devices' and production info (proper ACL have to be implemented).** Some of the added values that ebbits will provide to managers in the manufacturing domain require a continuous tracking of the production processes, metrics and modifications introduced. This requirement was derived from the use cases 5.2.1.1 Initial assessment of energy consumption 5.2.1.5 Monitoring of the performed modification 5.2.1.6 Decision about additional improvement 5.2.3.1 Decision about the energy reduction 5.2.4.1 Filling in a form about an accident 5.2.4.2 Collect the production data.
- **[ebbits-404] Farm's Management System should have access (through secure connection) to Feed Provider Resources Monitoring System.** The food traceability scenario requires an exchange of information between all the enterprises involved in the production chain. This requirement was derived from the use cases 8.2.1.1 Information about delivered feed 8.2.1.2 Calculation production and delivery plan 8.2.1.4 Invoke information about delivered feed.
- **[ebbits-405] Feed Provider should transfer delivery information about sent feedstuff.** The traceability relays on the successful exchange of information about the monitored processes linked to the tracked product. This requirement was derived from the use case 8.2.1.3 Feedstuff delivery.
- **[ebbits-406] Farm's Local server/repo should be accessible by RFID tag readers and National servers/repos/stores.** Information about animals is stored in farm local servers and used to retrieve information when reading RFID tags or when requested by National/European authorities. This requirement was derived from the use case 8.2.2.2 Storage information about animals.
- **[ebbits-407] Farm's Monitoring System should have access to local server/repo/store.** Monitoring system keep track of several process in the farm and needs the metadata stored in local server. This requirement was derived from the use cases 8.2.2.3 Monitoring of feeding and identification of deviations in behaviour of animals 8.2.2.5 Prediction of delivery to slaughterhouse.
- **[ebbits-408] Farm's Management Application Server should access local monitoring system servers/repos/stores for generation of reports.** Management/accounting systems perform their tasks based on the information stored/provided by local monitoring systems. This requirement was derived from the use case 8.2.3.1 Animal transportation to slaughterhouse.
- **[ebbits-409] Consumer should have access to meat reports.** Relevant reports about the produced meat since the piglet born will be used by ebbits in order to enhance the information provided to consumers about the meat they are buying. This requirement was derived from the use case 8.2.4.1 Receiving information about the meat.
- **[ebbits-410] Controller should have access to (online) queries to meat packages IDs.** Quality and health control authorities can rely on ebbits in order to track the distribution of meat packages when they discover some anomaly. This requirement was derived from the use case 8.2.4.2 Identification of potentially infected meat packages.
- **[ebbits-411] Farm's Management System should have access to field info repository.** Optimisation of the resources, like fields in an agricultural domain can be achieved by analyzing and processing information logged by their respective monitoring systems. This requirement was derived from the use case 6.2.1.2 Analysis of the history of the land use.
- **[ebbits-412] Farm's Management System should have access to external information (crop price, fertilizers price, consumables price, weather, etc).**

Consumables information can be exploited through ebbits in order to manage efficiently the farm's production processes. This requirement was derived from the use cases 6.2.1.3 Decision about the type of crops 6.2.1.4 Decision about fertilization 6.2.1.5 Decision about harvesting time.

- **[ebbits-413] Sow Farm Management System should have access to production/animal repository.** The knowledge obtained by tracking all production processes in the farm will allow managers to optimise them through a single platform. This requirement was derived from the use cases 6.2.3.2 Decision about insemination 6.2.3.3 Check heat 6.2.3.5 Vaccinate 6.2.3.6 Wean piglets 6.2.3.7 Medical intervention 6.2.3.8 Selling pigs to slaughterhouse.
- **[ebbits-414] Slaughterhouse Management System and Retail Management System should have access to both production (read) and slaughter (write) repositories.** Accounting though ebbits will be simplified thanks to the knowledge acquired by multiple systems in the food production chain. This requirement was derived from the use cases 6.2.4.1 Specification of payment to the farm 6.2.5.1 Order specification.
- **[ebbits-415] Accounting Management System should store orders, and have access to Supplier Management System (to send/receive orders/acks/invoices).** The ebbits paradigm can be exploited also for improving the efficiency of accounting task. This requirement was derived from the use cases Order goods Send invoice.
- **[ebbits-416] Accounting Management System should have access to bank's management system for sending/receiving payment orders/confirms.** The ebbits paradigm can be exploited also for improving the efficiency of accounting task. This requirement was derived from the use case Send payment.
- **[ebbits-417] Manager should have access to directory of stakeholders to interact with them (send/receive info).** The ebbits paradigm can be exploited also for improving the efficiency of accounting task. This requirement was derived from the use case Request for information.

4.1.3 Updated requirements

The analysis of the LLs has resulted in the modification of (new) requirement [ebbits-383].

- **[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).** This requirement has been updated because of LL WP4-3. The fit criterion "There is a unique mapping between physical devices and their semantic reflections" has been added.

4.1.4 Deleted requirements

No requirements have been deleted.

4.2 Change request and reengineering originating from WP5

4.2.1 Lessons Learned

In WP5, 11 Lessons Learned have been reported, validated and analysed. The following changes have been made to the initial set of requirements.

4.2.2 New requirements

The analysis of the LLs has resulted in the definition of fourteen new requirements:

- **[ebbits-131] The system should support soft logic algorithms.** The title has been changed (previous title "Support fuzzy or probability concepts for reasoning."). As the fit criterion has been changed, stating that the system needs to support at least two different soft-logic algorithms, the status had to be changed from *Implemented* to *Open*. This requirement has been updated in the scope of LL WP5-3.
- **[ebbits-134] The context model needs to be extensible during runtime.** The title of this requirement has changed (previous title "Ability to self-adaptation"). This requirement has been updated in the scope of LL WP5-11.
- **[ebbits-135] Dynamically loaded libraries must undergo a security check before their usage.** This requirement has now a more focused title (previous title "Protection of System integrity"). Dynamically loaded libraries (e.g. DLL or JAR files, or OSGI bundles) may contain malicious code and thus harm business systems. This requirement has been updated in the scope of LL WP5-6.
- **[ebbits-139] Support runtime reconfiguration.** After the update of [ebbits-134] this requirement does now depend on it. This requirement has been updated in the scope of LL WP5-11.
- **[ebbits-140] The middleware should monitor device's resource usage.** The title of the requirement is now more adequate (previous title "Transparency of device performance"). This requirement has been updated in the scope of LL WP5-4.
- **[ebbits-154] Aggregate data from various databases and sources.** The status of this requirement has been set to duplicate, as requirement [ebbits-43] "Aggregating collected sensor data at a central point". This requirement has been updated in the scope of LL WP5-4.
- **[ebbits-450] The system should compensate deviations of incoming data.** Collected sensor data may contain outliers (e.g. spikes), which should not influence the measurement. This new requirement derived from LL WP5-3.
- **[ebbits-451] Sensor fusion algorithm must be added during run-time in a modular and extensible way.** Sensor fusion algorithms vary greatly and can't be generalized only in one module. This new requirement derived from LL WP5-1.
- **[ebbits-452] Sensor fusion algorithms must be realized as a decoupled component.** Sensor fusion algorithms must be reusable by several other components. This new requirement derived from LL WP5-2
- **[ebbits-453] The system must be able to assign fused data as a context attribute of an entity.** A collection of sensors is required to provide contextual values, as entities cannot provide their own contextual values. For instance, temperature sensors inside a room provide contexts in terms of the climate of the room (entity). This new requirement derived from LL WP5-5.
- **[ebbits-454] The system must monitor the state of devices and entities.** Continuous monitoring of entities is needed to detect anomalies (e.g.: ill pigs, or overheated welding guns). This new requirement derived from LL WP-9
- **[ebbits-455] System needs to trigger business events based on changes of devices and entities states.** Enterprise applications have to be notified when the process starts and finishes, and further how much resources have been consumed for the process. This new requirement derived from LL WP-9
- **[ebbits-456] The system should be able to process a large number of sensor events.** During the user workshop at COMAU it was elicited that at 500 sensor values per seconds must be handled. This new requirement derived from LL WP-10.
- **[ebbits-458] Libraries must only be accessible only for permitted applications.** Libraries can contain functionality that should not be available to all kinds of applications. For instance, the calculation of quality rating of meat should only be allowed for

slaughterhouse applications, though not for consumer applications. This new requirement derived from LL WP5-7.

4.2.3 Updated requirements

The analysis of the LLs has resulted in the modification of one existing requirement:

- **[ebbits-64] Historical data should be recorded persistently.** The name has been changed to a more proper requirement (previous name: "Logging of Quality related information of each Manufacturing Part") and WP4 is now responsible as dealing with the persistence of events is beyond the scope of WP5. This requirement has been updated in the scope of LL WP5-8.

4.2.4 Deleted requirements

No requirements have been deleted.

4.3 Change request and reengineering originating from WP6

4.3.1 Lessons Learned

In WP6, two Lessons Learned have been reported, validated and analysed. The following changes have been made to the initial set of requirements.

4.3.2 New requirements

No requirements have been added.

4.3.3 Updated requirements

The analysis of the Lessons Learned has resulted in the modification of one existing requirement:

- **[ebbits-99] Mobile access to farm data in the ERP system.** This requirement has now a more focused title, previous title was "Mobile management of farms". Additionally, it was specified that this access should be browser-based in order to enable the maximum number of mobile platforms.

4.3.4 Deleted requirements

No requirements have been deleted.

4.4 Change request and reengineering originating from WP7

4.4.1 Lessons Learned

In WP7, seven Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the initial set of requirements.

4.4.2 New requirements

- **[ebbits-217] It must be possible to order events in the actual event sequence.** The delivery of events received from different sources might not follow in the original sequence at transmission. There might be communication delays etc that make them arrive in the wrong order. Nevertheless rules should be able to express temporal/sequence dependencies on events which reflect the actual temporal event sequence at the sources.
- **[ebbits-218] An event history should be maintained.** Rule definitions can refer to past events, and behaviour can be defined based on that event history.

- **[ebbits-220] Event model.** All events should be described by a set of metadata elements, and related in an event taxonomy. This taxonomy should be based on a common ebbits vocabulary (possibly derived from a higher level business vocabulary. The taxonomy should be encoded in a semantic event model.
- **[ebbits-461] Dependencies on past events possible.** An action executed by the system may be dependent on more than one event, and some of them could have occurred in the past. The architecture needs to support a persistent storage on some of the event-consuming nodes.
- **[ebbits-462] Scalable event processing.** The platform must be able to handle a large number/high frequency of parallel event streams. Need for lower layer filtering and fusion (WP5), as well as higher level (WP7) aggregation and semantic enrichment of events.
- **[ebbits-463] Semantic event processing.** It must be possible to interpret events in the context of the different layers in the architecture (from PWAL to a business rules layer).

4.4.3 Updated requirements

The analysis of the LLs has resulted in the modification of (new) requirement [ebbits-220].

- **[ebbits-220] Event model based on common vocabulary.** This requirement has been updated to more precisely express the need for an event ontology to be based on a vocabulary related to the overall ebbits data and process models. The name (Summary) was changed accordingly.

4.4.4 Deleted requirements

No requirements were deleted.

4.5 Change request and reengineering originating from WP8

4.5.1 Lessons Learned

In WP8, seventeen Lessons Learned have been reported, validated and analysed. The analysis has resulted in the following changes to the initial set of requirements.

4.5.2 New requirements

The analysis of the LLs in WP8 has led to the definition of 18 new requirements:

- **[ebbits-468] LinkSmart should support automatic builds.** This requirement was derived from the Lessons Learned LL WP8-1 and LL WP8-11. Note: This requirement is more relevant for WP9 Platform integration and deployment,
- **[ebbits-469] Local LinkSmart instances should properly handle local variables when remote environments are restarted.** LinkSmart should be more robust to different network issues, like temporary offline periods or restarting of remote instances. This requirement was derived from the Lessons Learned LL WP8-1 and LL WP8-11.
- **[ebbits-470] PWAL should support reconfigurable dynamic polling policies.** Applications could have different polling needs, which eventually could change in runtime, so the PWAL must offer an easy reconfiguration of the polling policies per parameter and per application. This requirement was derived from the Lessons Learned LL WP8-3 and LL WP8-4.
- **[ebbits-471] 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. This requirement was derived from the Lesson Learned LL WP8-5.

- **[ebbits-472] PWAL should support accurate time-stamping of data acquainted.** The PWAL should be able to properly handle time information of the data and events it access/generate. This handling must include the synchronization to the ebbits time dissemination service and compensation of hardware and communication delays if possible. This requirement was derived from the Lesson Learned LL WP8-5.
- **[ebbits-473] PWAL should expose suitable methods in order to enrich raw data.** Data acquainted through the PWAL needs to be enriched with meta-data (like source, geotag, timestamp, units, etc), which will be then used by upper layers and applications. Being the PWAL the lowest link between ebbits platform and the physical world, part of this meta-information could be already attached at this level, easing the processing of it by the multi-sensor fusion and context awareness services. This requirement was derived from the Lessons Learned LL WP8-6 and LL WP8-7.
- **[ebbits-474] PWAL should be able to match PLC symbols with ebbits ontologies.** The definition of symbols on an OPC server (i.e., variables of interest inside the PLC) could be made in accordance with the PLC programmer according to an agreed convention that could be exploited for an automatic matching with a device catalogue or ontology. This requirement was derived from the Lessons Learned LL WP8-6 and LL WP8-7.
- **[ebbits-475] PWAL should adopt a lock and semaphore-based policy to the access of PLC memory.** Since different applications could eventually be interested in a common variable, the PWAL must assure its access is controlled in order to avoid collisions in concurrent requests, as well as possible locks or restrictions to specific applications. This requirement was derived from the Lessons Learned LL WP8-8 and LL WP8-10.
- **[ebbits-476] PWAL should implement an error control strategy to assert correct data type and values written to the PLC.** Errors in writing variables to the PLC must be avoided at all cost, since they can lead to a halt in the running program. The PWAL has to adopt a suitable error control strategy in order to assert data has been introduced correctly (this eventually would require a control logic in the PLC program as well). This requirement was derived from the Lessons Learned LL WP8-8 and LL WP8-10.
- **[ebbits-477] PWAL should implement a heterogeneous multi-data aggregation in single events.** Event-driven data acquisition can easily generate scalability issues if single events are generated per sample. Thus aggregation of several samples in a single or few events has to be devised. This requirement was derived from the Lesson Learned LL WP8-9.
- **[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.** Some variables gathered through the PWAL could require high sampling frequencies and maybe just some feature of the acquired signal is of interest, so in order to save some bandwidth and avoid scalability issues, the PWAL could offer some basic feature extraction and sensor fusion capabilities. This requirement was derived from the Lesson Learned LL WP8-9.
- **[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.** Devices with multi-radio capabilities should be able to switch interface, and therefore network, without compromising the connectivity to the LinkSmart layer, this means that when migrating to a new interface, the device should register itself to the closest Network Manager available in that network. This requirement was derived from the Lessons Learned LL WP8-12 and LL WP8-17.
- **[ebbits-480] Multi-radio devices should avoid re-generation of HIDs when migrating to a different LinkSmart Network Manager.** Again, devices with multi-radio capabilities should be able to migrate to different networks without affecting the functionalities at the LinkSmart layer, including identifiability, thus when registering to a new Network Manager, the device should try to register itself with the previous HID, triggering a deregistration request to the previous Network Manager. This requirement was derived from the Lessons Learned LL WP8-12 and LL WP8-17.

- **[ebbits-481] Multi-radio devices should be use local data caching and delay tolerance networking.** Devices with multiple radio interfaces and in general devices with delay tolerance networking capabilities may experience periods with no networking, where some events may have happen, thus a local caching of its data may prevent any loss of data. This requirement was derived from the Lessons Learned LL WP8-13 and LL WP8-16.
- **[ebbits-482] Multi-radio devices with local data caching should implement suitable application specific data-expiration policies in order to prevent cache overflows.** Devices with local data caching for delay tolerance networking, may exhibit cache overflows if they generate big amounts of information and/or experience large offline periods, thus data-expiration policies need to be applied in order to prevent this. This requirement was derived from the Lessons Learned LL WP8-13 and LL WP8-16.
- **[ebbits-483] Multi-radio devices should be able to gather information about their network interfaces needed for the selection policies.** Multi-radio devices must be able to collect and expose some information about their interfaces, must be known like throughput, energy consumption, cost of traffic, quality of service, between others. Such information will be useful for defining interface selection policies. This requirement was derived from the Lesson Learned LL WP8-14.
- **[ebbits-484] Multi-radio devices should select the most proper network interface according to the application requirements.** Depending on the properties of the information (e.g., importance, quality of service, timeout, etc), multi-radio devices should select the network interface most suitable to the requirements of the application accessing it, which could be a energy or cost saving policy for instance, or an urgent event that should be transmitted at all costs. This requirement was derived from the Lesson Learned LL WP8-14.
- **[ebbits-485] 6LoWPAN networks should include frequency agility features in order to enhance the overall system reliability.** The ability to jump to a different channel automatically according to the channel occupancy or interference seems a promising solution in order to cope with the high electromagnetic pollution present in manufacturing scenarios, thus a frequency agility service should be included in 6LoWPAN networks. This requirement was derived from the Lesson Learned LL WP8-15.

4.5.3 Updated requirements

After the analysis of the LLs in WP8, 3 existing requirements have been updated (text in italics correspond to the extended/modified description):

- **[ebbits-51] The network infrastructure needs to have self-configuration capabilities.** This requirement has been updated in the scope of Lessons Learned LL WP8-12 and LL WP8-17.
    - **Rationale:** Due to the huge amount of heterogeneous devices that can be connected to one network, this network needs to support the deployment through some sort of self-configuration. *For the same reason, the platform should provide to new devices instruments to automatically connect and auto-configure the most proper network interface, also respecting the aimed value-added solution.*
- **[ebbits-53] New products should be networked with mainstream enterprise systems easily and cost-efficiently.** This requirement has been updated in the scope of Lessons Learned LL WP8-12 and LL WP8-17.
    - **Rationale:** New devices should be integrated into existing systems easily and cost-effectively, in order to support higher value-added, interoperable solutions. *Devices could use different network managers depending on the available network interface and on service being implemented.*

- **Fit Criteria:** A new devices can be connected to an existing enterprise system within one day by one person. *Devices accessing ebbits automatically should be auto-configured and should be able to detect which border network manager use.*

- **[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).** This requirement has been updated in the scope of Lessons Learned LL WP8-6 and LL WP8-7.
  - **Rationale:** *Ebbits enables value added service on the top of available information in relation with context awareness.*
  - **Details:** Another added value that ebbits could introduce in enterprise domains is efficiency tracking, which requires a monitoring and log of several metrics in devices/tools/machinery and resources in general. *By using suitable symbols at PWAL level these information can be identified and related with industrial processes.*

4.5.4 Deleted requirements

No obsolete requirements have been identified after the analysis of the LLs in WP8.

5. Validation Results

5.1 Summary of verification results

5.1.1 Overall requirements and system testing

In the first validation cycle the overall architecture and several critical components were still undergoing major design or development activities, and it was therefore decided to limit the scope of verification to just two selected use cases, one for each domain. The scope of verification will be broadened in next iterations.

For reference, in the following, a subset of requirements among those selected to drive the verification phase are reported:

Automotive Manufacturing:

- [ebbits-23] The ebbits platform should facilitate the integration of new physical devices into existing enterprise systems
- [ebbits-24] A product's lifecycle history can be traced within less than 24 hours
- [ebbits-36] Controlling of machines/stations in manufacturing plant remotely
- [ebbits-38] Flexible Integration of HW/SW components
- [ebbits-39] Retrieve manufacturing data history of any relevant event during production
- [ebbits-40] Lifecycle of a robot and its components is traceable
- [ebbits-42] Semantic relationships between data
- [ebbits-49] Access to energy-related information from production machines needs to be provided.
- [ebbits-50] Filtering to obtain relevant Information
- [ebbits-52] Interoperability needs to be created between various subsystems in the manufacturing area
- [ebbits-61] Display plant activities in real-time
- [ebbits-63] Diagnostic component to detect and correct malfunctions
- [ebbits-64] Historical data should be recorded persistently
- [ebbits-66] Correlate problems found with production batches
- [ebbits-84] Interfacing with Programmable Logic Controller of production robots
- [ebbits-85] 3 Measurement Points for every station in body welding
- [ebbits-91] Filter/fusion information for each operational process
- [ebbits-92] Early maintenance notification when needed
- [ebbits-93] Bring data from fieldbus network to ethernet network
- [ebbits-109] Recognition of energy wasting behaviours
- [ebbits-121] Items need to be traced within an enterprise
- [ebbits-158] Alarms are send when specific situations occur
- [ebbits-380] Monitored/sensed data should be contextualized (timestamp, geotag, type, etc).

Food Traceability:

- [ebbits-34] Each sow carries an electronic unique ID
- [ebbits-45] System can feed the farms data to research
- [ebbits-46] Affordable tagging/tracking system for pigs
- [ebbits-56] Farmers need to save and able to reflect breeding history information
- [ebbits-149] Retrieve the behaviour on an individual animal level
- [ebbits-152] The system should allow the correlation of information emerging from several sources
- [ebbits-167] Save historical information in farms

This activity resulted in the definition of two prototype architectures which have been used as test benches to drive the System Testing process. Figure 1 and Figure 2 show the architectures used for System Testing.

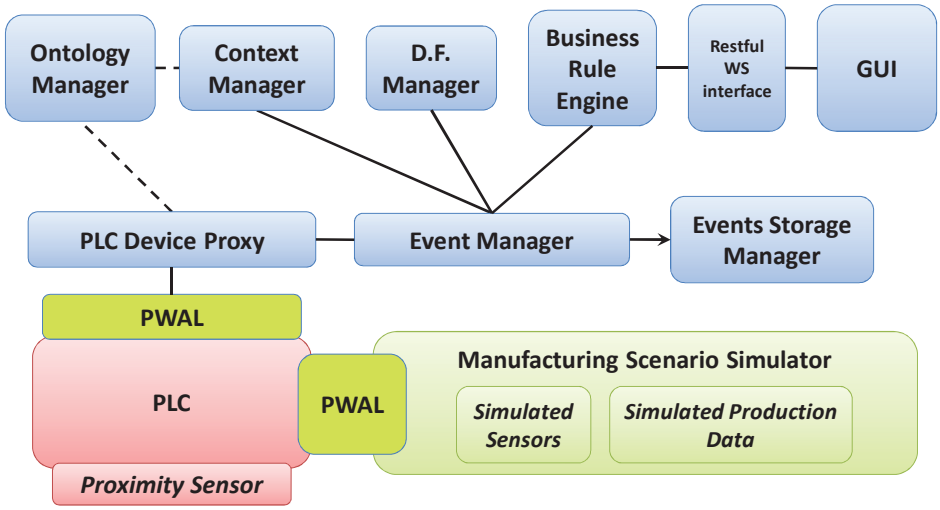

Figure 1 – System architecture for Automotive Manufacturing

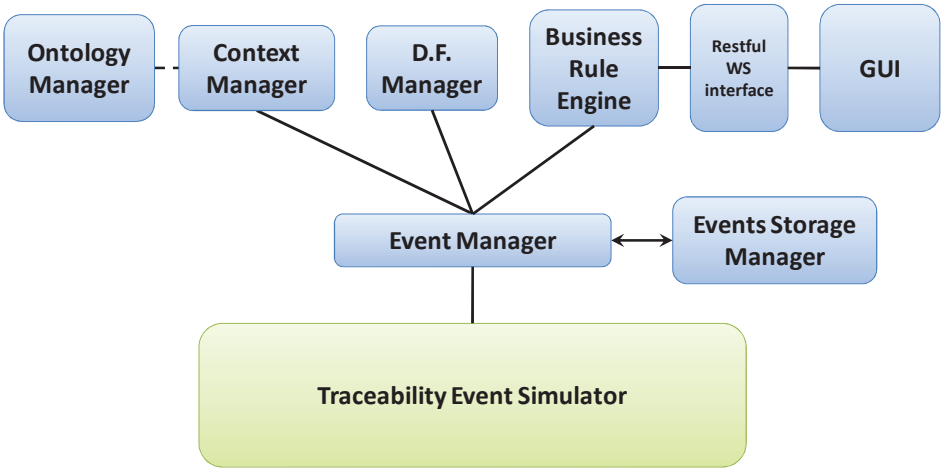

Figure 2 – System architecture for Food Traceability

To support System testing, simulators have been introduced to emulate unavailable elements of the real scenario.

5.1.2 System design and Integration testing

Derived from overall requirements, at System Design Level partners have defined a set of system requirements, representative of the behaviour of the corresponding scenarios. Their storylines included the definition of the behaviour of each component under different conditions and a correspondent reference thread of events generated by different components.

Based on the storylines the expected behaviour of each component has been defined, in both in-range and out-of-range conditions, hence driving an incremental process of integration testing, where separate components are integrated in couples. Integration sessions have been driven both remotely (via internet) and during physical integration sessions.

The results of Integration testing can be summarised as follows:

- All the different components have been integrated and tested. Under normal conditions the components have responded according to the internal requirements, both for in-range and out-of-range values.
- During integration tests, a number of issues have been identified on some underlying LinkSmart components. Such issues have been transformed into Lesson Learned and then into specific formal requirements.

5.1.3 Module design and Unit testing

Since the different modules in the aforementioned architecture were developed by different partners, different simple unit-test simulators have been applied by each partner to emulate the "outside world" while focusing on specific components, e.g. by providing a fake event generator.

This low-level verification phase has been employed mostly to support module development and to test single modules under different, variable and controlled conditions.

Controlled conditions used within the unit testing phases have been derived from the scenarios and have been tracked by partners using shared tables.

The main Unit testing results have been the verification that each single module responded correctly to both in-range and out-of-range values, thus pre-validating each single component for the integration testing phase.

5.1.4 Summary of evaluated requirements

This subsection contains a summary of evaluated requirements, exemplified by results from WP5 and WP8.

In the first validation cycle 40 requirements have been assessed in WP5:

- Two requirements have been implemented and meet the fit criterion, namely [ebbits-43] and [ebbits-78]
- Seven requirements are partly implemented being at a work-in-progress stage, viz. [ebbits-36], [ebbits-75], [ebbits-79], [ebbits-81], [ebbits-91], [ebbits-92] and [ebbits-141], and
- 31 requirements are open meaning the relevant implementation has not started yet.

The results are depicted in the pie chart in Figure 3.

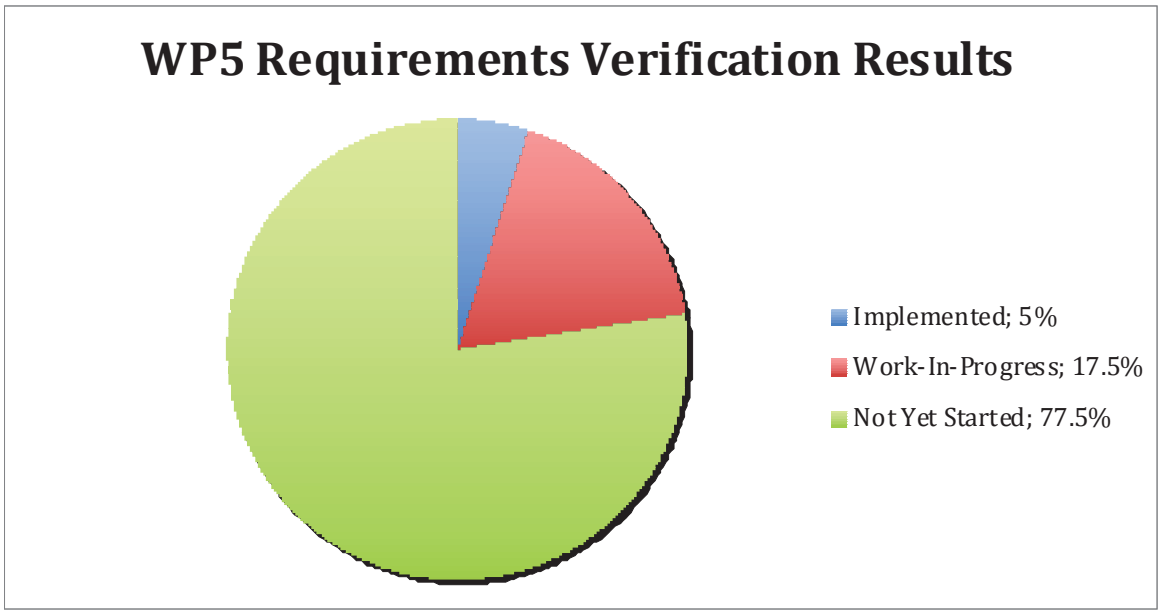

Figure 3 – WP5 Verification results

In the first validation cycle 21 requirements have been assessed in WP8:

- 4 requirements have been partly implemented being at a work-in-progress level, namely:
- [ebbits-84] Interfacing with Programmable Logic Controller of production robots
- [ebbits-34] Each sow carry an electronic unique ID
- [ebbits-46] Affordable tagging/tracking system for pigs
- [ebbits-149] Retrieve the behaviour on an individual animal level

17 requirements are still open, i.e. relevant implementation has not started yet.

The results are depicted in the pie chart in Figure 4.

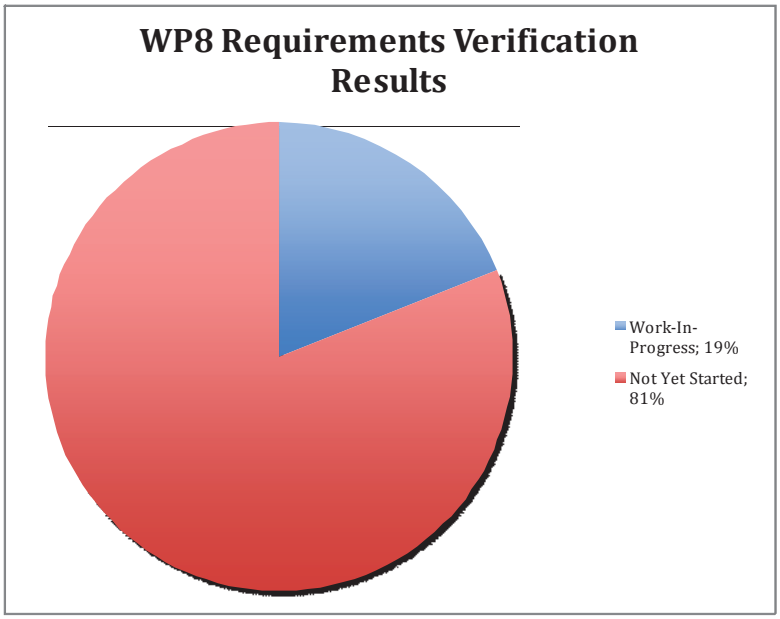

Figure 4 – WP8 Verification results

5.2 Summary of validation results

No end user validation activities have been performed at this stage of the project.

5.3 Summary of results from usability testing

Four prototypes have been demonstrated, two for each application domain, after six and twelve months, respectively. Usability has been assessed from a developer user's point of view. For a more detailed description, see *D2.7.1 Lessons Learned and results of usability evaluation 1*.

5.3.1 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 indicators (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.

5.3.2 Month 12 demos

For the M12 Automotive Manufacturing demo the initial architecture of the ebbits platform was validated by having a typical application developed by several developers, some of which developed 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.

5.4 Summary of outcomes of field trials

No field trials have taken place at this stage of the project.

6. Impact Assessment

6.1 Impact on overall architecture

At the end of the first iteration, the first version of the overall ebbits architecture is being finalised. This architecture reflects the ebbits platform as implemented by the proof-of-concept prototype. In parallel with the platform design, two demonstrators were implemented, for the Food Traceability and the Automotive Manufacturing scenario respectively. The development was based on an initial set of requirements as a complement to the baseline requirements as provided by the DoW. The ebbits requirements database can in principle be divided into two subsets, requirements directly emanating from the two scenarios (and corresponding use cases), and those related to the ebbits platform and its generic architecture. The efforts of this first round of requirements engineering have resulted in an improved precision of the technical requirements, as well as in the addition of a more focused set of scenario related requirements. The former has been achieved through an improved understanding by practical use of the baseline technology, the latter being attributed to the elaboration of the set of use cases during the period. For this iteration, the following aspects addressed by new and refined requirements can be highlighted:

- The need for common vocabularies as a basis for business process definition and horizontal process interoperability. In this iteration this is specifically highlighted for the Food Traceability scenario, i.e., requiring (inter)operability with external business systems related to the farming and food production processes. A common ebbits vocabulary may also influence technical solutions for the event processing, data management and service orchestration on several layers in the architecture, e.g. by facilitating semantic resolution.
- Semantic descriptions and the provision of ontology based models are promoted for knowledge representation across the ebbits platform. The approach is primarily to be based on the RDF/OWL syntaxes and the related tools for querying and inferencing. From an architectural point of view, there are trade-offs to be made here, balancing expressive power, scaling and the complexity of models.
- Annotation and contextualisation of arbitrary ebbits objects are put forward as a basis for supporting traceability. This appears to be equally important to both scenarios, i.e. not only Food Traceability. Continuous and fine-grained semantic annotations of objects (including sensor values, devices, events, data and even pigs) will impact the performance of the platform and consequently the architecture. Traceability also impacts the architecture in that it requires an identify-and-name scheme with components for identity resolution across network boundaries and on multiple layers in the software architecture.
- Event processing and sensor fusion, especially with respect to scalability, is vital to the ebbits platform performance. Several requirements have been rephrased or formulated to address this issue. The technical solutions to meet these requirements have an impact on the overall architecture of ebbits in that they affect the device integration and network layers (PWAL) as well as the upper layers dealing with business events.
- Rule engines and service orchestration. The use of "rule engines" is put forward by several work packages, as a means to addressing large data sets and high frequency event streams, on several layers in the architecture (i.e. from the PWAL to upper business processes). A number of requirements suggest the use of rules, and suggestions for use of standard rule systems have been made. A common approach to rules processing should be investigated and provided as input to the design of the architecture.

At this first phase of the project, Lessons Learned and other observations made are general in nature; no significant discoveries have been made that impact specifically on the architectures of the Automotive Manufacturing or Food Traceability scenarios or on the individual work packages.