Enabling the business-based
Internet of Things and Services

(FP7 257852)

# D10.1 Description of communication networks and common components

**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: 1.0 Submission date: May 2011

# Document control page

**Document file:** D10.1_Description of communication networks and common components_v1.0
**Document version:** 1.0
**Document owner:** ISMB

**Work package:** WP10 – End-to-end Business Applications
**Task**: T10.1 Networks and common components
**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 | Gonzalo Alcaraz, Claudio Pastrone, Maurizio Spirito (ISMB) | 2011-01-19 | Initial structure, Time plan, Tasks distribution |
| 0.15 | Matts Ahlsén (CNET) | 2011-02-03 | Comments on initial structure |
| 0.2 | Gonzalo Alcaraz, Claudio Pastrone (ISMB) | 2011-03-07 | Updated structure based on comments from partners. |
| 0.3 | Thomas Nejsum Madsen, Michael Jacobsen (TNM) | 2011-03-25 | Chapter 3 and 5contribution |
| 0.4 | Pietro Cultrona, Roberto Checcozzo (COMAU) | 2011-04-05 | Chapter 3 and 5 contributions |
| 0.5 | Thomas Janke (SAP) | 2011-04-08 | Section 3.4 contribution |
| 0.6 | Gonzalo Alcaraz, Claudio Pastrone, Maurizio Spirito (ISMB) | 2011-04-10 | Chapter and section updates. Section 4.1, 6.1, 7.1 |
| 0.7 | Marek Paralič (IS) | 2011-04-11 | Section 4.1 contribution |
| 0.8 | Thomas Janke (SAP) | | Section 3.4 Update |
| 0.9 | Peeter Kool, Matts Ahlsén (CNET) | 2011-05-03 | ebbits platform architecture, Integration Plan & Test Plan |
| 0.97 | Gonzalo Alcaraz, Claudio Pastrone (ISMB) | 2011-05-09 | Chapter and section updates |
| 0.98 | Gonzalo Alcaraz (ISMB) | 2011-05-13 | Device classification update |
| 1.0 | Gonzalo Alcaraz, Claudio Pastrone, Maurizio Spirito (ISMB) | 2011-05-31 | Final version submitted to the European Commission |

**Internal review history:**

| Reviewed by | Date | Summary of comments |
|---|---|---|
| Alexander Schneider | 2011-05-11 | Approved with comments |
| Jesper Thestrup | 2011-05-18 | Approved with comments |

# Table of Contents

# 1. Executive summary

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

For this aim, WP10 deals with the development of the two end-to-end business applications in charge of evaluating and demonstrating the features of the ebbits platform in real-world settings. Within the WP10, the main tasks will focus on developing and deploying end-to-end business applications in both ebbits scenarios; task T10.2 for manufacturing scenario and task T10.3 for the traceability scenario. Moreover, task T10.1 provides the network connectivity to components that will be used during the field trials at user sites and perform the technical testing of these functionalities as well. Within the ebbits project, four field trials have been defined at M10, M22, M34 and M46.

The main objective of this deliverable is to describe how the network infrastructure to be used during the field trials is composed. This deliverable starts with a description of all components that will be involved in the field trials, including networking devices, core components of the ebbits platform and physical world objects, being classified as powerful devices, constrained devices and sub-systems. Moreover, a brief introduction to relevant communication protocols (i.e., Profinet and 6LoWPAN) is included as a complement to the well-known TCP/IP protocol suite. This latter is the solution that will be mainly used in all the prototypes to be developed.

Descriptions of each of the field trials for manufacturing and traceability scenarios are included with information regarding relevant components and network infrastructure used within each prototype. Then, the integration and testing plans for both the network infrastructure and the End-to-end Business Applications are presented. It is worth mentioning that the descriptions related to prototypes due in M34 and M46 are less detailed compared to the ones provided for M10 and M22 prototypes. The results and the lessons learnt from the first two prototypes will be then used along the project to better describe M34 and M46 scenarios.

As far as the network infrastructure is concerned, the integration plan adopts an incremental approach and describes the steps or stages on the way to complete the integration of all network components. The testing plan instead includes the relevant tests such as network discovery, connectivity testing and throughput measurement: the final aim is to assure that the network infrastructure operates properly. Concerning the End-to-end Business Applications, the integration plan describes the necessary steps to be taken into account when building an ebbits application while the testing plan adopts an iterative and incremental approach for development The use of automatic testing is considered whenever possible.

In conclusion, communication networks and the related components are an essential part of the field trials. The heterogeneity of devices that will be integrated into ebbits shows a challenging environment regarding network reliability and architecture. For that reason, the network should be integrated and tested in an optimal way in order to efficiently provide IP connectivity to all the devices considered for the ebbits scenarios. In addition, both integration and testing plans adopt an incremental approach. In fact, iterative process used in the ebbits project and lessons learned from the first prototypes would be of help to introduce any relevant modification from the initial planning into the different steps related to both integration and testing processes.

# 2. Introduction

The ebbits project main goal is to research and integrate *Internet of Things* (IoT) technologies into the business domain. Business applications developed within ebbits will be able to incorporate physical objects, services and people into mainstream enterprise systems supporting real world as well as online end-to-end interoperability.

The challenging environment containing a massive amount of heterogeneous devices and business applications from the IoT will be managed by using semantic technology that allows automatic processing of information and autonomous collaboration among devices.

ebbits will open new possibilities to offer a wide range of novel business services based on the orchestration of physical devices, software services, and people. The latter concept is introduced in the ebbits project as the *Internet of People, Things, and Services* (IoPTS).

## 2.1 Purpose, context and scope of this deliverable

The main purpose of this deliverable is to describe the settings and configuration of the communication network components within ebbits, in order to provide them with network connectivity during the field trials and at user sites.

The work is performed within the context of task T10.1 included in WP10 "End-to-end Business Applications". WP10 main objective is to deal with the development of the two end-to-end business applications that will be used to evaluate and demonstrate the features of the ebbits platform in real-world settings, including population of user domain data. Concerning the relation to other WPs, WP10 receives inputs from WP9 regarding the ebbits platform and requirements from WP2. In addition, WP10 provides to WP2 the validation results in order to refine the initial requirements.

Regarding task T10.1, the main goal to be achieved is the integration and test of the network infrastructure and communication protocols to be used during the field trials.

## 2.2 Deliverable Organization

This deliverable is organized in the following chapters:

- Chapter 3 describes the systems components that will be used during the field trials.
- Chapter 4 describes the networks components in charge of providing network connectivity to the components described in chapter 3.
- Chapter 5 includes a description of the different field trials to be deployed during the ebbits project.
- Chapter 6 includes the description of the integration methodologies and approaches to be used during the field trials.
- Chapter 7 includes the description of the testing methodologies and approaches to be used during the field trials.

# 3. System components

The system components within ebbits relate basically to all systems and sub-systems which will be present in the ebbits platform. Considering the heterogeneity of devices, a device classification has been defined in order to group devices with similar characteristics in terms of making easier the overall integration process. The classification is as follows:

- **Powerful devices:** have significant amount of resources in terms of energy, computational power, memory and communication capabilities.
- **Constrained devices**: are mainly characterized by limitation concerning energy, computational and memory resources**.**
- **Sub-systems:** are a group of independent but interconnected elements usually controlled by a specific device which exposes the sub-system functionalities as a whole.

The chapter starts with the description of the ebbits platform architecture with the different subsets involved. Then, all components included in manufacturing and traceability scenarios and the enterprise systems components, such as ERP applications, are described as well.

## 3.1 ebbits Platform Architecture

The ebbits platform is the central production environment for the deployment of ebbits applications. It consists of five subsets, each responsible for its own part of the overall functionality.

- The Data Management subset is central to the high level functioning of applications and services deployed on the platform. It implements a model-driven architecture for application development and deployment, an open service oriented architecture for core service functionalities, data manipulation, data fusion and event handling. It also manages data transfer to and from nodes and stakeholders in the ebbits environment.
- The Service Orchestration subset will orchestrate the different services available in a pre-described sequence for execution. This component introduces higher abstraction mechanisms and makes the application developer independent of using a specific programming environment to coordinate ebbits applications.
- The Network Management subset is responsible for the physical communication between devices, people and external repositories. Each node will have its own Network Manager and each Network Manager will have an external Web Service based interface where it can exchange data with remote Network Managers. In addition, the Network Manager will support the adoption of opportunistic communication approaches e.g. multiradio connectivity and store and forward communication.
- The Security Management subset will perform mapping and brokering between security models, user and device profiling management, mapping and usability between trust domains, and semantic standards with generalisation ontologies development.
- The Application Development subset is an open SDK toolkit for model-driven development of applications that use the ebbits platform.

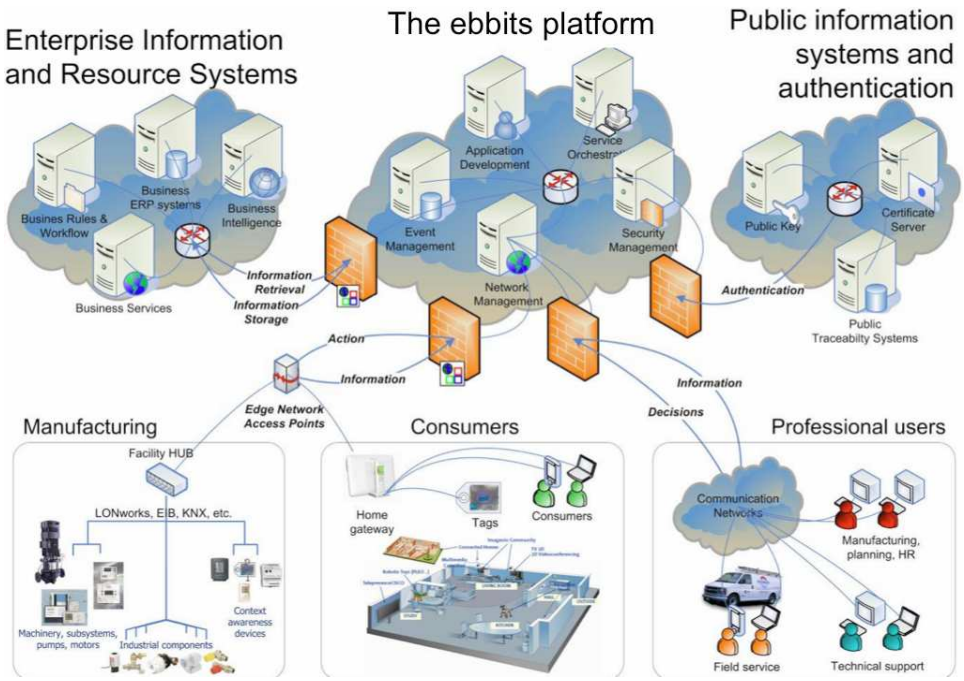


Figure 1 - The ebbits platform with a prototypical actualization of SOA

In order to put the data, service and event management subsets in context, we need an overall ebbits architecture.

An initial sketch of an ebbits data fusion architecture is presented in Figure 2. This architecture is intended as a frame of reference in the further design of service, event and data management in ebbits. We also note that this is an initial basic architecture resulting from the analysis and state-of-the-art work done in WP7, and which is subject to further design.

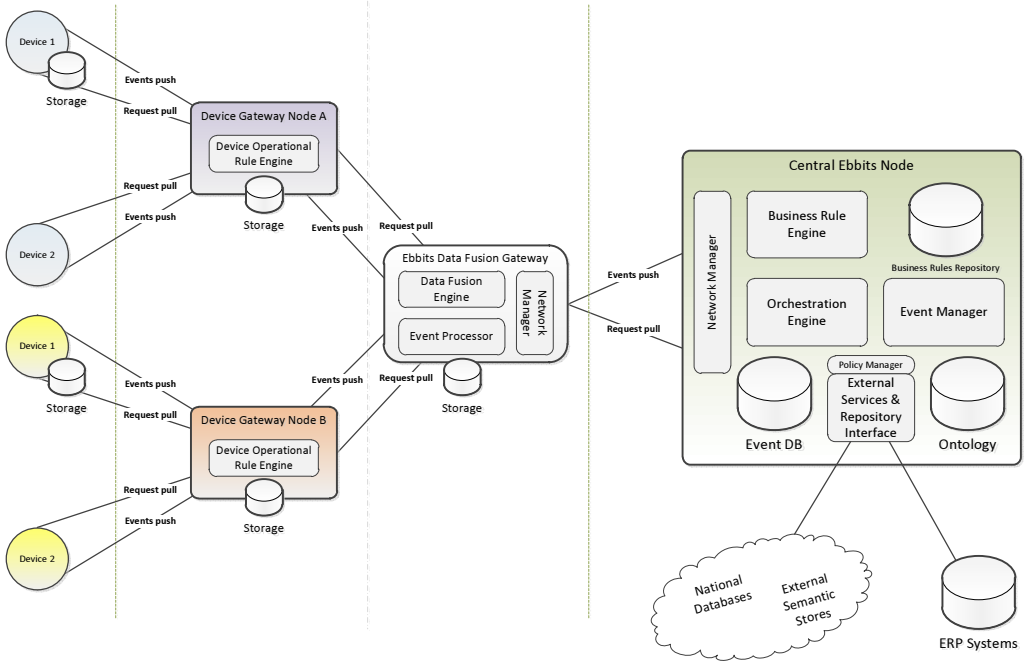


Figure 2 - ebbits data fusion architecture

The architecture perspective here is a functional component model, emphasizing the flow of data/events and control (it could also be projected as a layered architecture). Device components (layer) at the bottom left and the Business rules and services components at the far top right.

The approach to data fusion in ebbits relies on model reuse, i.e., both the data fusion architecture model and the layered distributed vs. centralized model should be able to overlap and hence be deployable in both food traceability and manufacturing domains.

An overview of the architecture components follows in order to better introduce their roles in the overall system:

- **Device 1, 2**

  These devices can be sensors, actuators or even subsystems spread over the ebbits physical environment. For example, it can be a temperature sensor, a Radio Frequency Identification (RFID) reader or some sensor measuring mechanic movement (e.g. pig feeder or roller rotate). Devices generate physical events such as a new temperature value (*stimuli*). ebbits device storage occurs as close as possible to the device itself. This is done in order to ensure full traceability capability in the ebbits architecture.

- **Device Gateway Node A & B**

  The ebbits device gateway node can be based on a fixed or mobile device such as personal digital assistant (PDA), Smartphone or PC. In WSANs (Wireless Sensor and Actuators Networks), gateways will adopt the IETF protocol 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) to communicate with nodes. Sensor nodes will use contiki OS.

  The gateway enables rule assessment derived from its associated network devices and performs rule execution on the gateway node level. Each gateway node can be extended with its own storage. This storage can be used for caching events and incoming data, etc. in case of network failures and thereby enabling store, carry and forward communication (opportunistic approach). Device gateway nodes combine physical stimuli into device events such as "light is turned on".

  *Device Operational Rule Engine*

  Each gateway will have a rule engine, executed locally, intended to run a set of device specific rules. For example, to prevent the temperature on a device 1 to exceed 25 degrees.

- **ebbits Data Fusion Gateway**

  The aim of ebbits data fusion gateway is to fuse data that are gathered via a number of ebbits environment's device gateway nodes. Alternatively, the device gateway nodes and the data fusion gateway can be implemented within the same operating platform and thereby enhance the local processing performance. The data fusion gateway combines events from one or several devices and creates application events.

  *Data Fusion Engine*

  The data fusion engine processes (e.g. aggregates and filters) data from sensors and devices, taking time into consideration. The fused data are further sent to the event processor component.

  *Event Processor*

  The event processor consumes events and data and creates new application events according to its event management logic. The event processor will dispatch events to the ebbits central node.

  *Network Manager*

  The Network Manager implements P2P network functionality. It provides a virtualization of service endpoints for devices and applications, enabling event dispatching and service invocation across network boundaries.

- **Central ebbits Node**

  The central ebbits node is intended to run on high performance machines in the ebbits architecture. Hence, it will be able to offer the computational power needed to support more intelligent components. This central node can be modelled and positioned as centrally as

possible within an ebbits environment, e.g. at the ebbits service provider. It requires stable communication with external resources and information providers.

*Business Rule Engine*

This rule engine processes a set of business rules defined by the ebbits platform user and describes the intended work flow organisation of the specific domain. Business rules are mapped to services via the orchestration engine. The rule engine uses its own repository. The business rule engine combines one or more application events into a business event which is forwarded to an external business system.

Business Rule Repository

Here the business rule engine stores and retrieves the set of rules.

*Orchestration Engine*

The orchestration engine performs tasks on the request pulls over the ebbits network. This can be configured partly by the business rule engine and partly manually through an interface.

Ontology Database

Here the device ontology is stored for use within the ebbits central node.

*Event Manager*

The ebbits event manager handles events that are broadcast throughout the network architecture. It deals with events processed on all levels in the ebbits platform and provides event management to external parties.

Event Database

Provides a chronological and/or source-based log for all events, intended to support data mining and traceability analysis.

*Network Manager*

Similar to the Data Fusion Gateway (see above), an instance of the Network Manager will also be required to run on the central ebbits node.

*[Policy Manager] + External Services & Repository Interface*

The policy manager provides access controls by handling communication to external resources. It guarantees security by restricting access to the different services/repositories involved in the ebbits global system.

ERP Systems

For example, SAP, COMAU, etc.

National Databases

For example, regulatory databases for national agriculture control.

External Semantic Stores

For example, a semantic store in a different ebbits node such as an Event database.

## 3.2 Components from the manufacturing scenario

The next sections will describe all devices included in the manufacturing scenario based on the classification introduced before.

### 3.2.1 Powerful devices

**PC-based HMI (Human Machine Interface):** computers used for monitoring the industrial process. Each HMI runs a customized program and connects to the Programmable Logic Controller (PLC) via Ethernet TCP/IP and OPC protocols. They convert the real-time signals status proceeding from the PLC into easy-to-understand visual pages depicting the process status. On an average automatic line one HMI per station is typically present.

**Scada PC (Supervisory Control And Data Acquisition):** it consists of a PC connected to the plant (usually via Ethernet or serial communication interfaces) and executes specific program used in industrial process control applications for centralised monitoring and recording the status of switches, temperatures, information about production. It provides the operator with a user interface to quickly visualize a full real-time overview of the production process. Usually there is one Scada PC per line providing the whole overview of the manufacturing operations.

**C5G Robot controller**: it is an industrial system created for user-friendly, efficient and multi-purpose management of all SMART5 COMAU Robot range, that can be interfaced through the most common communication interfaces (USB, serial, Ethernet) as well as with the most common field Buses and communication protocols. It can become an Ethernet node on the plant network to facilitate remote updating and diagnostics. In COMAU high density standardized automation stations, up to 16 robots can collaborate on one single production operation.

Figure 3 - C5G Robot controller

**Vision systems**: They are composed of a set of cameras connected to an elaboration unit that performs calculations on the images captured by the cameras and communicates the results to other devices through various communication protocols. They are used for inline real-time quality control, to guide the robots and to recognize production parts. The number of systems installed in a manufacturing plant is function of the process planned.

### 3.2.2 Constrained devices

**Local PLC:** It is a custom elaboration unit that presents reliability characteristics that surpass the quality standards of the commercial devices. Local PLCs are installed on a single production module and have the specific task to manage and synchronize the local process activities, for example to coordinate the robot working on the parts to produce. Local PLCs are devoted to the management of one single production module.

Figure 4 - Local PLC

**Frequency inverter:** It is a system for controlling the rotational speed of the motors installed in the station used, for instance, for moving rollers, lifters and turn tables. They are connected to the line using the most common field bus communication protocols and using a standard communication interface like Ethernet. They can give a full overview of diagnostic faults and they can provide a

remote updating of the configuration parameters. On the average a couple of inverter is present in a station.

Figure 5 - Frequency inverter

**Custom hardware HMI:** It is a product perfectly tailored to the individual hardware and software requirements. They use a locked firmware and a specific software with very few customization options. Sometimes they can have merely an alphanumeric display with just numeric keys and they give the operator only the chance to set few values. It's possible to find them in easy operation stations, normally one of them is used to manage a couple of stations.

Figure 6 - Custom hardware HMI

**Welding controller:** It is a box usually set above the Robot Controller cabinet and contains all the devices and circuits needed for the welding process. Usually, there is a welding controller connected to each robot by means of a multi-bus cable. Moreover, the welding controller can be accessed by a specific terminal or remotely by a PC (adopting the most common industrial communication interfaces) in such a way that the user can configure all the parameters considered for each welding spot.

Figure 7 - Welding controller

### 3.2.3 Sub-systems

**Line supervisor PLC**: It presents an architecture and characteristics very similar to the Local PLCs, but it has the different task to manage all the single production modules and to handle communication inside the plant giving all the information about the production processes, the faults in each operation to an external upper control level. For each single automatic line there is one supervisor that coordinates the activities of the local PLCs used in the line.

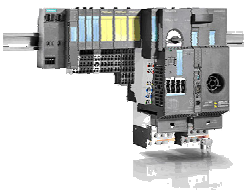

Figure 8 - Line supervisor PLC

**Field-bus interconnected devices:** some years ago each device was connected to the PLC directly with a cable, all this meant a big quantity of cables coming to the cabinets and running though the plants. For an easy installation and a better standardization the field-bus was born: a cable connecting all the devices and arriving to the PLC that replaced all the previous cables, using only one protocol and just one type of connection. Only one field-bus network per station is considered. Examples of these devices are valve pack, remote I/O and safety devices.

**Station subsystem:** this section includes all those devices (indistinctly constrained devices or powerful devices) that are interconnected and seen by the system like a single object: an example could be a station controlled by a PLC and composed by several I/O devices, valve packs to lock the items to work on, one or more robots (handling or welding for instance), networked safety devices like emergency stop buttons, safety doors and light curtains.

**Automatic backed-up devices:** in the welding plants each welding line has at least one Scada PC. On it an application for automatic backups is installed. Every a pre-set time it can execute a full backup of the system, saving the PLC program, the robot applications and all the welding parameters, the inverter parameter, the fieldbus map and all necessary information in case of failures. All these information are also sent to a central backup server It becomes very useful when it is necessary to replace a device in short times in order to reduce production losses due to downtime, restoring in few minutes the full functionality of the device replaced.

## 3.3 Components from the traceability scenario

Modern agricultural production involves heavy use of ICT. The agricultural sector consists of relatively few suppliers of input to the primary production as well as few large purchasers of the output from the farms. Input could be feed, fertilizer, chemicals etc. and output is typically pigs, cattle, milk and crops. The suppliers and purchasers (feed production factories, slaughter houses, dairy processing plants) are typically large factories with centralized ERP systems (like SAP) for full control of production and costumers. In relation to ebbits and food traceability, the obvious mode of communication between these suppliers/purchasers and external systems will be trough web services.
On the farms the data communication is slightly more unstructured. Devices and sub-systems according to the basic production processes (described in D5.1.1 "Concepts and technologies in intelligent service structures 1", sec. 4.3: TNM Business and Expertise in Meat Production Lifecycle) are commonly used. In recent years, the typical way of communication between farm automation equipment (ventilation systems, feeding systems, milking systems etc.) has moved from serial communication with vendor specific (and often secret) protocols towards standardized TCP/IP.

The modern farm uses *one* LAN for communication between the management system and the production equipment, even if the production takes place on different sites. Wireless technology is often used for network connections between two production sites (wireless bridges) and for Wi-Fi coverage in the production areas. Relevant devices considered in the LAN are described below.

### 3.3.1 Powerful devices

**PC:** The farmer normally has a PC/server placed in his office. This PC is used for all his business activities, banking and e-mailing as well as managing the production. Management of the production in pig production can be done using software packages such as AgroSoft WinSvin. In animal production there will typically be a computer at each production site for surveillance and management. The employees will have remote access to these computers from home for use in case of technical disorders outside normal working hours. The PC is also used to control the subsystems mentioned later in this section.

### 3.3.2 Constrained devices

**PDA:** Agricultural production implies a lot of daily registrations (mating, birth of animals, replacement of animals, spraying, fertilizing etc.) and many software packages for farm management come with a module for execution on a mobile device like a PDA. Registrations on the PDA are typically synchronized with the server/PC based management system by use of a cradle or by Wi-Fi.

Figure 9 - PDA

**PLC:** Within the traceability scenario, slaughterhouses present many similarities with manufacturing plans. Similarly, PLCs can be used in order to control production lines.

### 3.3.3 Sub-systems

**RFID tags and readers:** Animals in cattle and pig production are identified by RFID tags mounted in the ear. The farmer can choose among various ear tag readers, but common for them all is, that they are able to communicate with the management system via Bluetooth or direct serial connection to a mobile device like a PDA.

Figure 10 - RFID tags (left) and RFID readers for PDAs (right)

**Farm automation systems:** The farm automation systems are considered sub-systems as they consists of a collection of sensors (e.g. RFID readers) and actuators (e.g. "hand out feed to animal" valves and pumps) that are *"hidden"* behind the controller (which exposes the functionality as a whole). A number of farm automation systems exist.
The feeding system feed the animals. In some cases the feeding is controlled using RFID tags on the animals and information on the status of the animal. This is the case with the Skiold DM-5010 system. Other systems feed the animals according to input (number and size) made by the farmer. This is the case with the Skiold DM-6000 system. The ventilation system used temperature sensors to determine the speed of ventilation and if it is necessary to spray the animals with water.

As mentioned before, the modern farm automation equipment communicates via LAN. The normal setup e.g. for the ventilation system would be with a small local controller for each house (section) in the stable and a central unit (a PC, embedded PC or extended controller) that gathers data from the small controllers. This central unit typically has a user front end which is used for monitoring and change of settings. This unit also serves as the communication link to the world around the specific system. Most of the equipment communicates with other sub-systems like the management system by use of vendor specific data formats and protocols. But in the near future many vendors are expected to standardize communication according to standards like ISO 11788 (ISO 2000).

## 3.4 Enterprise systems components

ERP (Enterprise Resource Planning) applications provide a way to manage and monitor all resources in an enterprise. This includes employees (human resources), capital (financial resources), operational resources as well as production material. Moreover, standard ERP applications also cover the following aspects: accounting, controlling, research and development, sales and marketing and master data management. In addition to modelling the aspects of the entire enterprise, ERP systems provide integrated workflows and business processes on top of the enterprise data.

SAP is the world's largest provider of ERP applications with a global market share of above 26 %. As a result, it is very likely that there are use cases for the application of ebbits which require the integration of SAP ERP systems. Nevertheless, there are of course a lot of other ERP applications on the market and in the field, ranging from very simple domain specific solution to generic systems. In order to reflect this aspect and to foster the application of ebbits to various systems, an additional component, an ERP Hub, has been introduced. This server component provides an interface for persisting and retrieving data required in the manufacturing as well as the traceability use case. It allows adding concrete implementations for specific ERP systems later as required. Moreover, the interface can be used to access data from dedicated ERP systems by means of various client implementations. In the demo prototype already developed in ebbits, there are two web clients, one for the manufacturing and one for the traceability use case. At this point of time, the ERP Hub provides an implementation for a SAP ERP system. The various components are introduced in more detail in the following sections.

### 3.4.1 ERP Hub

The ERP Hub has been introduced to decouple the ebbits integration platform from concrete ERP system. In doing so, it is guaranteed that the integration with ERP systems is not bound to one system but rather agnostic. The Hub is implemented as an application server which provides a restful Web Service interface designed to meet the requirements of the manufacturing and traceability use cases. Specifically, this means it provides methods to store (HTTP POST), to delete (HTTP DELETE) and to retrieve (HTTP GET) use case specific data, e.g. medication information of pigs as well as water and electricity consumption data. The supported data formats are XML and JSON. In the M6 demo, data is provided by a dispatcher component registered to the LinkSmart middleware. This, in turn, retrieves data from a water pump or a medical PDA application.

In the current state, the ERP Hub contains an implementation of a local data store as well as can be configured to forward the retrieved data to a SAP ERP system. At the moment, the protocol used for that is HTTP. The ERP Hub is a self-contained application server. Therefore, it does not depend on another host server, as e.g. tomcat[1] or jetty[2]. The port is freely configurable.

### 3.4.2 SAP ERP

SAP ERP is the core solution of SAP's Business Suite, the generic and customizable All-In-One product for large enterprises. Within this product, SAP ERP deeply integrates with other important solutions like CRM (Customer Relationship Management) and SCM (Supply Chain Management) in order to support end-to-end business processes. For a more detailed description of the available components and system architecture the interested reader is referred to Deliverable D.7.1 "Concepts and technologies for unified lifecycle, persistent data fusion architecture".

The technological foundation for SAP ERP and all other SAP solutions is SAP NetWeaver. In general, the infrastructure consists of one database server and several application servers. This concept of a central data storage and load-dependent distribution of application and presentation-logic has proved as being highly scalable.

SAP ERP provides various interfaces for accessing data. Examples are:

[1] http://tomcat.apache.org/

[2] http://jetty.codehaus.org/jetty/

- BAPI (Business Application Programming Interface): BAPIs are standardized, platform-independent interfaces to SAP Business Objects (BO) which provide an open and object-oriented view on business processes. Typically they are invoked using the ABAP programming language but there are also Java libraries (JCo[3]). Network transfer is realised using TCP/IP on ports 3300-3399.
- SAP WebAS (Web Application Server): The WebAS exposes interfaces for data exchange in HTTP(S), SMTP and SOAP/XML to the internet and intranet.

The SAP ERP is hosted internally at SAP and provides an HTTP interface for the two ebbits use cases. The data sent from the ERP Hub is processed using ABAP and stored in the central NetWeaver database. Due to license restrictions, the server is not accessible from the Internet at the moment.

### 3.4.3 Web Applications

The data stored in the ERP system and provided by means of the ERP Hub interface can, apart from ERP specific client application, be consumed and visualised by various client implementations. In the current development phase of ebbits, we provide various web applications to access ERP data. The reason for that is that, in contrast to desktop application, they are platform and device independent, lightweight and easy to maintain. Apart from that, those dedicated implementations allow for demonstrating individual aspects of various use cases rather than adopting an existing fully-fledged ERP frontend. The current prototype consists of two web interfaces both based on the Adobe Flex framework. As a result they can be embedded in any HTML website. The latter can either be used only locally or served by means of a Web server.

**Manufacturing Frontend**


Figure 11 - Manufacturing web interface

The UI provides information about the water and electricity consumption of a water pump. The needed data is obtained from the ERP Hub. Moreover, the UI allows to enable and disable the corresponding pump. In order to do so, the UI calls a Web service provided by a LinkSmart component using HTTP POST. Furthermore, a webcam stream is integrated into the UI showing the status of the water pump. The stream is provided by ISMB and is publicly available from the internet via HTTP on port 80.

**Traceability Frontend**

The traceability Web interface allows monitoring the health state of pigs. The data is illustrated by three different icons: a check depicts that all pigs are healthy, an exclamation mark that in the range of 3-10 days at least one pig has been medicated, and a cross that in the last three days pigs were

[3]http://help.sap.com/saphelp_nw04/helpdata/en/6f/1bd5c6a85b11d6b28500508b5d5211/content.htm

medicated. To see the medication details (dose etc.) the farmer can select a pen and choose a specific pig from a list.

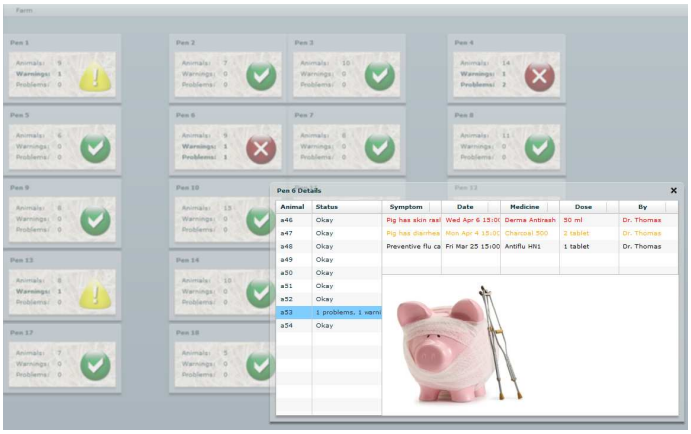


Figure 12 - Traceability web interface

The corresponding data is retrieved from the ERP Hub. There is no other connection needed.

# 4. Network infrastructure

The network architecture is a vital part of the prototypes. Applications within ebbits usually hide the underlying network by the use of services. Nevertheless, they will depend at lower level from different network components and technologies. These components should be interconnected to support connectivity, allowing end-to-end communication between ebbits devices.

This chapter will provide a brief introduction to network components and protocols which are essential to support the required IP connectivity that should be provided to every device needing to access the network during the field trials.

## 4.1 Network Components

The components that will be described below will be in charge of forming the network system which provides IP connectivity to the different field trials. For each component a brief description will be provided, considering they are well-known networking devices, together with the role which has within the network infrastructure.

### 4.1.1 Switches

Switches provide data link layer connectivity based on the interpretation of MAC addresses information, creating a unique collision domain within the network. Other tasks include communication filtering, data transfer and broadcasting. Switches are usually the primary wired network equipment where all devices connect.

During field trials switches will be used to provide devices and sub-systems using Ethernet, the first point of access to the network. The list of devices is detailed as follows:

- Manufacturing scenario
    - PC-based HMI and ERP Systems
    - SCADA PC
    - C5G Robot Controller
    - Sub-systems
- Traceability scenario
    - PCs (Controller, ERP systems)
    - Farm automation systems

### 4.1.2 Wireless access points

An access point (AP) provides wireless access to the network as its name states. They are used in places where cables do not represent a suitable solution or just to support connection redundancy. For instance, in rural scenarios, where distances among ebbits and network devices are relatively far, the optimal way to share the Internet connection among devices inside warehouses is by using wireless links. Wireless access points in most of the cases are connected to switches, which provide the network connectivity.

During field trials switches will be used mainly in the traceability scenario, to provide wireless access to the PC based management and PDAs.

### 4.1.3 Routers

The router works at the layer 3 of the OSI reference model, so it deals with the network protocol that solves two main tasks in the network – addressing of network interfaces and routing.

Routers deal with the address of destination node and, based on information available in routing tables, forward the packet to the appropriate network interface. The performance of the overall computer network is often determined by the performance of the network layer components – i.e.

routers. Typical performance metrics are link speed, aggregate throughput, packet rate and power consumption.

During field trials routers will be used to interconnect switches and access points.

### 4.1.4 Network Address Translation (NAT)

During the last years, the problem of lack of "free" IPv4 addresses has been arising. One of the solutions advanced to face the problem is moving to the IP version 6 that uses 128-bit addresses. One solution in order to reduce the impact of the IPv4 problem is to repeatedly utilise private addresses, which are not visible and accessible world-wide. Network interfaces inside a private network may use any IP addressing scheme, regardless of whether it is recognized as legitimate by the Internet authorities. As soon as the destination for the communication is outside of the private network and the reply is expected, the internal private address of the source has to be changed to a valid public IP address. A special device (working at the layer 4 of the OSI model – the transport layer) opens the IP datagram and replaces the client's private IP address with an Internet-recognized IP address. This process is known as NAT (Network Address Translation).

NATs will be implemented during field trials in both scenarios, mainly because of the number of devices involved and because some devices should not be able to be accessed from the outside of the network.

### 4.1.5 xDSL/GPRS/UMTS/HSPA Modems

A modem is a device which connects to a wired/wireless network to get Internet access. For instance, smartphones using GPRS or UMTS/HSPA could be used as modems in order access the Internet. Another common example could be the ADSL modem that can be found is most of our homes nowadays. This ADSL modem uses the telephone network to get Internet access and share it to devices within our houses.

These devices will provide the connection to the Internet and will be connected either to switches, or routers. In some cases, PCs and PDAs could have included integrated modems as well.

### 4.1.6 Firewalls

Firewall is a specialized device, or a computer installed with specialized software, that selectively filters or blocks traffic between networks. A firewall typically involves a combination of hardware and software and is located between a private network and a public network (such as the Internet) (Dean 2010).

Firewalls can operate in different ways. They can examine the addressing information in the IP packets and based on positive or negative filter rules stop the unwanted or not allowed packets. Of course, the processing of every packet is a time and computer power consuming task, so sometimes the filtering based on stream connection is preferred. As far as the connection is proved to be allowed, packets belonging to this connection are no longer checked. According to the type of checked information (e.g. address in the IP packet, communication end point in TCP connection, application specific data…) we can distinguish layer 3 firewalls, layer 4 firewalls or layer 7 firewalls (often called application proxy).

During field trials, firewalls will be used to define security policies regarding private enterprises and general policies.

## 4.2 Communication protocols

Within the network architecture, communication protocols provide mechanisms to make possible the exchange of information among devices. Communication protocols define messages formats (i.e., length and meaning of the many fields) and are constituted by a set of rules, defined in a number of algorithms. It's only through sharing common rules (and associated timetables) and packet formats that devices can understand each other.

The ebbits platform will contain several devices with PC-like functionalities; therefore the TCP/IP protocol suite is the most suitable in order to allow the interconnection of such devices, mainly in the traceability scenario. Nevertheless, in the ebbits platform some devices will not be able to use

the TCP/IP protocol directly. In fact, the TCP/IP protocol sometimes lacks specific functionalities with respect to manufacturing environments or wireless sensor and actuators networks. For that reason, some protocols were designed in order to adapt the TCP/IP protocol for devices included in those types of networks or environments.

The following subsections briefly describe the protocols 6LoWPAN and PROFINET with reference to the adaption of TCP/IP into the environments explained before.

### 4.2.1 6LoWPAN

Due to computational and energy constrains, using the Internet Protocol over WSAN would not be possible without some stack modifications. For that reason, the adaptation protocol called 6LoWPAN[4] was created. This protocol allows sending IPv6 packets efficiently through IEEE 802.15.4 networks. Its main concept is to compress IPv6 packets headers, reducing size, to then transmit these packets using IEEE 802.15.4 links. The 6LoWPAN stack adopts as well solutions with respect to routing (e.g. AODV), auto-configuration and network management. Implementations at the moment are not yet widely spread; nevertheless a number of operating systems like TinyOS[5] and Contiki[6], mainly designed for embedded devices and sensor nodes, have adopted the protocol considering 6LoWPAN as a promising technology.

### 4.2.2 PROFINET

Regarding the manufacturing scenario, an open automation standard supported by PI[7] (PROFIBUS and PROFINET International) named PROFINET has been created in order to provide demand-oriented data exchange services within industrial plants. The standard is compatible with well-known technologies such as IEEE 802.3 (Ethernet), IEEE 802.11 (Wi-Fi), UDP/IP, XML and OPC. It incorporates as well automation features to be used specially in manufacturing scenarios regarding discrete control, process automation, motion control, peer-to-peer integration, vertical integration with enterprise IT systems, functional safety and energy. In addition, PROFINET provides a "proxy" solution enabling the seamless integration of several networks, such as PROFIBUS, Modbus, DeviceNet, and computer networks as well.

[4] http://datatracker.ietf.org/wg/6lowpan/charter/

[5] http://www.tinyos.net/

[6] http://www.sics.se/contiki/

[7] http://www.profibus.com/

# 5. Field trials

This chapter will provide a description of the different field trials to be deployed during the ebbits project. Actually, 4 setups are considered in different cycles at months 10, 22, 34 and 46 respectively. For obvious reasons, the descriptions of the last two field trials will be less detailed considering that additional information will be defined in the following months.

## 5.1 M10 setup

This first field trial will take place in June 2011 and will serve as a base work for the upcoming field trials. This first prototype of the ebbits platform will be used as a preliminary proof-of-concept.

### 5.1.1 Manufacturing Scenario

As for the traceability scenario, also for the manufacturing environment the first setup aims to the extraction of the data coming from the field.
The testing cell will be composed by a welding robot equipped with a welding gun and its controller. In order to achieve our goal, specific sensors will be selected and installed in the system, as shown in the picture below:

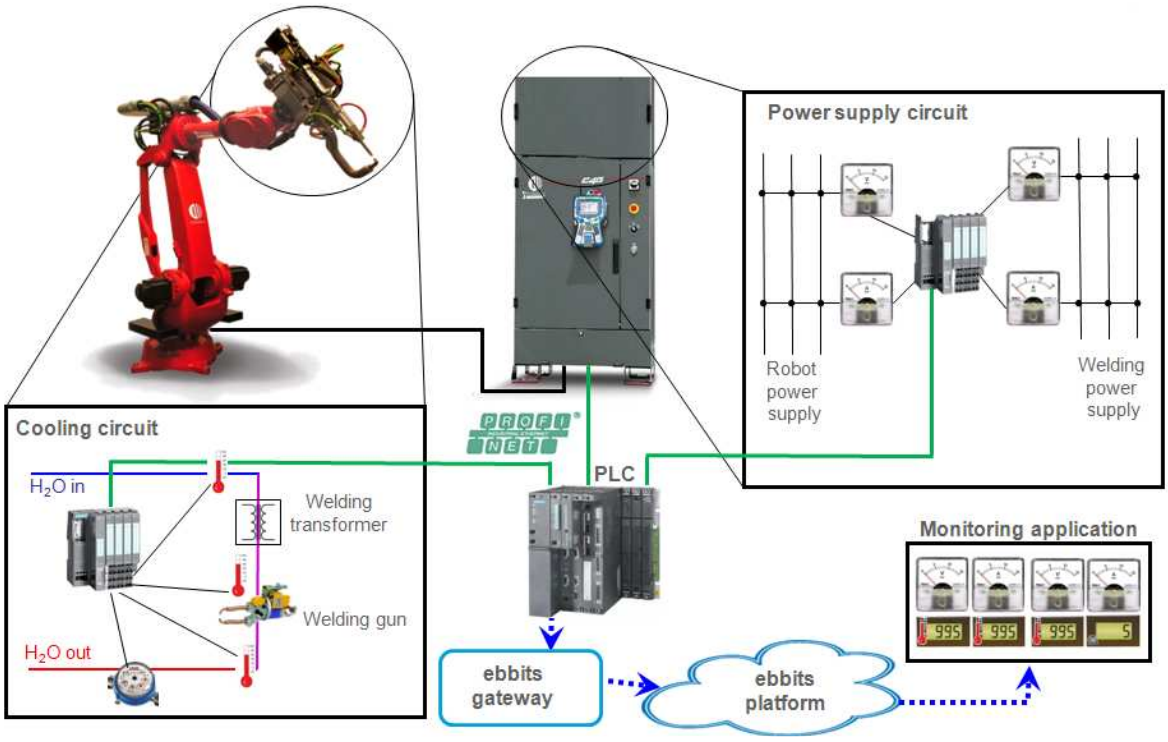


Figure 13 – M10 Manufacturing prototype setup

In this scenario we consider two systems to collect field data on the described robotized subsystem: the cooling circuit and the power supply circuit.

**System components**

*Cooling circuit*

The first part of the measuring system considers the cooling system of the welding transformer and the welding gun, in order to collect information on water usage of the welding system. The acquisition system is composed by a flux meter which retrieves data from the water flowing in the pipe of the cooling circuit; different thermometers detect the temperature of the inflow and the outflow of each cooled device. Using the data collected from the system, we are able to elaborate

the amount of water used and wasted and moreover we can calculate the amount of heat drained from each cooled device. Using these data we can develop a strategy to optimize the use of water to cool the system during the welding operation.

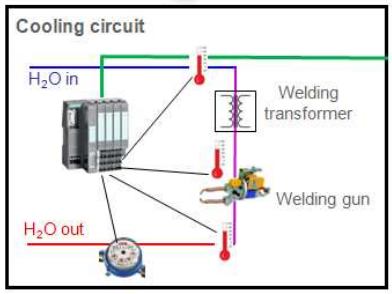


Figure 14 - Cooling circuit

*Power supply circuit*

The second part focuses its attention on the power supply circuit: it uses two separate circuits for the robot axis power supply and for the welding power supply, because they use voltages of different magnitude (about 380 V for the robot and 5-6 V for the welding circuit). In particular using a voltmeter and an ampere-meter on each power supply we go to analyse the current consumption and its voltage. In this way we can understand some important information about power consumption, the cos$\varphi$, which is the ratio between the active power and the apparent (or reactive) power. This value has to be as low as possible to ensure a high efficiency of the plant, because a high amount of reactive power through the plant means an increase of current flow and therefore an increase in total losses of transmission resulting in a reduction of efficiency of the installation.
Usually the welding circuit should not encounter problems, because the charge is only made of resistive power, but in actual circuits using an inverter the frequency increases from 50 Hz to 1KHz, in this way it is possible to reduce the size of welding transformer

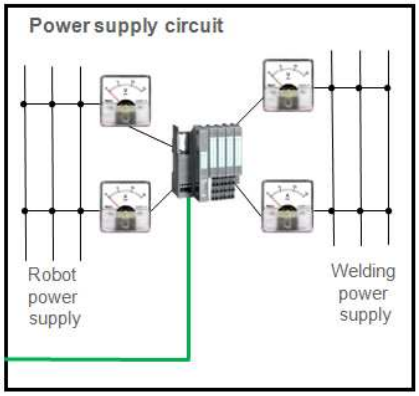


Figure 15 - Power supply circuit

*PLC*

In a working manufacturing plant a Team is designed for analyse the data coming from the production plant. They have also to produce trends about the main issues, the most frequent stops, part produced and their production times, and so on. The main problem they face is the heterogeneity of information coming from sensors, safety elements, PLCs, barcode readers and RFID. All these information also come in different moments and often are not correlated also when referring to the same event or production status.
The solution to this problem is to collect them through a middleware: a device placed between the ebbits platform and the production plant. The designed device to this aim is the PLC: it is physically placed inside the production line and it can easily provide all the information about produced parts, stop times, safety stops, cycle times and so on.

**Network infrastructure**

The welding robot and the PLC are connected through a fieldbus, and in particular the Profinet. The monitoring application communicates directly with the ebbits platform connected to the PLC through a gateway, in the specific an OPC Service. In this first setup the connection between the sensors and the monitoring station is completely wired.

### 5.1.2 Traceability Scenario

The first traceability setup considers extracting information from the feeding and management system of a pig farm and integrating it into the ebbits platform.

The relevant specific pig farm is "Frihedslund" which is a pig farm with a capacity of 1000 sows and production of 27000 32 kilogram pigs. The farm also produces feed from crops from their own fields.

**System components**

The feed is both mixed and distributed by feeding systems from Skiold A/S. The sows are fed according to their fertility state by a DM-5010 transponder system. There should be a module that extracts the relevant information from the DM-5010 system and forwards it to the ebbits system.

The piglets are feed according to their age/weight by a Skiold DM-6000 system. The DM-6000 controls the feed mills, pumps, valves etc. from a central PC. There should be a module that extracts the relevant information from the DM-6000 system and forwards it to the ebbits system.

The management system on the farm is AgroSoft. This particular management system stores data in a "BTrieve" database system. There should be a module that can extract relevant data from the database and put it into the ebbits system. The AgroSoft system is a PC based management system which supports synchronizing data to and from PDAs on which data can be entered. The synchronization can be done either via the cradle (a docking station) or via the wireless network.

**Network infrastructure**

The feeding system and the management system are installed on two different standard Windows PCs connected to the LAN of the farm. The LAN is a standard NAT network behind a standard consumer router connected to the Internet via ADSL. The LAN inside the animal houses is cabled as wireless connectivity within the animal houses has been shown to be unreliable. The problems are probably due to the construction of the house as the pens and sections within it are made mostly of iron which reflects the signals of the wireless network. However, it may be possible to improve the situation by using better equipment such as professional grade access points and directional antennas.

This particular network setup is very common among farms. However, in some cases a farm has segmented networks in order to separate the production relevant systems from e.g. the home usage of the network.

Furthermore, as farms are generally placed in remote and sparsely populated areas, they face problems with respect to the quality of their internet connection. The quality of a wired connection is inversely proportional to the distance between the end points. In the worst cases, it is sometimes not possible to get a wired (ADSL, fiber) connection and a wireless solution, such as WiMAX, must be used. In any case, we can assume that the internet connection is slower and could be more unreliable than connections found at, for example, residential places. In our particular field trial we work on a farm with a standard quality ADSL connection.

## 5.2 M22 setup

The second field trial should be deployed in June 2012. This field trial, based on the prototype developed in M10, will be extended by including new features focused on production optimization.

### 5.2.1 Manufacturing Scenario

The second prototype should include the ability to connect the sensors directly to the ebbits gateway.

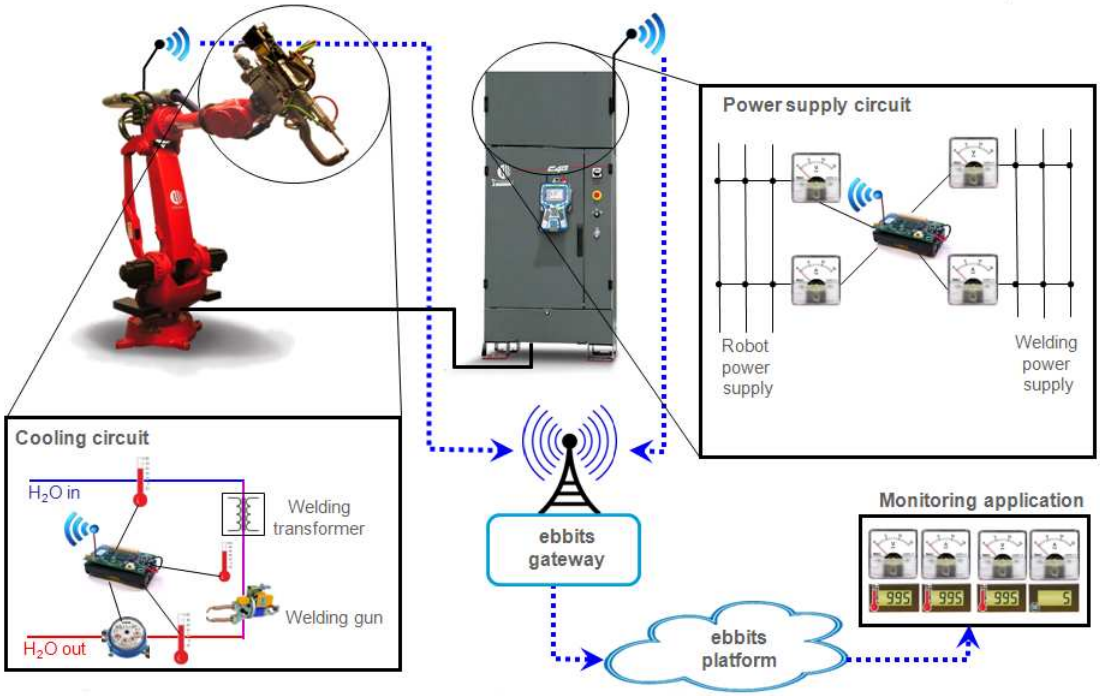


Figure 16 – M22 Manufacturing prototype setup

**System components**

The system components implicated in this second scenario are the some sensors installed on the robot circuits in the first setup with implementation of wireless interfaces.

**Network infrastructure**

Regarding to the network infrastructure in the second setup, as shown in the picture above, the sensors will interact directly with the ebbits platform through the ebbits gateway. In order to achieve this objective, wireless transmitters using the 6LoWPAN communication stack will be integrated. More information about this protocol can be found in the chapter 4.2.2.

### 5.2.2 Traceability Scenario

The second prototype should include the ability to include data from the slaughterhouse.

**System components**

The production data of the slaughterhouse is stored in the ERP system. Data from the ERP system should be extracted and put into the ebbits platform.

**Network infrastructure**

The network at the slaughterhouse is more restricted than the network seen at the farms. The slaughterhouse is setup as an industrial facility and ebbits will need to handle more restrictive firewall setups.

It is anticipated that the network infrastructure within the slaughterhouse is in place with respect to the data that the traceability scenario demands.

## 5.3 M34/M46 setup

The last field trials will show the partially/full set of ebbits features in a real world environment based on the previous field trials. The prototypes should be developed by June 2013 and June 2014, respectively.

### 5.3.1 Manufacturing Scenario

Within a manufacturing plant there are several production lines each of them with the express purpose to realize a finished product or a sub-product needed in other production processes.
When the task of analysing the production trend is not limited to a single production line, but to the whole manufacturing plant, it is not easy to follow the complete production flow: consolidated data are needed and they should be quick to interpret, to help solve problems or to develop new production strategies a short time.

Figure 17 - Production line inside a manufacturing plant

To achieve these goals it is necessary to apply in large scale what was used in the second scenario: this is the third step of the demonstration, in which we will demonstrate the scalability of the project: in other words the application will be able to be expanded to the whole manufacturing plant simply including new measuring devices to the system, increasing the measuring capabilities like bricks that can be combined at will, making the application extremely versatile. All the production lines will be correlated and all the collected data will give a full, easy-to-understand, overview about the status of the whole manufacturing plant. This big quantity of information is important for the manufacturing suppliers for internal studies, efficiency tests of the provided plants and faults simulation. Normally the plant KPI simulation group has large computational tools, but few data and very fragmented, that's why some different plants of the same company could be connected together through the ebbits platform, giving a complete overview and a large amount of data.

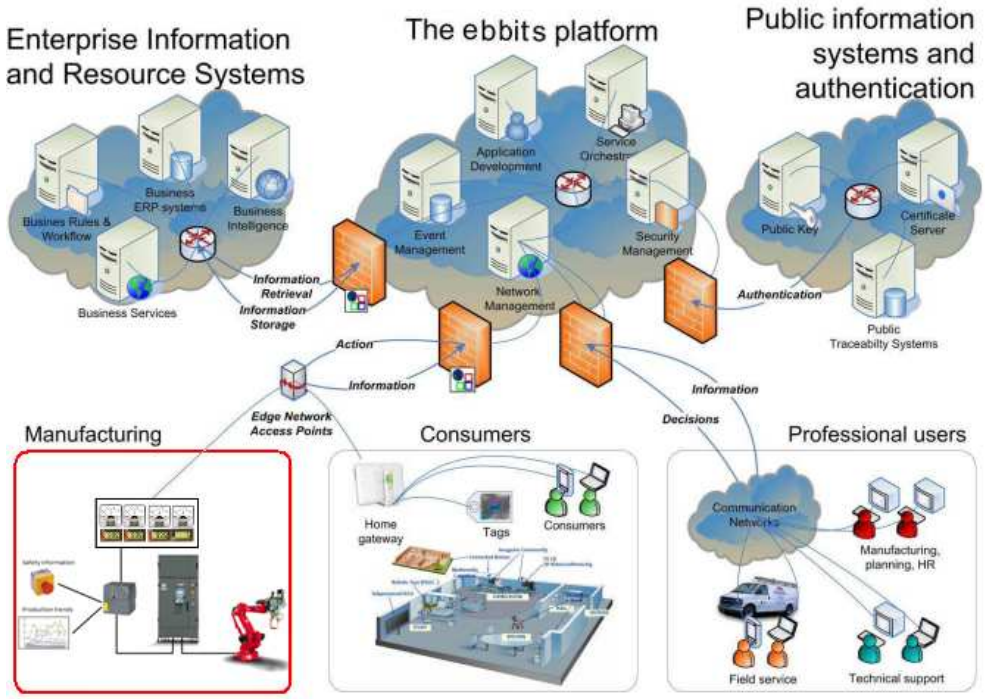


Figure 18 - Manufacturing scenario within the ebbits platform

**System components**

**ERP and MES**

The ERP is a system which integrates all the information related to the organization, the manufacturing, the finance, the sales, etc. of a manufacturing plant. The whole system is automated through a software application. Its purpose is to simplify the information flow between the different business unities. Unfortunately the manufacturing suppliers have not access to these confidential data and for the moment it cannot be implemented in the ebbits platform.

The Manufacturing Execution Systems (MES) are information technology systems that manage manufacturing operations in factories. Most MES systems include connectivity as part of their product offering. Direct communication of plant floor equipment data is established by connecting to the PLC. Often, plant floor data is first collected and diagnosed for real-time control in a SCADA system.

**Network infrastructure**

The network in a manufacturing plant is more restricted than the network of a single line and ebbits will need to handle more restrictive firewall setups.

### 5.3.2 Traceability Scenario

The last prototypes should include the end-user. The end-user should have the possibility to access the traceability data collected throughout the lifetime of the given product.

**System components**

Packages should be marked such that a device of a consumer can read it. The device could be a cell phone. Thus the ebbits system should now allow access from consumer devices such as cell phones.

**Network infrastructure**

In order to facilitate cell phones to obtain traceability information it should be possible to access the data via the internet and thereby through e.g. the 3G network.

# 6. Integration plan

The integration plan will specify the methodologies or approaches required to define the different steps towards the overall integration of the components conforming the ebbits platform. The integration is done in terms of providing network connectivity among the different components and applications that will be used to evaluate and demonstrate the features of the ebbits platform in real-world settings. The integration plan is described in two parts: Network infrastructure and Applications.

## 6.1 Network infrastructure

Generally, the integration process can be divided into two main methodologies: single-stage integration and incremental integration (Sommerville 2001). The single-stage methodology consists in performing the integration all at once, meaning that all sub-systems involved are integrated simultaneously. On the other hand, the second methodology is based on the hierarchy of the involved components using an incremental approach. This means that the whole process is divided into steps or stages depending on the level of importance or availability of the components to be integrated.

In case of large, possibly complex, networks, the recommended approach is always incremental. The simple fact that the network will not be small shows that the integration will be efficient if the process is completed after several stages or steps. Another important point to mention relates to specify which components are going to be first integrated. Considering that the network architecture has a hierarchical structure, the first components to be integrated are the ones which strictly have other components below relying on them. In a simple way, components integrated in the last steps will rely on components integrated in the first ones.

The steps or stages towards the different components integration are described as follows:

1. *network system:* the network system contains the network components forming a stable, possibly heterogeneous, network providing IP connectivity which serves as support for all ebbits components.
2. *ebbits core components:* these components are the heart of ebbits. These components based on the connectivity provided by the network system provide the services to all ebbits devices.
3. *ebbits devices:* all devices (powerful and constrained devices, sub-systems and SAP ERP) that will required network connectivity in order to access the ebbits platform.

The following figure gives a graphical view of the steps listed above:

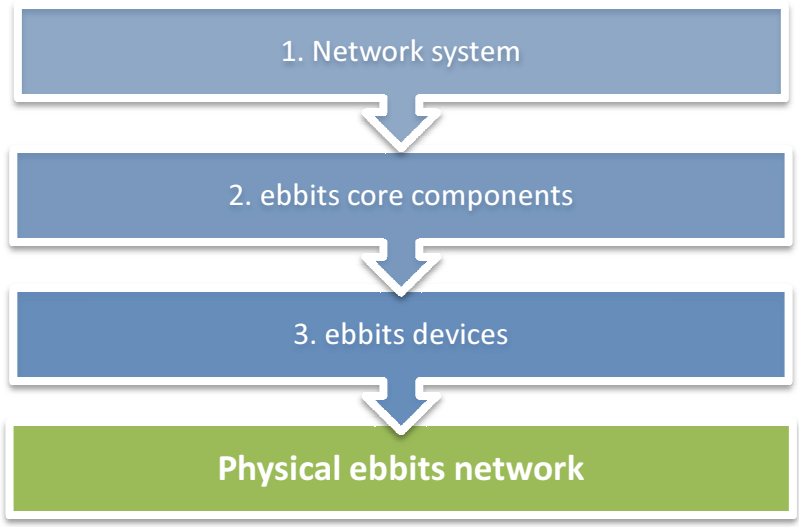


Figure 19 – Steps towards integration

The full list of components considered in the integration plan is described in chapter 3.

## 6.2 Applications

The building of applications in ebbits involves primarily enabling device/sensor communication and configuration of different managers. A large part of the application building will consist of defining rules and events specific for the application. For instance a rule could be to stop a robot if the temperature is over a certain threshold.

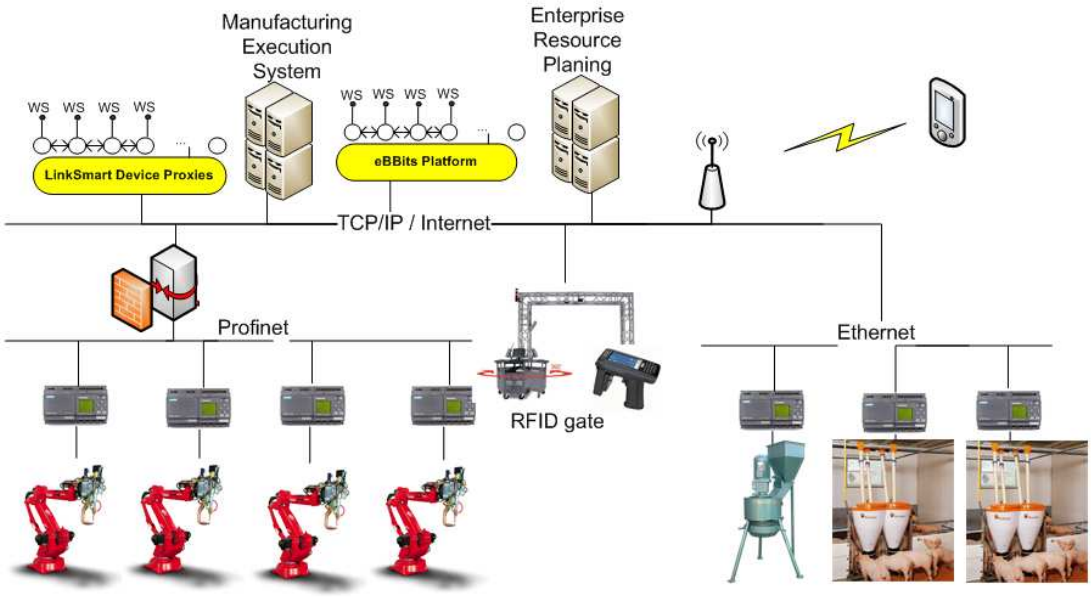


Figure 20 - Deployment example of an ebbits application

The following integration points are necessary to take into account when building an ebbits application:

1. Device proxy creation for the devices/sensors to be used in the application.
2. Configuring External Services & Repository Interface to be able to communicate with the external services such as ERPs etc.
3. Rule and event manager configuration
4. Changes in the evolving ebbits platform (since it is under development).

In a distributed project as ebbits it is important that the bulk work of integration can be done in a distributed fashion without requiring face to face meetings, i.e. we will do incremental integration (as opposed to integrating all units at once). Therefore we need to use common repositories for code (including rules) and other configuration related information. This will enable initial integration and testing to take place in a distributed fashion.

Since the ebbits platform itself will be developed in parallel to the applications it is also important to be able to test the applications continuously against the evolving platform. This process should be speeded up with the use of unit tests in the application code. This will ease distributed initial integration tests between the platform and the applications.

The following sections give an overview of the usage of common repositories in the ebbits project. The repositories used for the applications will be the same as for the ebbits platform, thus allowing for addition of automatic unit testing against the platform.

### 6.2.1 Source Code and Configuration Management

An important part of modern software development is the Software Configuration Management. Configuration Management is often considered to have originated in the frame of software

development in recent years and that it merely consists of a tool for versioning of source files. In software development the core of configuration management is made up of a version control system. Source code (or all source documents of the final product, respectively) is considered a system that spans time and space. Files and directories (which, of course, can contain files) form the space, while their evolution during development forms time. A version control system serves the purpose of moving through this space.

Configuration management contains version control, and extends it by providing additional methods of project management. Software configuration consists of software configuration items (SCI) which can be organised in three categories and which exhibit several types of versions. The following activities constitute configuration management (Bruegge et al 2000):

- identification of SCIs and their versions through unique identifiers
- control of changes through developers, management or a control instance,
- accounting of the state of individual components,
- auditing of selected versions (which are scheduled for a release) by a quality control team.

The Institute of Electrical and Electronics Engineers (IEEE) sees configuration management as a discipline that uses observation and control on a technical as well as on an administrative level. Software configuration management deals with the governing of complex software systems (Westfechtel et al 2003). In ebbits we base the software configuration management on Subversion which addresses many of the problems listed in (Ommering 2003; Bruegge et al 2000) and which is described in more details in D9.1 "Test and integration plan".

# 7. Testing plan

Following the idea of the integration plan, the testing plan will provide the means to test network connectivity of ebbits components and to test end-to-end business applications developed to demonstrate the features of the ebbits platform. The network infrastructure and application testing plans are described in the following sections.

## 7.1 Network infrastructure

Testing the network architecture is an important step in the overall testing plan. The network infrastructure testing plan will consist in a description of the methodologies and expected results from tests made in order to check the network connectivity and performance of devices connected to the ebbits platform. These tests will be performed in both wired and wireless connections, regarding the physical interface used to join to the network.

Within the network infrastructure, several of the components described in chapter 3 are interconnected providing network links to devices willing to access to a domain (e.g enterprise/private network, isolated farm network, etc) within the ebbits platform. Testing the network connectivity of these devices is also an important part of the process, however first we must have the knowledge about how the global architecture (network equipment) is conformed by performing a network discovery. Having a complete map of the network infrastructure will allow debugging any problem that can occur during the testing process.

Once the global architecture of the network is known, the tests can start. Basically, the relevant tests are divided into two typologies: network connectivity and network performance. The division was made considering the higher layer in which the testing process takes place. In the first case, the network connectivity test reaches the Internet layer; instead, the network performance test reaches the transport layer.

### 7.1.1 Network discovery

The network discovery process will consist in detecting all main network components which are currently forming the global network architecture and determining how they are interconnected the one with the other.

For this task, several software products are capable of showing a map of the whole network and how they are related. One example of software able to perform this task is commonly known as IT infrastructure monitoring tool. It provides a monitoring system able to manage and monitor a complete network infrastructure.

Examples of these software products are:

- Nagios[8]
- Big Brother[9]
- NetworkView[10]

### 7.1.2 Network connectivity testing

The network connectivity testing will consist in checking, at network layer, the device network connectivity within a domain. To perform this task, a ping test will be used. The test will run in a PC connected to the ebbits network that will try to reach the relevant ebbits device under test (DUT), which in this case is the device willing to access the ebbits network.

The following chart describes how the test will be executed, the possible outcomes and a final interpretation of the results.

[8] http://www.nagios.org/

[9] http://www.bb4.org

[10] http://www.networkview.com/

| Network Connectivity Test (ping test – Ethernet/Wi-Fi) |
|---|
| Requirements |
| Equipment required<br>*PC with Ethernet card / Wi-Fi card*<br>Software required<br>***ping*** *should be installed in both devices. Note that the test will run even if both devices use different operating systems.* |
| Procedure and configuration |
| Initial configuration<br>Both devices should be connected to the same domain within the ebbits network<br>Steps to implement the test<br>Perform the command ping "ebbits DUT IP" from the PC connected to same domain within the ebbits network |
| Expected Outcomes |
| The ping test will provide information related to the round trip time from our location to the ebbits DUT. The following examples shows an example about some of the outputs that ping provides:<br>**Successful ping test**<br>*admin@localhost# ping en.wikipedia.org*<br>*PING text.pmtpa.wikimedia.org (208.80.152.2) 56(84) bytes of data.*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=1 ttl=50 time=77.9 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=2 ttl=50 time=75.0 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=3 ttl=50 time=74.8 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=4 ttl=50 time=75.3 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=5 ttl=50 time=75.6 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=6 ttl=50 time=78.6 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=7 ttl=50 time=75.3 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=8 ttl=50 time=75.5 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=9 ttl=50 time=75.3 ms*<br>*64 bytes from rr.pmtpa.wikimedia.org (208.80.152.2): icmp_seq=10 ttl=50 time=75.1 ms*<br>*--- text.pmtpa.wikimedia.org ping statistics ---*<br>*10 packets transmitted, 10 received, 0% packet loss, time 9885ms*<br>*rtt min/avg/max/mdev = 74.865/75.880/78.619/1.244 ms*<br>The results show the round trip time (RTT) values, packet loss percentage, test time duration. This test at simple sight shows us if the ebbits DUT can be reach or not.<br>In case of non-connectivity of the ebbits DUT, the ping results will be similar to the following:<br>*admin@localhost# ping en.wikipedia.org*<br>*PING text.pmtpa.wikimedia.org (208.80.152.2) 56(84) bytes of data.* |

```
Request time out.
Request time out.
Request time out.
Request time out.
Request time out.

--- text.pmtpa.wikimedia.org ping statistics ---
5 packets transmitted, 0 received, 100% packet loss
rtt min/avg/max/mdev = 0/0/0/0 ms
```

In this case, no response is sent from the ebbits DUT. This means that the device could not be reached. Some of the possible problems could be:

- Incorrect IP address
- Ethernet or Wi-Fi Adapters are not correctly connected to the PC
- Ethernet cable is unplugged
- Network access points (switch or wireless APs) are not connected to the ebbits network

Note: The software ping uses ICMP messages that could be blocked or filtered by firewalls within the network. Then, firewalls could be properly configured in order to avoid the above issue.

As mentioned before, the connectivity test in done within a specific domain. The use of NATs and private addresses make not possible to perform a ping test to a relevant device integrated in another domain. In this case, the connectivity between two domains is tested between the relevant border routers connected to the Internet. In fact, the test procedure described before can be also applied to check connectivity between border routers. The main difference consists in that the test will be initiated by one of the border routers, performing a ping command to the one on the other side.

### 7.1.3 Network performance testing

The network performance testing will measure the throughput of the network link from both devices within a domain, by creating data streams and sending them from one point to another. The task will consist in a client-server test sending data from device A to device B, and measuring the unidirectional or bidirectional throughput. These tests will be performed with open source tools available for different platforms, such as iperf[11] or ttcp[12].

The following chart describes an example about how the ttcp test could be executed and a final interpretation of the results.

| **Network performance Test (ttcp test – Ethernet/Wi-Fi)** |
|---|
| Requirements |
| Equipment required<br>*PC with Ethernet card / Wi-Fi card*<br>Software required<br>***ttcp*** *should be installed in both devices. Note that the test will run even if both devices use different operating systems.* |
| Procedure and configuration |
| Initial configuration<br>Both devices should be connected to the same domain within the ebbits network |

[11] http://sourceforge.net/projects/iperf/

[12] http://www.pcausa.com/Utilities/pcattcp.htm

Steps to implement the test

**Server side (Device B)**

Execute the command *pcattcp -r -c* from the PC connected to the same domain within the ebbits network that will run as server (receiver).

**Client side (Device A)**

Execute the command *pcattcp -t "ebbits DUT IP"* from the PC connected to the same domain within the ebbits network that will run as client (transmitter) once the receiver is already running.

Test results

The ttcp test results will provide information related to the throughput between the involved devices. The following example shows the output that a ttcp client provides:

**ttcp test client results**

```
PCAUSA Test TCP Utility V2.01.01.13 (IPv4/IPv6)
  IP Version  : IPv4
Started TCP Transmit Test 0...
TCP Transmit Test
  Transmit    : TCPv4 0.0.0.0 -> 192.168.15.112:5001
  Buffer Size : 8192; Alignment: 16384/0
  TCP_NODELAY : DISABLED (0)
  Connect     : Connected to 192.168.15.112:5001
  Send Mode   : Send Pattern; Number of Buffers: 2048
  Statistics  : TCPv4 0.0.0.0 -> 192.168.15.112:5001
16777216 bytes in 0.289 real seconds = 56650.01 KB/sec +++
numCalls: 2048; msec/call: 0.145; calls/sec: 7081.252
```

The previous results show the throughput value as a relation of the quantity of data sent in a period of time. The values obtained in this test serve as a reference to know if the device will be able to send all the necessary information and communicate to the ebbits platform efficiently. The throughput thresholds which determine if a device will be able to perform well; will be set according to the type of device and its exigencies.

## 7.2 Applications

In general, there are numerous dimensions of a testing approach and a continuum of values within each dimension:

- *When will testing be performed?* Will testing be done every day, as components are developed, or when all components are put together?
- *Who performs the testing?* Are developers responsible or are independent testers responsible?
- *Which pieces will be tested?* Will everything, a sample, or nothing be tested?
- *How will testing be performed?* Will testers have knowledge of only the specification of the component under test (blackbox testing) or also knowledge of implementation (whitebox testing)?
- *How much testing is adequate?* Will no testing be done or will exhaustive testing be done?

The choice of approach is dependent on the chosen development approach. The development approach in ebbits is iterative and incremental, comprising several complete cycles of analysis, development, and validation in which components from different partners are frequently integrated.

It is increasingly clear that the approach to testing in such conditions should preferably be "agile" or "lean". This means that ebbits should use automatic testing whenever possible. In addition "adequate" testing needs to be seen in the context of what is currently needed of the platform. For

demonstrators, e.g., the primary objective is to demonstrate partial functionality. Thus, focus is here not on doing a complete acceptance test (or even tests conforming to statement coverage), but rather on integration testing.

### 7.2.1 Testing Levels

Testing is typically considered to take place on four different levels:

1) Unit testing
2) Integration testing
3) System testing
4) Validation

#### Unit Testing

A "unit" in ebbits is a closed functional part. Unit tests must be automated and be written using a unit testing framework. In the case of Java this is JUnit[13]. In the case of .NET this is NUnit[14]. Upon completion of an increment of a unit, the following must be considered

- Code checked into the Subversion repository must not break the build process
- Prior to committing new versions of a unit to the Subversion repository there must be reasonable automated, functional unit tests. We do not prescribe any specific coverage criteria for blackbox or whitebox tests.

To enhance quality of the software it is recommended to create a new unit test for each detected bug if this was caused not by an error in the programming but in a misunderstanding of a requirement. After fixing the bug the commit statement has to contain the bug number so that an automated tracking of bug fixes can be performed.

#### Integration Testing

Integration testing takes place whenever multiple units need to work together. We do incremental integration (as opposed to integrating all units at once) but do not prescribe a specific approach such as bottom-up or top-down integration.

#### System Testing

On a system level, use case/high level functional requirement and non-functional/quality requirement testing can be tested. These tests will be automated whenever possible.

A best practice in integration and system testing is that the developers who wrote parts of the system/integrated unit should not write and perform the tests. Given that ebbits is a large, complex project it will almost always be the case that an integrator did not write part of the units being integrated and thus the integrator is permitted to write and perform the tests.

*Web Service Interoperability testing*

Interoperability between services is very important to ebbits. This is partly because the platform will be developed in different environments as well as on different platforms, but primarily because we want to have an open architecture which is SOA based. It should be easy to replace and extend services without using a specific programming platform as long as one adheres to the standards of Web Services.

The interoperability tests to be made on the ebbits application services can be divided into two main groups:
Testing done on WSDL files between components during development and integration.

[13] http://www.junit.org

[14] http://www.nunit.org

More formal testing of components using WS-I tools. These test both design time interoperability (based on a WSDL file) and run-time interoperability (whether the web services responds according to WS-I at run-time).

The Web Services Interoperability Organization (WS-I)[15] has developed testing tools that evaluate Web services conformance to Profiles. These tools test Web service implementations using a non-intrusive, black box approach. The tools focus is on the interaction between a Web service and user applications.

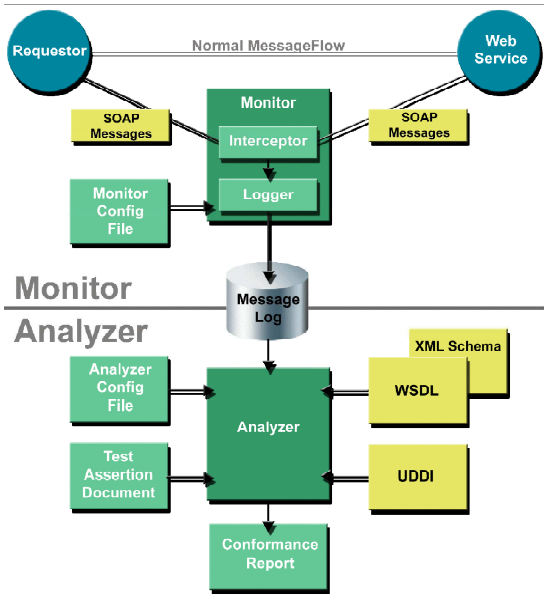


Figure 21 - Testing Tools Architecture

The testing infrastructure is comprised of the Monitor and the Analyzer and a variety of supporting files (see Figure 21):

- *Monitor* **-** This is both a message capture and logging tool. The interceptor captures the messages and the logger re-formats them and stores them for later analysis in the message log. The monitor is implemented using a man in the middle approach to intercept and record messages.
- *Analyzer* – This is an analysis tool that verifies the conformance of Web Services artefacts to Profiles. For example, it analyzes the messages sent to and from a Web service, after these have been stored in the message log by the Monitor.
- *Configuration Files* – These are XML files used to control the execution of the Testing Tools:
  *Monitor Configuration File* – controls the execution of the monitor
  *Analyzer Configuration File* – controls the execution of the analyzer
  *Test Assertion Document* – defines the test assertions that will be processed by the analyzer
- Other files or data artefacts will be accessed, which are not part of the test framework, but dependent on the Web Service to be tested:
- *Web Service artefacts* – these inputs to the Analyzer are target material for testing, and will be reported on:
  *Message Log* – contains the monitoring trace of messages captured at transport level.
  *WSDL definitions* - contains the definitions related to the Web Service
  *UDDI entries* - contains references to Web Service definitions, as well as bindings.
- *Generated Files* – These are XML files produced by Testing Tools, that are specific to the Web Service being tested: *Message Log* – (also a "Web Service artefact")
- *Conformance Report* – contains the complete conformance analysis from the specified inputs.

As the ebbits platform is based on different environments there may be initial interoperability problems with regard to the WSDL files and service invocations. In order to understand a

[15] http://www.ws-i.org

conformance report output one needs to understand that WS-I defines a number of assertions it checks. Figure 22 shows an example of an assertion.

Assertion: BP2416

| | |
|---|---|
| **Result** | **Passed** |
| **Assertion Description** | Every QName in the WSDL document that is not referring to a schema component, is either using the target namespace of this WSDL or the target namespace of a directly imported WSDL component. |

Figure 22 - Sample WS-I assertion

To find explanations for all assertions, see the WS-I webpage

# 8. Conclusions

Communication networks and all relevant components constitute an essential part of the ebbits platform. In fact, ebbits operations are based on the availability at lower level of diverse heterogeneous networks and devices being integrated into the ebbits platform. To enable this resulting scenario, network connectivity to all the ebbits components, including networking devices, ebbits platform core components and physical world objects, should be provided.

To handle the considered heterogeneity efficiently, the network infrastructure considered for field trials should be integrated in the most simple way possible in order to support all considered devices from both scenarios and at the same time be able to debug any issue that can appear during the process of providing IP connectivity.

The selected communication protocols and network components will provide the basic network structure to run the applications defined for each field trial, thus enabling the demonstration of ebbits features in real world scenarios. The characteristics of the identified field trials could also provide an idea of potential issues the network infrastructure could experience or suggest which devices could require more attention during the integration process.

Both integration and testing plans are using incremental approaches. In fact, the iterative process driving the project and the experience acquired on the first prototypes would allow to properly adjust the different steps considered in both integration and testing processes.

The network infrastructure testing plan will be in charge of describing the different tests needed to check whether the network connectivity is provided to all the devices being integrated into the ebbits platform. The software tools chosen to perform the different tests have been selected among several non commercial solutions. Some of the tools are installed by default in most of the operating systems available today (e.g. ping).

Testing of end-to-end business applications will follow an approach that depends on the iterative and incremental development approach used within the project. For such reason, the testing should preferably be "agile" or "lean", meaning that automatic testing should be considered whenever possible.

It is worth mentioning that M34 and M46 prototypes are less detailed compared to M10 and M22 prototypes, since they will be better defined based on results and lessons learnt during the first two iterations of the ebbits project.

# 9. List of Figures

# 10. References

(Bruegge et al 2000) Bruegge, B., Dutoit A.H. (2000). *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Prentice Hall.

(Dean 2010) Dean T. (2010). Network+ Guide to Networks, Fifth Edition. Course Technology, Cengage Learning.

(ISO 2000) ISO (2000). ISO 11788-3:2000, Electronic data interchange between information systems in agriculture; Agricultural data element dictionary: Pig farming. ISO.

(Sommerville 2001) Sommerville (2001). Software engineering (6th edition). McGraw-Hill.

(Ommering 2003) Ommering, R. (2003). *Configuration Management in Component Based Product Populations*. In: Westfechtel, B. (ed.): Software Configuration Management, Springer.

(Westfechtel et al 2003) Westfechtel B., Conradi R. (2003). *Software Architecture and Software Configuration Management*. In: Westfechtel, B.; Hoek, A. v. d. (eds.); Software Configuration Management. Springer.