


Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	



*Deliverable D44.2*  
*Mobile Business platform*  
*specifications and architecture*  
*M21 issue*

Document Owner:	Giovanni Casella – SOFTECO
Contributors:	Enrico Morten, Simona Stringa, Paolo Giovene SOFTECO, Mauro Isaja ENG, Charalampos Vassiliou SL
Dissemination:	Public
Contributing to:	WP 44
Date:	28/06/2013
Revision:	1.0

## VERSION HISTORY

	DATE	NOTES AND COMMENTS
<b>0.1</b>	<b>03/06/2013</b>	<b>DELIVERABLE STRUCTURE REVISED</b>
<b>0.2</b>	<b>04/06/2013</b>	<b>SECTION 1 REVISED</b>
<b>0.3</b>	<b>05/06/2013</b>	<b>SECTION 3, 4 AND 5 UPDATED (TECHNICAL ROADMAP ADDED)</b>
<b>0.4</b>	<b>06/06/2013</b>	<b>SECTION 6 EXTENDED</b>
<b>0.5</b>	<b>10/06/2013</b>	<b>SECTION 7 REVISED</b>
<b>0.8</b>	<b>13/06/2013</b>	<b>SOFTECO INTERNAL REVIEW – DOCUMENT REVISION</b>
<b>0.9</b>	<b>24/06/2013</b>	<b>REVISION ACCORDING MSEE PEER REVIEW COMMENTS</b>
<b>1.0</b>	<b>25/06/2013</b>	<b>FINAL VERSION</b>

## DELIVERABLE PEER REVIEW SUMMARY

ID	Comments	Addressed ( ✓ ) Answered (A)
1	Each index must be in a different page (ToC, Figures, Tables)	✓
2	In the section 2.2. the different mobile operating systems are described. What about the Firefox OS?	✓
3	Do not use Wikipedia as a valid reference	✓
4	Many figures, diagrams and tables not listed along the deliverable	✓
5	A table in the conclusions section?	Moved in Section 5.

## TABLE OF CONTENTS

<b>EXECUTIVE SUMMARY</b>	<b>6</b>
<b>1. INTRODUCTION</b>	<b>8</b>
1.1. <i>Objective of the Deliverable</i>	8
1.2. <i>Structure of the Deliverable</i>	9
<b>2. MOBILE DEVICES AND OPERATING SYSTEMS</b>	<b>10</b>
2.1. <i>Mobile Devices</i>	10
2.2. <i>Mobile Operating Systems</i>	12
<b>3. RELEVANT TECHNOLOGIES</b>	<b>15</b>
<b>4. THE ROLE OF THE MOBILE BUSINESS PLATFORM IN MSEE</b>	<b>18</b>
<b>5. MOBILE BUSINESS PLATFORM</b>	<b>22</b>
<b>6. MOBILE BUSINESS PLATFORM MODULES SPECIFICATION</b>	<b>27</b>
6.1. <i>Mobile Development Module</i>	27
6.2. <i>Mobile Delivery Module</i>	33
6.3. <i>Ambient Intelligence Module</i>	38
6.4. <i>Mobile Collaboration Module</i>	50
6.5. <i>Multimodal Module</i>	56
<b>7. CONCLUSIONS</b>	<b>70</b>

## LIST OF FIGURES

Figure 1: Relevant MSEE deliverables for this document .....	8
Figure 2: Top-down approach used to structure the deliverable .....	9
Figure 3: Cell phones.....	10
Figure 4: PDAs .....	11
Figure 5: Smartphones.....	11
Figure 6: Tablets .....	12
Figure 7: App Store and Android Market.....	14
Figure 8: The MSEE Service System Life Cycle .....	18
Figure 9: Transitions among the servitization levels.....	19
Figure 10: MSEE IT System .....	19
Figure 11: The Service System Life Cycle.....	20
Figure 12: Mobile Platform Modules .....	22
Figure 13: Mobile Business Platform Technical Roadmap.....	25
Figure 14: REST Services and Mobile Applications.....	28
Figure 15: Mobile Development code generation process .....	29
Figure 16: Mobile Development Module .....	30
Figure 17: FI-WARE SDF high level architecture .....	33
Figure 18: Mobile Delivery Deployment Diagram .....	34
Figure 19: Mobile Delivery Module.....	34
Figure 20: Mobile Delivery search sequence diagram .....	35
Figure 21: Home and Manufacturing Environments .....	38
Figure 22: User, Environments and Devices .....	41
Figure 23: AMI Module .....	42
Figure 24: Ambient Intelligence device identification sequence diagram .....	44
Figure 25: Barcode .....	45
Figure 26: QRCode.....	46
Figure 27: IEC access from a mobile device .....	51
Figure 28: Mobile Collaboration Module Architecture.....	52
Figure 29: A representation of multimodal man machine interaction loop.....	57
Figure 30: Application modes. ....	59
Figure 31: Media button control.....	60
Figure 32: UI model. ....	61
Figure 33: Multimodal Module Internal Architecture .....	65
Figure 34: Application and environment sensing.....	69

## LIST OF TABLES

Table 1: Report on mobile operating systems diffusion.....	13
Table 2: Mapping of the MSEE IT Platforms with the phases of the Service System Life Cycle.....	21
Table 3: List of Mobile Platform modules .....	24
Table 4: Ambient Intelligence module requirements .....	40
Table 5: Mobile devices relevant features for multimodal applications .....	58
Table 6: Multimodal input UI components .....	63
Table 7: Multimodal output UI components .....	64

## Executive Summary

This document describes the final specification and architecture of the MSEE Mobile Business Platform (Mobile Platform) resulting from the work done in WP44 “MSEE Generic Mobile Business Platform”.

The task T44.1 of WP44 “Support to Mobile and Ubiquitous business” focuses on the exploitation of the ubiquitous computing environment provided by mobile devices in order to support business scenarios with personalised services utilized by users in mobility, and in order to support industrial model for Innovation Ecosystems enhancing mobile collaboration. To achieve the goal of T44.1 the Mobile Platform provides mobile extension to other MSEE platforms namely the MSEE Generic Development Platform (WP42), the MSEE Generic Service Delivery Platform (WP43) and the Innovation Ecosystems Platform (WP26). The extensions are provided through the implementation of three modules namely Mobile Development, Mobile Delivery and Mobile Collaboration. In addition to extending other platforms, the Mobile Platform addresses two relevant topics for mobile devices namely Multimodal Interaction and Ambient Intelligence.

The task T44.2 focuses “on the study and the implementation of new multi-modal user-interaction models, based on adaptive interfaces, which may use visual and speech communication, natural-language recognition of free text, speech or gestures” [MSEE DoW<sup>1</sup>]. The Mobile Platform addresses this topic supporting the development of multimodal mobile applications through a software library representing the Mobile Platform Multimodal module. In MSEE this module is tested and demonstrated through the development of a concrete mobile application to be used in the manufacturing context. One of the main goals of the application is to assess the advantages of a multimodal interface when used by professionals who need to perform at the same time multiple tasks in a dynamic and unfriendly environment (i.e. noisy, brightly, dangerous, etc.).

Finally the task T44.3 addresses the innovative scenarios enabled by the Ambient Intelligence issue in the context of MSEE use cases and end-users. The Ambient Intelligence module of the Mobile Platform exploits the engaging features of smartphones and tablets in order to discover and identify the available devices (machines, sensors, etc.) in the user environment providing additional functionalities for these devices. The module also provides a simple extension mechanism enabling developers to extend it in order to address domain specific requirements.

The deliverable is organized as follows:

Section 1 – Introduction: clarifies the objective of the deliverable, its internal structure and the relations with other MSEE deliverables relevant for this document.


Section 2 – Mobile Devices and Operating Systems: provides some background information on mobile devices (cell phones, PDAs, smartphones, tablets) and their operating systems.

Section 3 – Relevant Technologies: presents, very briefly, the technologies used to develop the Mobile Platform in order to simplify the reading of the subsequent sections.

Section 4 – The role of the Mobile Business Platform in MSEE: specifies the role of the Mobile Platform in the context of the Service System Life Cycle composed by the “servitization process” and by the “governance and innovation process” and highlights the position of the Mobile Platform in the MSEE IT System<sup>2</sup>.

<sup>1</sup> MSEE DOW: Manufacturing Service Ecosystem Description of Work

<sup>2</sup> MSEE\_D41.1: MSEE Service-System Functional and Modular Architecture

Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	

Section 5 – Mobile Business Platform: presents the general architecture of the platform and introduces its modules.

Section 6 – Mobile Business Platform Modules Specification: depicts the internal architecture of each module and presents its specification.

Section 7 – Conclusions: summarizes the results presented in the document.

Respect to the first version of this document (D44.1) Sections 1 and 7 (introduction and conclusions) have been revised while Sections 3 and 4 (presenting relevant technologies and the role of mobile platform in MSEE) have been updated. Section 5 and 6 presenting the specification of the Mobile Platform modules have been updated and extended. Section 5.1 presents the technical roadmap of the Mobile Platform.

The initial version of this document focused mainly on the Mobile Development, Mobile Delivery and Ambient Intelligence modules which were implemented and released with the first prototype (D44.3 due at M18). The design of these modules included in D44.1 was appropriate and major changes were not required.

The main contribution of this document regards the Mobile Collaboration and the Multimodal modules. The design of these modules has been completed and refined in order to develop and release them as part of the final version of the Mobile Platform (D44.4 due at M30). The final version of the Mobile Platform will also include some extensions, which not require design changes, of the modules released with the first prototype.

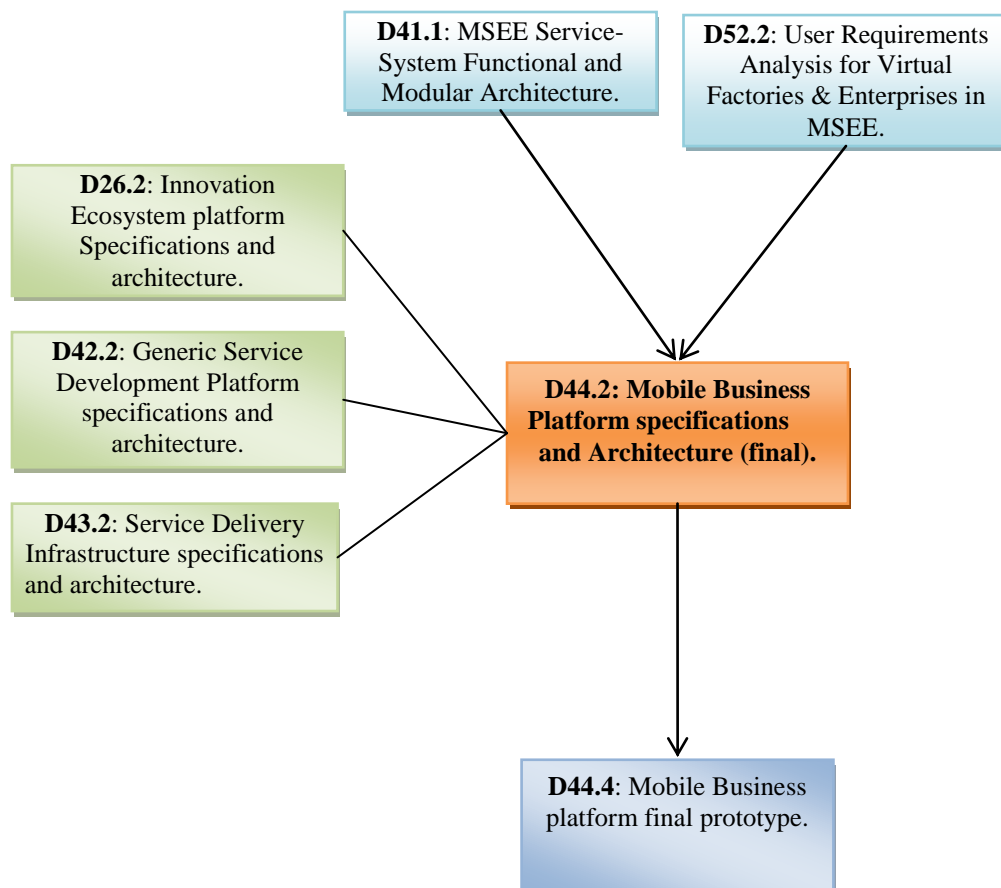
## 1. Introduction

### 1.1. Objective of the Deliverable

The main objective of this deliverable is to provide the final specification of the MSEE Mobile Business Platform:

- Providing a short background on mobile devices and related technologies and providing some links to get further information.
- Highlighting the role of the platform in the MSEE project and describing the relations with the others platforms developed in the project.
- Presenting the Mobile Platform final architecture and specification.

The document presents the results obtained in the WP44 of MSEE namely “MSEE Generic Mobile Business Platform” and in particular in the three work package tasks T4.4.1 “Support to Mobile and Ubiquitous business”, T4.4.2 “Multimodal Interaction” and T4.4.3 “Ambient Intelligence”.



**Figure 1: Relevant MSEE deliverables for this document**

Figure 1 presents the MSEE deliverables with a particular relevance for this document. The Mobile Platform is part of the MSEE IT System presented in *D41.1* and address the user requirements collected in *D52.2*.

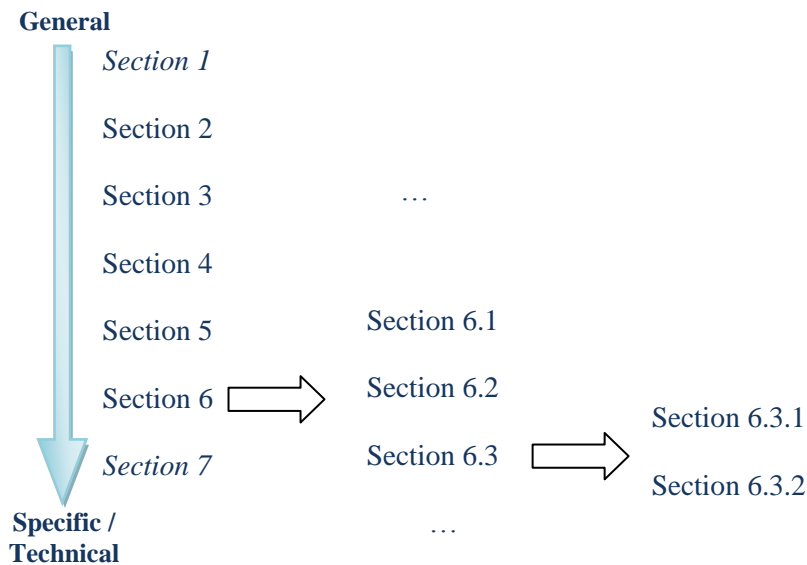
The Mobile Platform focuses on mobile devices extending the other MSEE platforms composing the MSEE IT System, whose specification is provided in the deliverables *D26.2*, *D42.2* and *D43.2*.



The architecture and specification presented in this document will be used to develop the Mobile Platform final prototype (*D44.4*).

## 1.2. Structure of the Deliverable

To improve readability, the document presents the information regarding the Mobile Platform following a top-down approach both for the overall document and inside each section. Firstly general information is presented, and afterwards specific information and/or technical details are provided.



**Figure 2: Top-down approach used to structure the deliverable**

Figure 2 shows a graphical representation of this approach. While Section 2 and 3 includes some general background information on mobile devices and mobile relevant technologies Section 4 depicts the role of the Mobile Platform in MSEE and Section 5 and 6 present the platform specification and architecture.

The top-down approach is used also in each Section.

For example Section 6 presents the specification of the Mobile Platform modules, Section 6.3 presents the details of a particular module and Section 6.3.1 includes some technical information on a specific aspect of the module.

Having simpler contents, Section 1 “Introduction” and Section 7 “Conclusions” do not follow the top-down approach.

## 2. Mobile Devices and Operating Systems

### 2.1. Mobile Devices

In the last years the mobile device market has shown an exponential growth and new technologies have generated a great excitement. In addition to the original cell phones today several devices are available such as PDAs, smartphones, tablets, notebooks, laptops, handheld game console, portable media player, personal navigation device and so on.

Some of these mobile devices (i.e. game console, portable media player, personal navigation) are out of the scope of the Mobile Platform. Also laptops and notebooks will not be addressed by the Mobile Platform because they are able to execute applications designed for Personal Computer so they can use functionalities provided by the other MSEE platforms.

In the rest of the Section will be presented some candidate device categories to be addressed by the Mobile Platform while the precise set of addressed devices is defined in the last part.

#### Cell Phones

The first models of cell phones (Figure 3) were developed during the first half of the 20<sup>th</sup> century. They were very expensive devices, weighting more than 30 kilos, and their use was limited to demonstrations and to military uses during the Second World War. The unique purpose of these phones was to send and receive voice communications.

Modern cell phones were developed around 1990. They were quite expensive small devices equipped with an alpha numeric display where the main feature was again to send and receive voice communications. However the rapid expansion of the mobile phone market promoted the addition of new features.

Today cell phones can be small or medium sized and are equipped with a graphical color display and a numeric physical keyboard (sometime a QWERTY physical keyboard). New functionalities have been added like the possibility to send and receive text and multimedia messages (SMS and MMS) and to capture low resolution photo. A predefined set of applications (i.e. phone book, calculator, convertor, games, alarm, etc.) is pre-installed on these devices. To find and install new applications is quite complex.



Figure 3: Cell phones

#### Personal Digital Assistant (PDA)

PDAs (Figure 4) are medium sized devices usually equipped with a medium/large graphical display. Their main purpose of PDAs is to help the user in the management of personal information like addresses, contacts, task lists, appointments, meetings, reminders, etc.

The main PDA input method is usually a resistive touch screen to be used with a stylus. While some PDAs are equipped with a physical QWERTY keyboard the major part are equipped with a virtual QWERTY keyboard.

Traditionally PDAs did not provide communication features like telephone/fax, Internet and networking. While cell phones are still on the market due to their price (today they are quite cheap) and features (they are usually small and simple to use) PDAs have been largely replaced by smartphones.



**Figure 4: PDAs**

PDAs are equipped with a predefined set of applications that it is difficult to expand while it is easy to synchronize the PDA contents (calendar, address book, etc.) with the contents stored on a Personal Computer.

### Smartphone

Smartphones (Figure 5) can be considered a combination of cell phones and PDAs. In fact they are able to manage personal information and to provide GSM (Global System for Mobile Communications) communication capabilities.

Today the main focus is on smartphones (and on tablet presented later) and everyday new models (with new features) appear on the market.

Smartphones are pocket sized devices bigger than a cell phone and similar to PDAs. Their display is usually a high resolution graphical display supporting multi touch (even if in some cases they are still equipped with a qwerty physical keyboard). The display is not resistive (as in PDA requiring a stylus) but capacitive enabling a comfortable and precise use using only fingers.

Smartphones are equipped with powerful CPUs (over 1Ghz of speed, sometime dual-core or quad-core) and have a large storage space (some gigabytes). They are able to detect their precise location using the Global Position System and to sense the ambient through several sensors (luminosity sensor, accelerometer, digital compass). Finally smartphones are able to guarantee high speed internet access and other communication channels like Wi-Fi and Bluetooth.



**Figure 5: Smartphones**

Smartphones are sold with a large pre-defined set of applications easy to expand. Finding new applications is fast and simple thanks to the availability of dedicated virtual marketplaces of mobile applications.

### Tablet

Tablets (Figure 6) are portable devices quite big (they are usually equipped with a 7'-10' display) in comparison to smartphones and cell phones.



**Figure 6: Tablets**

They have a large multi-touch capacitive touch screen and are able to execute both applications implemented for smartphones and applications specifically developed for tablets able to fully exploit tablet's features (in particular their large screen) and ensuring an improved user experience. As smartphones, tablets have many communication capabilities (GSM, Bluetooth, Wi-Fi), support GPS, are equipped with several sensors and are sold including many applications. It is possible to add new applications easily exploiting available mobile application marketplaces.

### **Devices addressed by the Mobile Platform**

Due to their engaging features and diffusion the main categories of mobile devices addressed by the MSEE mobile platform are smartphones and tablets. They are high technology devices, sold at an accessible price, including state of the art technologies in the mobile device field. Suitable operating systems and powerful programming languages exist for such devices and users are able to easy find new applications and to install them in a user-friendly way. Finally they are equipped with powerful CPU and have a large storage space.

## **2.2. Mobile Operating Systems**

Today the general feeling is that the mobile operating systems provided by Google (leading the Open Handset Alliance) and Apple are dominating the market and this feeling is confirmed, among others, by a recent press release published by comScore<sup>3</sup> in March 2013 (see Table 1) highlighting the most spread mobile operating systems. It is interesting to compare this table with the table included in the previous version of this document (referring to the period January-March 2012). While the overall ranking has not changed, Google and Apple have further strengthened their position at the expense of other systems.

<sup>3</sup> [http://www.comscore.com/About\\_comScore](http://www.comscore.com/About_comScore)

Top Smartphone Platforms Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Dec-12	Mar-13	Point Change
<b>Google</b>	53.4%	52.0%	-1.4
<b>Apple</b>	36.3%	39.0%	2.7
<b>RIM</b>	6.4%	5.2%	-1.2
<b>Microsoft</b>	2.9%	3.0%	0.1
<b>Symbian</b>	0.6%	0.5%	-0.1

**Table 1: Report on mobile operating systems diffusion<sup>4</sup>**

The Open Handset Alliance, supported by Google, provides the Android platform, a Linux-based operating system for mobile devices (i.e. smartphones and tablet computers) of different manufacturers. Android applications are written in Java (a customized version) and several tools and libraries are available. Android makes easy to write engaging applications through the provision of powerful APIs with different goals like handset layouts management, internal storage use, connectivity management (bluetooth, Wi-Fi, etc.), access to phone messaging and calls, media support (audio/video), multi-touch support, etc.

Apple provides the IOS platform equipping Apple devices like iPhone and iPad. iOS applications are written in Objective-C using the Cocoa framework. Even for iOS a powerful development environment is available, namely xcode, as well a complete API and a set of additional libraries.

RIM provides the operating system (a proprietary multitasking environment) equipping worldwide known BlackBerry devices.

Microsoft provides the Windows Phone operating system (the successor of the Windows Mobile platform) and is trying to conquer the mobile market through an alliance with Nokia. However Windows Phone operating system, launched in the second half of 2010, has still a small diffusion (3%).

Symbian, that today is not very diffused, was the most advanced and used operating system before the comparison of smartphones and equipped millions of Nokia cell phones.

Finally in the 2011 was announced by the Mozilla Corporation the "Boot to Gecko" project that in the 2012 was rebranded as "Firefox OS". The new mobile operating system has the goal of supporting the development of HTML5 apps written using open web technologies rather than platform-specific native APIs (like other mentioned operating systems). While Firefox OS has not still gained a relevant market share it has a big potential if it will be able to satisfy both developers and customers' requirements.

Android and iOS have defeated the competitors thanks to their unique features and in particular to the features dedicated to the installation of new applications.

In fact users are able to access to online markets (i.e. App Store and Android market shown in Figure 7) directly by their phones and they can easily choose an application (from the thousands of free and not-free applications available) or a game to install on it. The download and installation of the applications is really simple while developers are fully supported in terms of tools and documentation.





**Figure 7: App Store and Android Market**

The main difference between iOS and Android is that iOS is installed only on Apple devices, while Android is installed on the devices of many brands like HTC, Samsung, LG, Sony etc. It is clear that the iOS diffusion is related to the diffusion of Apple products while the diffusion of Android is not related to the products of a specific brand. Moreover while Apple products are quite expensive other brands sell expensive devices but also cheap ones.

Another key difference between Android and iOS is that while Android is open source (supported by the Open handset Alliance) and it is possible to develop an Android application using a generic personal computer, iOS is an Apple proprietary system and a MAC computer is required to develop an iOS application.

In most cases the functionalities of the Mobile Platform will be available to smartphones and tablets regardless of their operating systems while when a strictly interaction between the platform and the mobile device hardware will be required, a native application, specific operating system dependent, will be developed. In this latter case the addressed operating system will be Android mainly due to its diffusion but also to its open source nature.

### 3. Relevant Technologies

---

This Section provides some general information on the relevant technologies for the Mobile Platform. While the descriptions are short and concise a web site is indicated for each technology where the reader can find additional details if needed.

#### **Ajax<sup>5</sup>**

AJAX is the acronym of Asynchronous JavaScript and XML and indicates a particular technique to exchange data with a server enabling to update parts of a (mobile) web page without reloading the whole page. In practice Javascript is used, on the browser, to interact with the server and to update the web page shown to the user while XML is used as data format to exchange data between the server and the browser.

#### **Android<sup>6</sup>**

Android is an open source operating system for mobile devices installed on many mobile devices of almost all manufacturers (the Android presence on the market is increasingly constantly). The Android features, the presence of a vibrant community around the technology, the availability of documentation (books, developer manuals, tutorials, videos, etc.) and the availability of the Android application market enable developers to produce engaging applications that users can easily find and download.

The Android ADT is a powerful Eclipse plug-in supporting developers to implement mobile applications.

#### **Apache HttpComponents<sup>7</sup>**

The Apache HttpComponents library is provided by Apache with the goal of simplifying Java based developments exploiting the HTTP protocol.

Designed for extension while providing robust support for the base HTTP protocol, the HttpComponents library is of interest for building HTTP-aware client and server applications such as web browsers, web spiders, HTTP proxies, web service transport libraries, or systems that leverage or extend the HTTP protocol for distributed communication.

#### **Apache Jersey<sup>8</sup>**

Apache Jersey is an open source framework supporting developers in the provision of REST web services and representing the reference implementation of the JAX-RS (JSR 311) specification. The specification defines a set of Java APIs for the development of Web services built according to the Representational State Transfer architectural style.

#### **Eclipse IDE<sup>9</sup>**

Eclipse is a powerful and extensible integrated development environment, including runtimes and application frameworks, for building, deploying and managing software across the entire software lifecycle.

#### **HTML5<sup>10</sup>**

---


<sup>5</sup> <http://www.w3schools.com/ajax/default.asp>

<sup>6</sup> [www.android.com](http://www.android.com)

<sup>7</sup> <http://hc.apache.org/>

<sup>8</sup> <http://jersey.java.net/>

<sup>9</sup> <http://www.eclipse.org>

Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	

HTML is a markup language for structuring and presenting content for the World Wide Web and HTML5 represents the fifth revision of the HTML standard (created in 1990 and standardized as HTML4 as of 1997). HTML5 is still under development (as August 2012) and its goal is to improve HTML4 supporting the latest multimedia formats as well as location services while keeping it easily readable by humans and consistently understood by computers and devices.

## **Javascript<sup>11</sup>**

JavaScript is a lightweight scripting language supported by all major browsers. Usually Javascript is embedded in HTML pages and using it is possible to ask the browser to perform some actions like to check if a required form field has been left empty.

## **JQuery Mobile<sup>12</sup>**

The JQuery Mobile framework is designed to support developers in the implementation of web sites, designed to be used by mobile devices regardless by device features (screen dimension and density, available input modes, etc.) and device operating systems.

JQuery Mobile provides a unified user interface system that works seamlessly across all popular mobile device platforms.

The framework includes an Ajax navigation system that brings animated page transitions and a core set of User Interface widgets.

## **JSON<sup>13</sup>**

JSON (JavaScript Object Notation) is a lightweight textual data-interchange format that is easy to read and write for humans and machines. Although JSON is based on a subset of the JavaScript Programming Language it is completely independent by any programming language or operating system. JSON is widely used to represent resources exchanged by REST web services.

## **REST<sup>14</sup>**

REST is the acronym of REpresentational State Transfer and denotes a particular type of software architecture for distributed systems (client/server) adopted by the World Wide Web. In a REST architecture clients send requests to servers that return an appropriate resource. In the World Wide Web the client role is played by the browser (IE, Firefox, etc.) and the requested resource is usually a web page.

In general the client role can be played by a desktop application, a web application or a mobile application and a resource is a representation of a business object of interest in a given domain.

A RESTful web service (also called a RESTful web API) is a web service implemented using HTTP and the principles of REST.

<sup>10</sup> <http://www.whatwg.org/specs/web-apps/current-work/>


<sup>11</sup> [http://www.w3schools.com/js/js\\_intro.asp](http://www.w3schools.com/js/js_intro.asp)

<sup>12</sup> <http://jquerymobile.com>

<sup>13</sup> <http://www.json.org/>

<sup>14</sup> [http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)



Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	

## XML<sup>15</sup>

XML is the acronym of Extensible Markup Language (XML) and denotes a textual markup language defining a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity, generality, and usability over the Internet.

Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

## XMI<sup>16</sup>

The XML Metadata Interchange (XMI) is an Object Management Group standard for exchanging metadata information via Extensible Markup Language. It can be used for any metadata whose meta-model can be expressed in the Meta-Object Facility language.

The most common use of XMI is as an interchange format for UML models, although it can also be used for the serialization of models of other languages.

<sup>15</sup> <http://www.omg.org/spec/XML/>

<sup>16</sup> <http://www.omg.org/spec/XMI/>

#### 4. The role of the Mobile Business Platform in MSEE

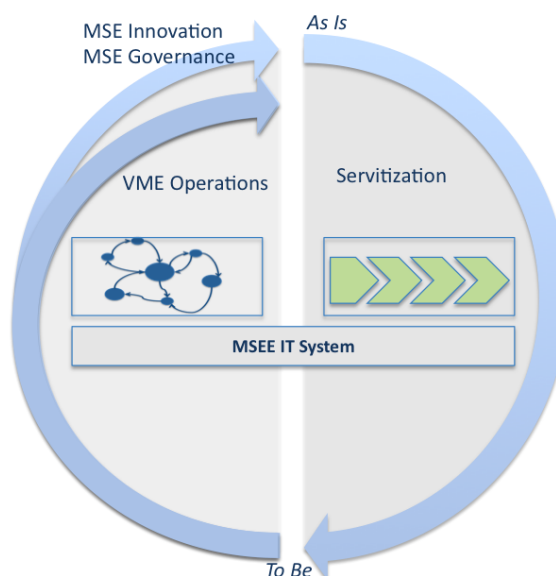
MSEE supports manufacturing enterprises to go from product-centric offerings to a value-proposition based on product-service bundles. This is a complex process that MSEE supports proposing a structured approach, in order to mitigate the risks and maximize the opportunities, and providing suitable models, methods and tools.

The MSEE IT System<sup>17</sup> is an integrated IT system which will support the whole life cycle of a manufacturing service ecosystem, including the servitization process, and specifically aiming at developing, operating and governing the target service ecosystem.

The scope of the MSEE IT System must be considered in the context of the overall Service System Life Cycle (see Figure 8) that MSEE aims at supporting and guiding in the manufacturing industry.

The Service System Life Cycle is composed by two main processes:

- The servitization process is responsible for the design of the Virtual Manufacturing Enterprise (VME) in order to switch the value offered from physical products to product-service “bundles”.
- The governance, operations and innovation process, on one side allows the Manufacturing Service Ecosystem governance and on the other side delivers the new value proposition to the customer through collaborative processes (VME Operations).



**Figure 8: The MSEE Service System Life Cycle**

The MSEE IT System, while taking into account the whole Service System Life Cycle, has a specific focus on the servitization process that will be supported by an integrated system composed by three main platforms: the Development Platform<sup>18</sup>, the Delivery Platform<sup>19</sup> and the Mobile Platform.

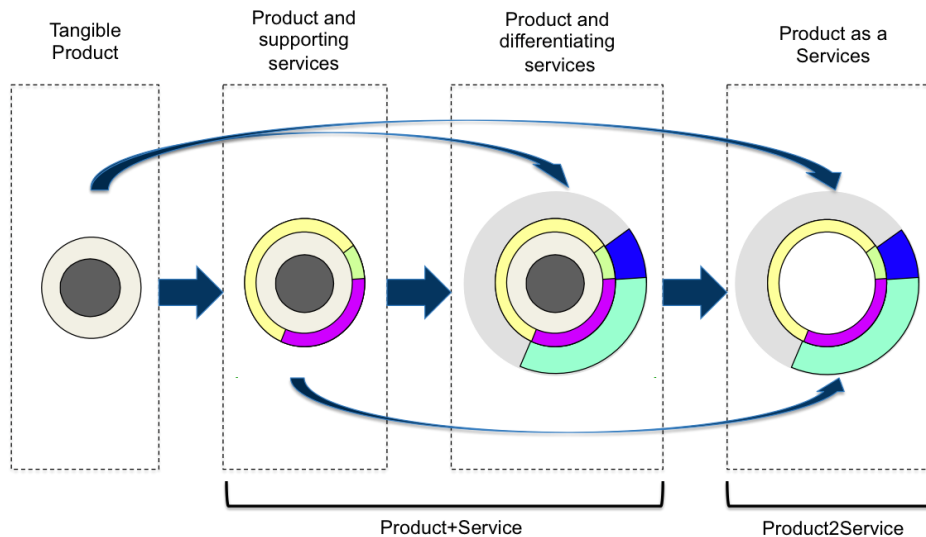
The servitization process<sup>20</sup> starts when an enterprise decides to re-purpose its value offer, shifting from its current servitization-level to a higher one (see Figure 9).

<sup>17</sup> MSEE\_D41.1: MSEE Service-System Functional and Modular Architecture.

<sup>18</sup> MSEE\_D42.1: Generic Service Development Platform specifications and architecture.

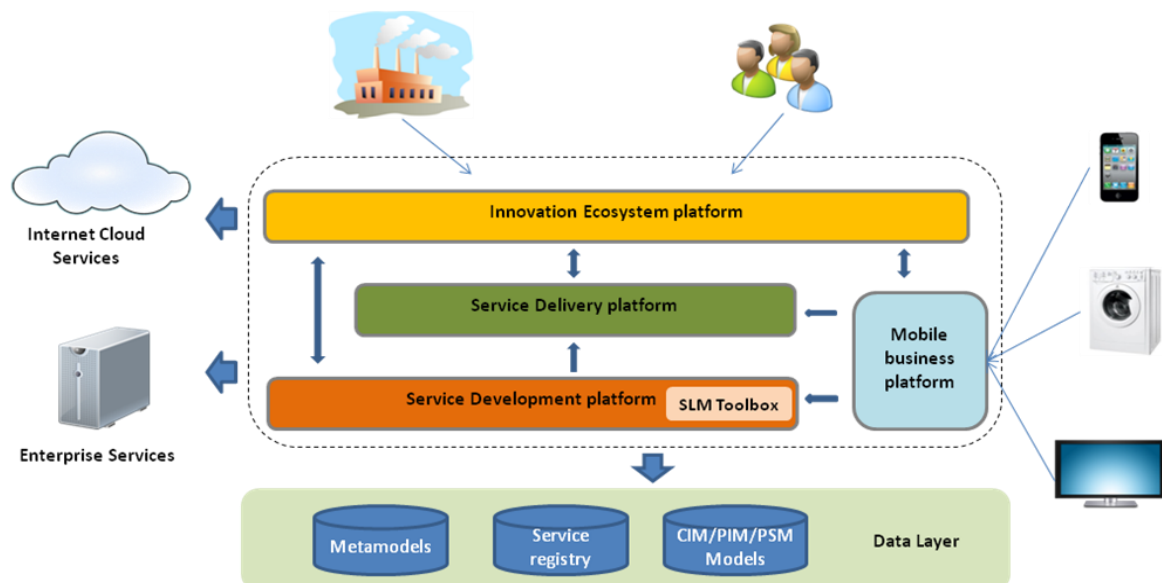
<sup>19</sup> MSEE\_D43.1: Service Delivery Infrastructure specifications and architecture.

<sup>20</sup> MSEE\_D11.1: Service concepts, models and method: Model Driven Service Engineering.



**Figure 9: Transitions among the servitization levels**

The more an enterprise increases its servitization-level, the more its value proposition is decoupled from the physical product.  
Figure 10 shows a high level overview of the MSEE IT system from the point of view of its main components.



**Figure 10: MSEE IT System**

The Service Development Platform provides tools for business modeling and service lifecycle management, through the SLM Toolbox, and for the technical development of IT services and applications.

The Service Delivery Platform is in charge of managing service registration and delivery as well as semantic search and invocation.

The Innovation Ecosystem Platform<sup>21</sup> (IEP) supports both the ecosystem governance and the service innovation process interacting with all the other platforms. Moreover it is also the access point to MSEE IT System for the final users (people and enterprises in the ecosystem).

<sup>21</sup> MSEE\_D26.1: Innovation Ecosystem Platform specifications and architecture.

In this context the role of the Mobile Platform is to exploit the exciting and continuously increasing capabilities of mobile devices to provide a full set of functionalities supporting Manufacturing Innovation Ecosystems and enabling mobile users to achieve the benefits provided by MSEE.

The Mobile Platform extends other platforms (Development Platform, Delivery Platform, Innovation Ecosystem Platform) functionalities addressing mobile devices in different contexts.

In particular the goals of the mobile platform are:

- to enhance the mobile collaboration inside ecosystems (IEP extension)
- to support developers in the implementation of mobile applications based on back-end services extending the functionalities provided by the Development Platform (Development Platform extension)
- to enable business users, involved in the definition of new services, to find already delivered services and applications providing a mobile access to the Delivery Platform (Delivery Platform extension)
- to enable service providers to monitor their services from mobile devices (IEP and Delivery Platform extension)

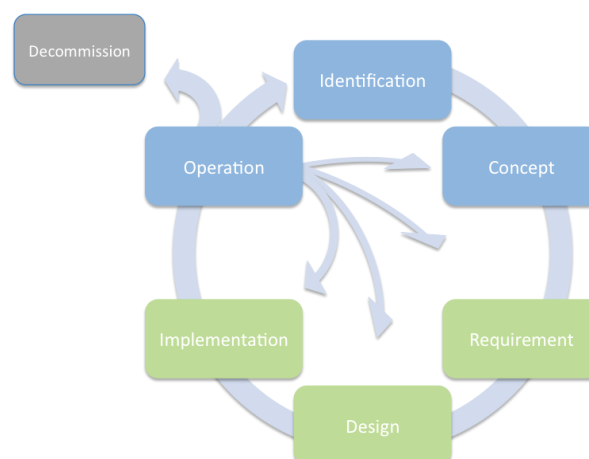
The Mobile Platform also considers two additional relevant topics for mobile devices namely Ambient Intelligence and Multimodal Interaction, as described in the MSEE Description of Work. These aspects are relevant for mobile devices for different reasons:

Ambient Intelligence refers to (electronic) environments that are sensitive and responsive to the presence of people and are able to interact with them being aware of their personality and needs, desires. Humans almost always carry on with them their mobile devices and the innovative features of these devices make them the ideal candidates to play the role of “universal human interface” in the Ambient Intelligence scenarios.

Multimodal interfaces can improve the user experience when using mobile applications. In fact applications developed for smartphones have some constraints mainly due to the small size of the devices limiting the available space for screens (output) and virtual keyboards (input). The impact of these constraints increases if the users are involved at the same time in multiple activities and works in changing environments like the manufactory ones.

Multimodal interfaces exploit nontraditional adaptive input/output modes (such as speech, touch, gestures, body movements...), some time in combination, to enhance human-computer interactions and in particular “human-mobile device” interactions.

The MSEE project reuses the generic entity/system life cycle phases defined in the ISO 15704 standard to identify a reference and generic Service System Life Cycle. Figure 11 gives a graphical representation of such cycle.



**Figure 11: The Service System Life Cycle**

The phases of the Service System Life Cycle are the following:

- **Service System Identification:** identify domain and existing components, objectives, challenges for a transition from product to service (or product + service);
- **Service System Concept:** identify and define the main concepts (models, functions, values, etc.) to create services around a product.
- **Service System Requirement:** identify, describe and model end-users requirements for the service system.
- **Service System Design:** design, specify and simulate the system that will provide the service.
- **Service System Implementation:** describe how the designed service system will be realized, deliver and implement physically all the needed components.
- **Service System Operation:** the service system is operational for use by customers and by all the stakeholders. This includes service consumption and interaction with customers, monitoring, evaluation and maintenance.
- **Service System Decommission:** end of the service system. The service system is removed and dismissed, whereas its components are reused.

Table 2 shows a high-level mapping of the MSEE Platforms with the phases of the Service System Life Cycle.

	Identification	Concept	Requirement	Design	Implementation	Operation	Decommission
Generic Service Development Platform (includes SLM ToolBox)							
Generic Service Delivery Platform							
<b>Generic Business Mobile Business Platform</b>							
Innovation Ecosystem Platform							

**Table 2: Mapping of the MSEE IT Platforms with the phases of the Service System Life Cycle**

The Mobile Platform mainly addresses the implementation and operation phases where mobile devices can play an important role. Secondly the Mobile Platform supports the Identification, Concept and Requirement phases through the extension of the IEP platform.

## 5. Mobile Business Platform

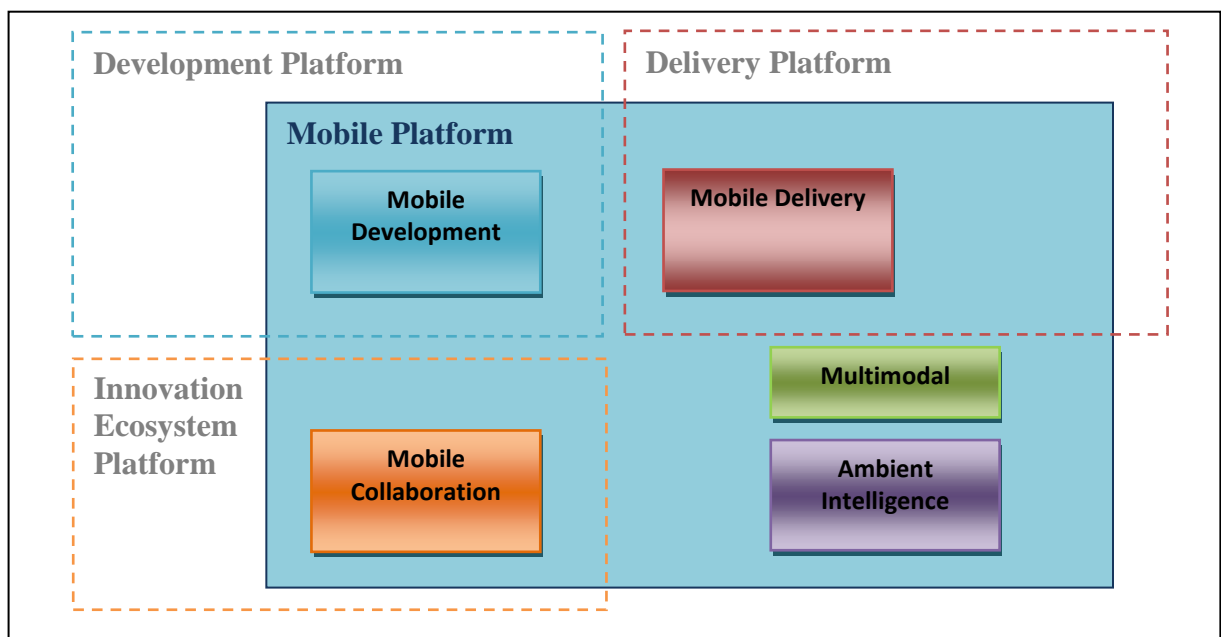
The main goal of the Mobile Platform (as described in the MSEE DoW) is to provide mobile access to other MSEE platforms making available MSEE functionalities to mobile users, taking advantage of the capabilities of smartphones/tablets and addressing some relevant topics for mobile devices.

The following list summarizes the main objectives of the Mobile Platform:

- Support manufacturing enterprises to provide mobile applications to their customers and to their internal staff.
- Enable mobile users to search/monitor services and applications delivered by MSEE ecosystems.
- Improve the mobile collaboration inside ecosystems.
- Support Ambient Intelligence scenarios.
- Exploit multimodal user interfaces to improve the user experience when using mobile applications.

To achieve these objectives the Mobile Platform is composed by several modules. These modules are implemented by some components executed in the cloud (deployed as web services or as web sites optimized for mobile devices) and by some components executed on the mobile devices (deployed as native mobile applications).

A special case is represented by the Mobile Development module that is implemented as an extension of an Integrated Development Environment and is deployed in a development machine.



**Figure 12: Mobile Platform Modules**

As far as possible the modules are independent from particular operating systems, languages or technologies and are based on state-of-the-art technologies possibly open-source.

The following list summarizes the Mobile Platform modules shown in Figure 12:

- **Mobile Development:** end-users access software services provided by MSEE ecosystems through a set of applications (desktop, web, mobile). The Mobile Development module extends the Development Platform focusing on the development of mobile applications.

- **Mobile Delivery:** the module represents the mobile channel of the Delivery Platform enabling business users to discover and rank the services they need, exploiting their mobile devices. The module also enables mobile users to monitor the services they provide.
- **Mobile Collaboration:** the module extends the collaboration features of the Innovation Ecosystem Platform enhancing ecosystem communications and collaboration.
- **Ambient Intelligence:** the module enables users to exploit their mobile devices as their personal universal mobile interface managing the interaction between the owner and the available devices in the ambient.
- **Multimodal:** this module provides a software component simplifying the implementation of mobile multimodal interfaces and shows the advantage of multimodality in specific situations, such as when users are involved in multiple activities and/or are in a changing environment.

Section 6 presents an overview of each module. Some modules have mainly been addressed in the first development cycle, dedicated to the implementation of the Mobile Platform initial prototype, while other modules are mainly addressed in the second development cycle dedicated to the implementation of the final prototype of the Mobile Platform. The modules developed in the first cycle are refined and extended in the second cycle thanks to MSEE end-user feedback.

Table 3 includes the list of the Mobile Platform modules and highlights if they are developed in the first or in the second development cycle. Moreover it shows that all the modules are reusable by third party external applications and that some modules, according their natures, are able to exchange complex data with external applications. In these cases standard widespread data formats like JSON and XMI (see Section 3) have been exploited.

Module	Nature	Dev. Cycle	Usable from 3 <sup>rd</sup> party sw	Standard data formats	Notes
Mobile Development	Plug-In of an open source Integrated Development Environment.	1	Yes.	Yes.	The module can be invoked and extended by others IDE plug-ins following standard OSGI mechanisms. The module can import data in the XMI standard data format.
Mobile Delivery	Web Application + REST Service	1	Yes.	Yes.	The module exploits the JSON format to exchange data. The REST service can be used from external applications to exploit the core functionalities of the module.
Ambient Intelligence	Android Mobile Application	1	Yes.	Not Applicable.	The Ambient Intelligence App can be launched by external applications using standard Android mechanisms. Primitive Java data types are used to exchange data.
MultiModal Module	Android Library	2	Yes.	Not Applicable.	The library can be imported in any Android project and used to develop multimodal applications. The API of the library includes the classes used to exchange data with the library.
Mobile Collaboration	Android Mobile Application + REST Service	2	Yes.	Yes.	The module exploits the JSON format to exchange data. The REST service can be used from external applications to exploit the core functionalities of the module. The Mobile Collaboration App can be launched by external applications exploiting standard Android mechanisms.

**Table 3: List of Mobile Platform modules**



## 5.1. Technical Roadmap

The roadmap shown in Figure 13 highlights when technical results related to the Mobile Platform are available during the life of the MSEE project.

The following list provides a description of the items included in the legend:

- Specification: includes the complete description of the module, its internal architecture and the specification of its components in terms of functionalities and dependencies.
- Early Prototype: the goal of an early prototype is to show the main ideas behind a module. It may contain bugs and/or missing features. Early prototypes are used only for demonstration purposes (demos are shown to end-users by developers) to collect feedback in order to implement modules fully addressing end-users requirements and expectations.
- Initial Prototype: an initial prototype contains the more significant features of a module. End-users will be able to carry on it initial test and experimentations .
- Final Prototype: final prototypes are improvement of initial prototypes where bugs, if present, are corrected and missing features are added. Final prototypes may be further developed and consolidated to produce commercial products or services after the end of the project.

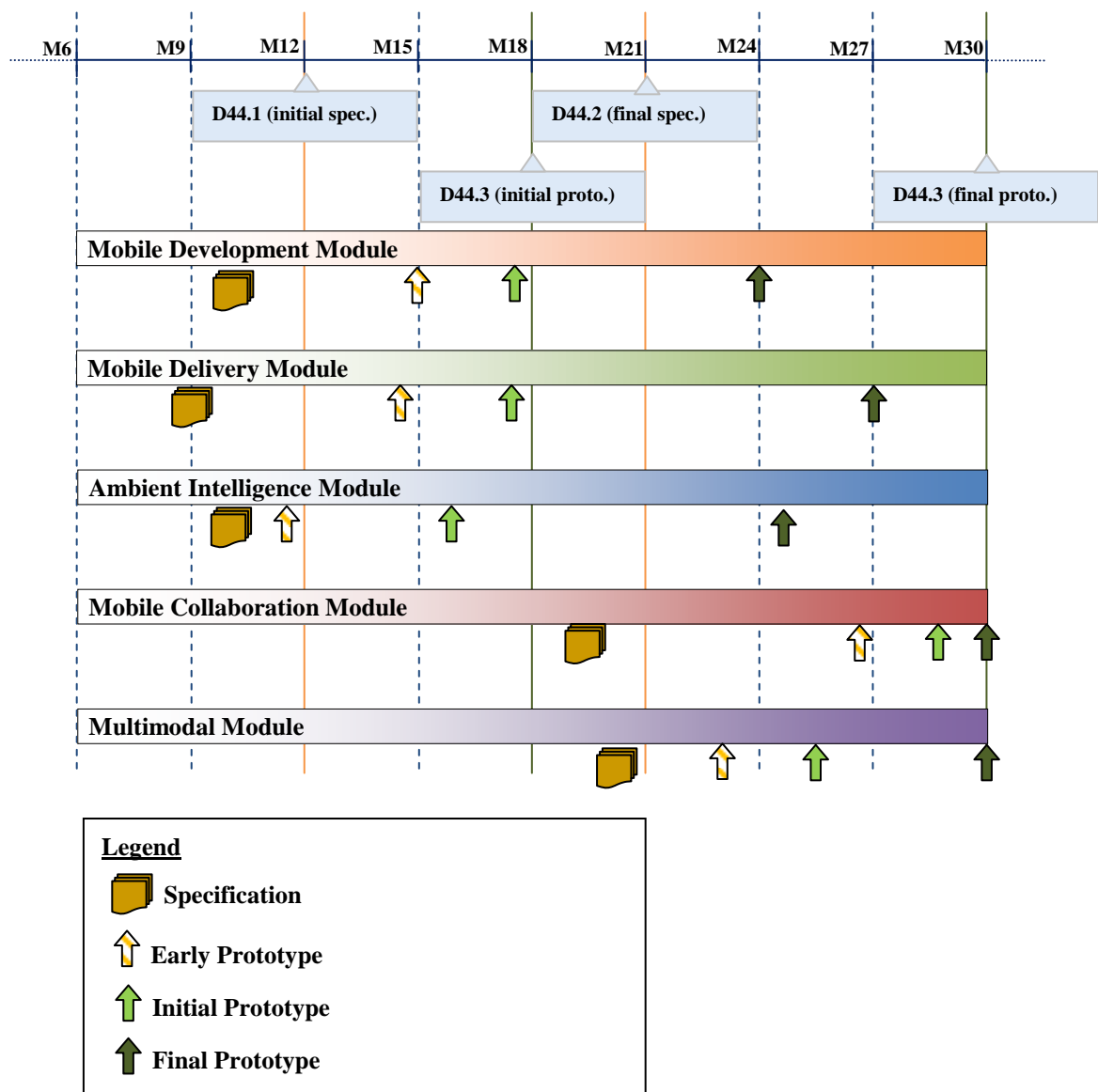



Figure 13: Mobile Business Platform Technical Roadmap

Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	

The roadmap follows the life of WP44 (the work package in which the mobile platform is designed and developed) that starts at month 6 and ends at the month 30 of MSEE. Technical results and software components are included in WP44 deliverables. D44.1 and D44.2 (this document) contain the initial and final architecture and specification of the Mobile Platform while D44.3 and D44.4 represent the initial and final prototype respectively.

The goal of the roadmap is to provide information about the WP objectives and timing and the availability of the Mobile Platform. The dates in which deliverables are provided are defined in the DoW and we plan to comply with them, however due to the research nature of the MSEE project, the dates regarding internal prototypes and specification may be changed depending on the difficulties encountered in the research activities carried on.

## 6. Mobile Business Platform Modules Specification

### 6.1. Mobile Development Module

Today there is a great interest in mobile applications and many enterprises offer to their customers mobile applications to add value to the enterprise products and to support internal staff to accomplish specific tasks when out of the office (i.e. to place orders, to access the enterprise information system, to collaborate with colleagues, etc.).

MSEE end users plan to provide some mobile applications to their customers, to their personnel or to other members of their ecosystems (partners, suppliers, etc.). For example Indesit wants to offer to its customers a mobile application to monitor and control their smart washing machines, Bivolino is going to develop some mobile applications in order to access a larger group of consumers (young mobile and tablet users) while TP Vision already offers some mobile applications to interact with smart TVs and will add new applications to its portfolio.

To design, develop, test and distribute a mobile application is a very complex task involving a lot of experts, therefore the support of the complete development cycle is a goal beyond the scope of the Mobile Development module. There are already some platforms supporting the development of mobile applications for the most popular mobile operating systems like the Android Development Tool<sup>22</sup> (an Eclipse plug-in for creating Android applications) and the Xcode application (an integrated development environment for creating iPhone/iPad applications).

The goal of the Mobile Platform is to complement the available development platforms helping MSEE users to develop the IT parts of their services addressing mobile devices. In particular, as described in the MSEE description of work document, the Mobile Platform will support the development of front-end mobile applications using one or more back-end web services.

#### 6.1.1. Development Process supported by the Module

MSEE users design their services exploiting the concepts and the methodologies provided by the MSEE sub project 1, namely “Service Orientation in Virtual Factory and Enterprises”, and using the MSEE Service Lifecycle Toolbox described in the deliverable D15.2<sup>23</sup>.

A service is represented by a set of models (Business Service Models, Technology Independent Models, Technology Specific Models) describing its organizational-human, manufacturing-physical and information-IT aspects as defined in the deliverable D11.1<sup>24</sup>.

The information regarding the informational-IT aspects include the high level specification of the software, in particular web services and applications, and of the hardware (IT infrastructure) needed to provide the IT parts of the service.

The MSEE Development Platform helps the developers to extend and refine these models in order to specify all the details needed to implement the software components and produces a set of XMI (XML Metadata Interchange) documents including the specification of these IT components.

Then XMI documents are used by the MSEE Development Platform and by the Mobile Development module to generate the programming code, needed to implement software components and addressing a specific set of technologies, automatically.

The MSEE Development Platform and the Mobile Development module are integrated in a unique integrated development environment and the functionalities provided by the Mobile Development module, enhancing mobile development, will extend the functionalities of the

<sup>22</sup> <http://developer.android.com/tools/sdk/eclipse-adt.html>

<sup>23</sup> MSEE\_D15.2: “Specification and Design of SLM Platform”

<sup>24</sup> MSEE\_D11.1: “Service concepts, models and method: Model Driven Service Engineering”

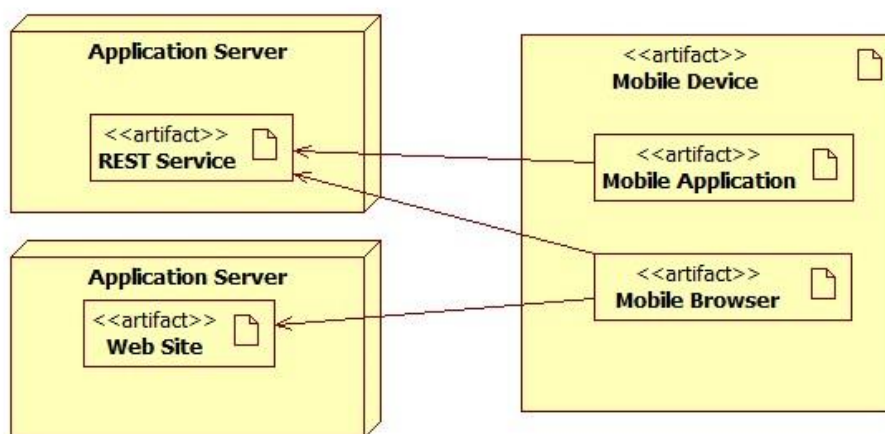
MSEE Development Platform supporting the development of services and desktop/web applications.

In particular the Mobile Development module takes in input the representation of a business object, included in the model of the IT part of the service elaborated with the SLM toolbox and the Development Platform, and supports the user in the development of a web service able to handle the business object and in the development of a mobile application providing the user interface to exploit the service.

### 6..1.2. REST Services and Mobile Applications

The Mobile Development module supports a well-established mobile design pattern for the development of front-end mobile applications exploiting back-end services.

This design pattern suggests the use of REST services to enable mobile applications to handle business objects (resources in the REST terminology) stored on the cloud.



**Figure 14: REST Services and Mobile Applications**

For example a REST service could enable clients (web applications, mobile applications, etc.) to add, delete, update some appointments in a shared calendar and to get the list of appointments. Clients are in charge of managing the user interface and of accessing the service.

Figure 14 shows a deployment diagram where the REST service is deployed on an Application Server and is accessed through the HTTP protocol from a mobile application deployed on a mobile device. Moreover the figure shows a Mobile Browser deployed on the mobile device accessing a web site providing HTML/Javascript code and accessing the REST service (the Javascript code of the web site handles service calls).

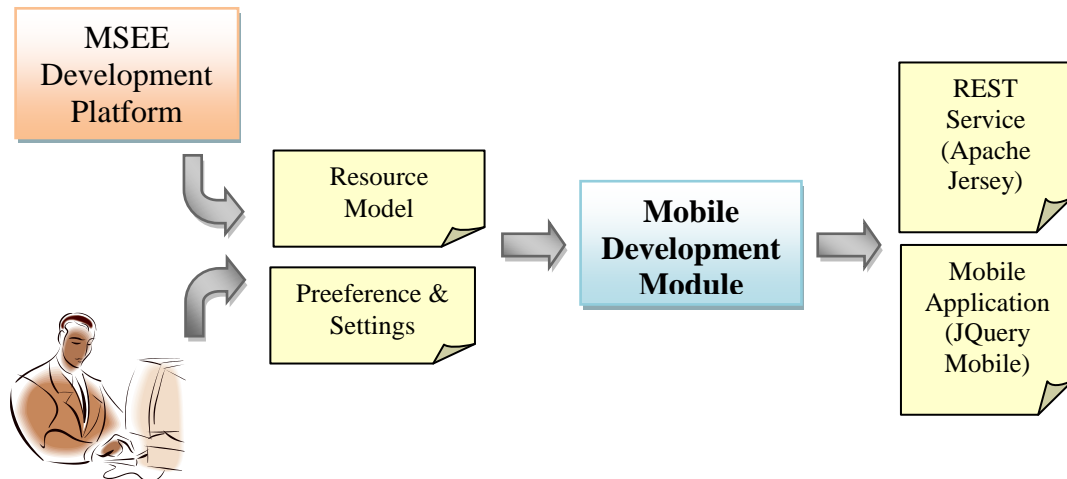
The Mobile Development module will support developers in the implementation of REST services and clients using the Apache Jersey and the JQuery Mobile frameworks.

Apache Jersey represents the standard implementation of the Java API for RESTful Web Services, an API supporting the creation of web services according to the REST architectural style.

JQuery mobile is a javascript library providing specific functions to interact with REST services and suitable stylesheets to develop web applications designed for mobile phones. The module will be extensible to use other frameworks or language (i.e. Android) and possible extensions will be considered during the project.

The Mobile Development module takes as an input the REST resource to manage (i.e. a business object representing a machine or a customer offer or a tv program, etc. ) and generates all the needed code on the client and the server sides according the developers settings and preferences (see Figure 15).

The model of the REST resource to manage is extracted by the XMI documents provided by the MSEE Development Platform or by other representations of these resources like Java Beans or JSON documents.



**Figure 15: Mobile Development code generation process**

REST services provide access to one or more resources (or collection of resource) through the support of the HTTP methods GET, POST, PUT, DELETE.

The following table presents an example describing a hypothetical REST service providing access to the list of products and to a single product delivered by an on-line shop:

Resource	GET	PUT	POST	DELETE
<a href="http://myshop.com/products/">http://myshop.com/products/</a>	<b>List</b> the URIs and perhaps other details of the products.	<b>Replace</b> the entire collection with another collection.	<b>Create</b> a new product in the collection. The new entry's URL is assigned automatically and is usually returned by the operation.	<b>Delete</b> the entire collection of products.
<a href="http://myshop.com/products/item21">http://myshop.com/products/item21</a>	<b>Retrieve</b> the details of a product.	<b>Replace</b> the product or if it doesn't exist, <b>create</b> it.	Treat the addressed member as a collection in its own right and <b>create</b> a new entry in it. For example item21 could represent a collection of shirts of the same model but of different colours and item21/red can represent the red shirt.	<b>Delete</b> the product.

Given the above REST Service it is possible to create a mobile application enabling customers to browse the catalog of the online shop and a web application enabling the shop staff to manage the catalogue. Today many well-known platforms (Facebook, Salesforce, Flickr, Yahoo, Delicious, etc.) expose their APIs as rest services.

The main steps to implement a REST service and a client, supported by the Mobile Development module, are the following:

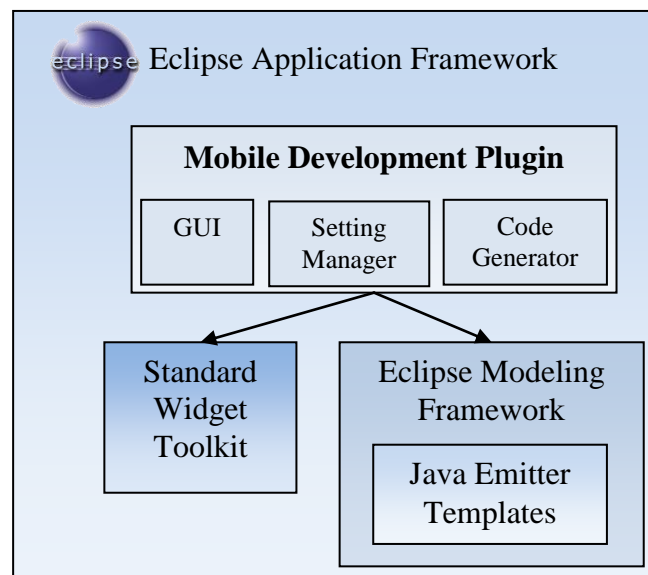
1. Identify the resource to manage
2. Name the resources with URIs: the URI must be designed in such a way that each resource is addressable uniquely (i.e. \*/products identifies a list of products uniquely while \*/products/item33 identify a unique product).
3. Design the representation(s) of the resource exchanged between the client and the server: usually a resource as an XML document or a JSON document. In mobile application the lightness of the JSON language is usually preferred.
4. Implement the uniform interface: provide an implementation for the HTTP operations namely GET, POST, PUT, DELETE.
5. Develop the service client.

This time-consuming and error-prone process is simplified and accelerated by the module.

### 6..1.3. Specification

The Mobile Development module is provided as a plug-in of the Eclipse integrated development environment.

Eclipse (see Section 3) includes a well-defined, supported and documented extension mechanism. In fact, Eclipse is composed by a small kernel containing a plug-in loader surrounded by hundreds of plugins executed into the environment managed by the kernel. Each plug-in contributes to the whole in a structured manner, may rely on services provided by other plug-ins, and each may in turn provide services on which other plug-ins may rely. Eclipse is distributed with a large set of pre-installed plugins and new plug-in can be easily installed to extend the basic features at any time.



**Figure 16: Mobile Development Module**

A plug-in extends Eclipse features through the provision of particular editors, wizards, views, code generation features, and so on. The Mobile Development module offers a wizard enabling the developer to provide the representation of the resource to manage, to set all the preferences and settings (i.e. frameworks to use, paths, etc.) and to generate the REST code automatically.

Figure 16 shows the main components of the Mobile Development module. The module is developed as an Eclipse plug-in, living inside the Eclipse Application Framework, and is

composed by three functional components namely GUI, Setting Manager and Code Generator.

The GUI component is in charge of building the graphical user interface (GUI) and of managing the user interactions. The GUI is built exploiting the graphical components provided by the Eclipse Standard Widget Toolkit.

The Setting Manager component is in charge of configuring the Eclipse projects (library inclusion, classpath setting, etc.) where the code is generated in order to avoid design-time errors (i.e. compilation errors due to the lack of one or more libraries).

Finally the Code Generator component is in charge of generating the REST code exploiting the Java Emitter Templates (JET) tool provided by the Eclipse Modeling Framework. JET enables to write programs able to generate code (code that generates code) using a JSP-like syntax making easy to write templates that express the code to generate. In practice JET is a generic template engine that can be used to generate SQL, XML, Java source code and other output from templates.

The following tables report some details on each component.

<b>GUI</b>	
Description	This component is in charge of building the user interface and managing the interactions with the user.
Functionalities / services	<ul style="list-style-type: none"> <li>• GUI graphical building.</li> <li>• GUI event management and update in response of user interactions.</li> <li>• GUI update in response to notification of other components.</li> </ul>
Dependencies	Eclipse Standard Widget Toolkit (SWT): the SWT provides a rich set of widgets (buttons, labels, links, menu etc.) and other components (events, commands, key bindings, etc.) that can be used to create either standalone Java applications or Eclipse plug-ins.
User	The user (developer) and all the other components.



<b>Setting Manager</b>	
Description	This component is in charge of modifying the target project where the code is generated in order to add the generated code without errors.
Functionalities / services	<ul style="list-style-type: none"> <li>• Java Class Path setting</li> <li>• External library Import</li> <li>• Setting of export options</li> </ul>
Dependencies	Eclipse Application Framework: the module modifies the project properties through the API provided by the platform.
User	This module is used by the GUI to carry on the code generation process.

<b>Code Generator</b>	
Description	This component is in charge of generating the code needed to implement a REST service able to manage the resource received as input and a REST client enabling the user to exploit the service through a suitable graphical interface.
Functionalities / services	<ul style="list-style-type: none"> <li>• Server side code generation</li> <li>• Client side code generation</li> </ul>
Dependencies	Eclipse Modelling Framework/Java Emitter Template
User	This module is used by the GUI to carry on the code generation process.



## 6.2. Mobile Delivery Module

The Mobile Delivery module acts as the mobile channel for the MSEE Service Delivery Platform that is responsible for the registration, discovery, ranking, selection, grounding and invocation, and low level monitoring of the services and applications that are used in manufacturing service ecosystems. The Delivery Platform itself is provided as a set of decoupled infrastructure services: Registry, Discovery, Ranking, Selection, Invocation and Monitoring. The Mobile Platform mainly uses the Discovery, Ranking, Selection, and Monitoring services enabling mobile users to consume them.

In addition to guarantee mobile access to the MSEE Delivery Platform, the Delivery module of the Mobile Platform is designed to interact with any software component playing the Service Broker role introduced by the Future Internet core platform<sup>25</sup> (FI-WARE) project.

FI-WARE is an FP7 European Project (05/2011-04/2014) having the main goal of providing innovative infrastructure for cost-effective creation and delivery of services, providing high QoS and security guarantees. The Reference Architecture of the FI-WARE platform is structured along a number of technical chapters and the Applications/Services Ecosystem and Delivery Framework<sup>26</sup> (SDF) chapter introduces several key business roles (see Figure 17) including the Service Broker one.

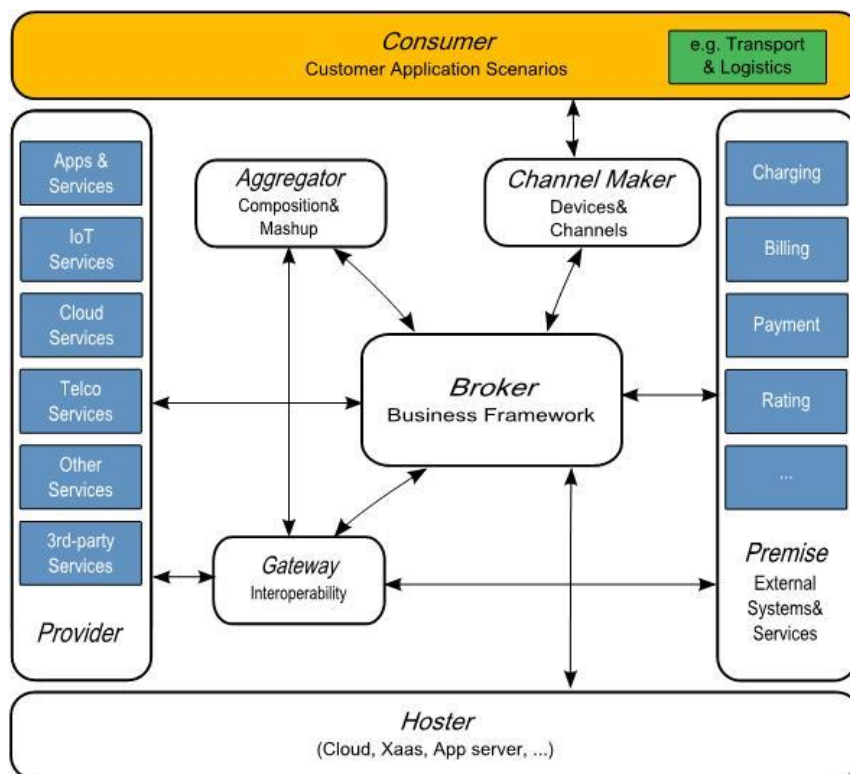


Figure 17: FI-WARE SDF high level architecture

The Service Broker role supports exposing services from diverse providers into new markets, matching consumer requests with capabilities of services advertised through it. In MSEE the Delivery Platform supports the Broker role by providing functionalities to discover services and applications based on their business/technical description. It is expected that in the near future, the Service Broker role will be played by other components developed by research projects and/or by industrial efforts.

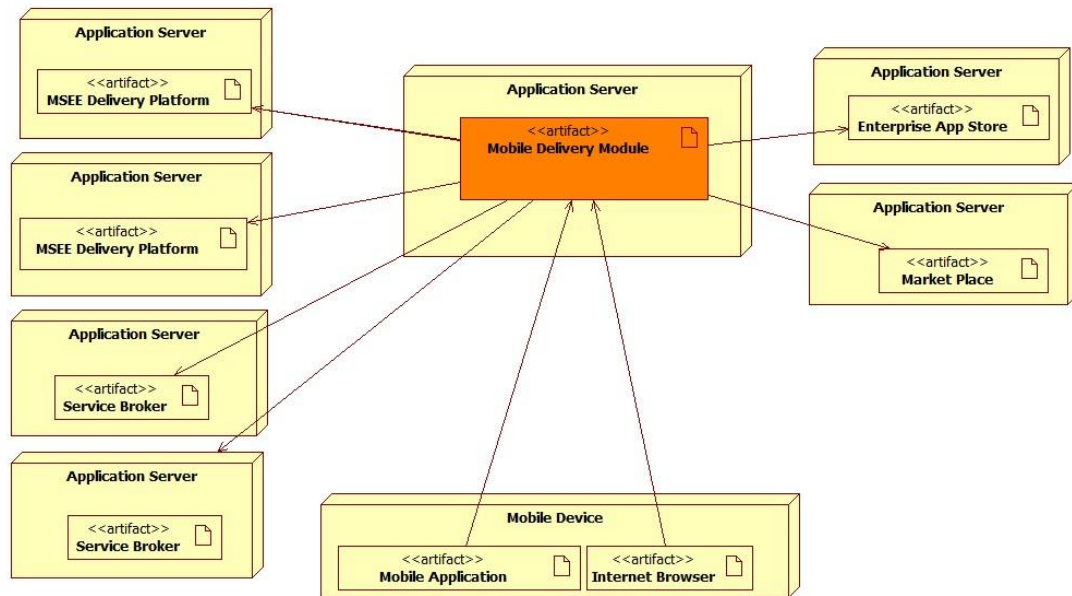
<sup>25</sup> <http://www.fi-ware.eu/>

<sup>26</sup>

[http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Applications/Services\\_Ecosystem\\_and\\_Delivery\\_Framework](http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Applications/Services_Ecosystem_and_Delivery_Framework)

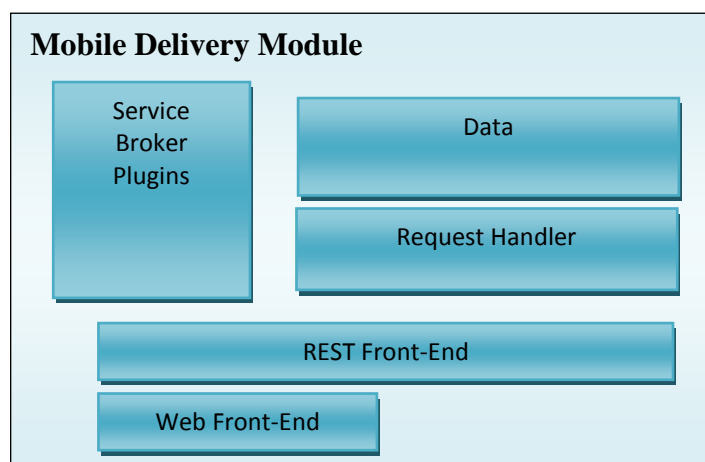
As said before the Mobile Delivery module is designed to interact with multiple instances of the MSEE Delivery platform and in general to interact with several components playing the Service Broker role. In this way the mobile user will be able to search for services and applications exploiting multiple sources.

Figure 18 shows a deployment diagram in which the Mobile Delivery module interacts with two instances of the MSEE Delivery Platform and with other four Service Brokers.



**Figure 18: Mobile Delivery Deployment Diagram**

The Mobile Delivery Module offers its functionalities through a web site designed for mobile devices and through a REST service (see Figure 19). The user is able to access the Mobile Delivery Module using a web browser or using a mobile application able to interact with the web interface provided by the module. For example the Ambient Intelligence App described in Section 0 enables the user to find available services and applications for discovered devices using the Mobile Delivery module.

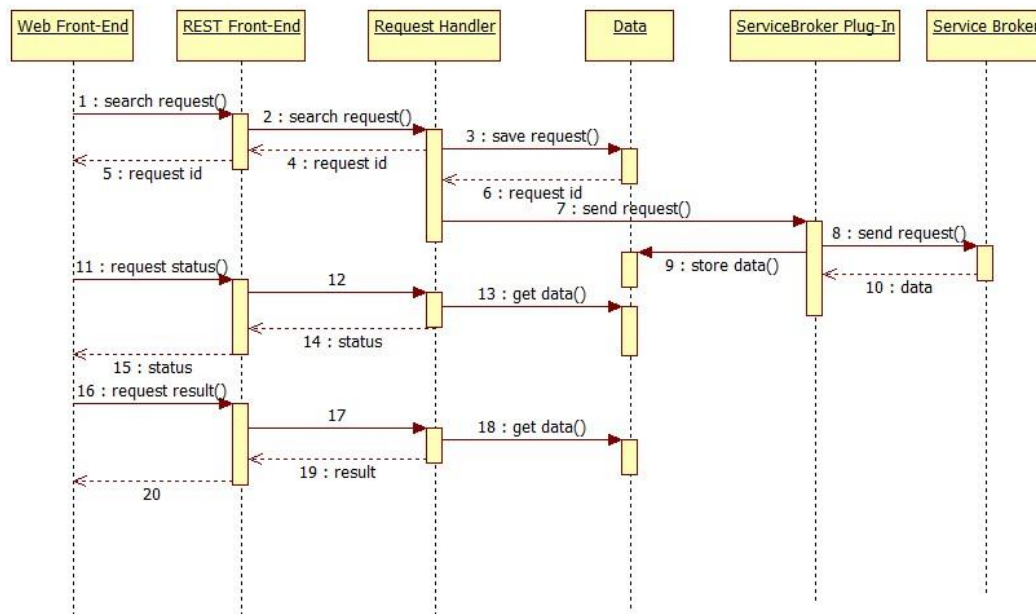


**Figure 19: Mobile Delivery Module**

The Web Front-End component enables users to access the Mobile Delivery module with mobile browsers and will be implemented as a HTML5 / Javascript (including some Ajax components) web site exploiting internally the REST Front-End component of the module. The Request Handler is in charge of managing the searches for services issued by users through the Web Front-End and/or the REST Front-End and of managing the response received by the Delivery Platform as well as by other Service Brokers.

The Service Broker Plugins includes a specific plug-in for each Service Broker to contact since different Service Brokers are accessible through different interfaces and/or technologies. Finally the Mobile Delivery module stores (for a certain amount of time) the response received by of the Service Brokers allowing clients (browsers and mobile applications) to access these responses asynchronously (i.e. with Ajax).

The following diagram depicts the main interactions between the various components.



**Figure 20: Mobile Delivery search sequence diagram**

The user exploits the Web Front End to start a search request. The Request Handler saves the request through the Data component that assigns a unique id to the request. The assigned id is returned to the client that from this moment, using the id, can request information about the search status and/or the search result.

A search can take some time since different Service Brokers are contacted and some of them can be slow or a network delay can occur. With the asynchronous approach the client can avoid to freeze the user interface while it can exploit the “request status” operation and the “request results” operation to update the UI, highlighting the search status or returning the search result.

After saving the request, the Request Handler alerts the registered Service Brokers Plug-Ins. Each plug-in manages the interaction with a specific Service Broker saving the search results using the Data component.

The following tables provide more detailed information on each component.

Web Front End	
Description	This component is implemented as a web site designed specifically for mobile devices. The web site makes use of Ajax to interact with the REST Front-End component asynchronously ensuring an engaging user experience.
Functionalities / services	<ul style="list-style-type: none"> <li>• Manage the User Interface</li> </ul>
Dependencies	
User	This module is used by business users in mobility searching for services.

REST Front End	
Description	This component is implemented as a REST service providing all the functionalities to add new searches and to request update on the search status and on the search results.
Functionalities / services	<ul style="list-style-type: none"> <li>• Add a search</li> <li>• Cancel a search</li> <li>• Get search status</li> <li>• Get search results</li> </ul>
Dependencies	
User	This module is used by the Web front end or by external applications.

Request Handler	
Description	This component handles requests accepted by the REST front end.
Functionalities / services	<ul style="list-style-type: none"> <li>• Handle Request</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Data component: the Request Handler depends by the Data component to interact with the internal database.</li> <li>• Service Broker Plugins component: the Request Handler depends by plugins to handle search requests.</li> </ul>
User	This module is used by the REST Front end.

Data	
Description	This component manages the internal data of the Mobile Delivery module.
Functionalities / services	<ul style="list-style-type: none"> <li>• Save search</li> <li>• Cancel search</li> <li>• Get search result</li> <li>• Get search status</li> <li>• Clean Data</li> </ul>
Dependencies	
User	This module is used by the Request Handler component to save searches and to retrieve search status and results. The component is used by plug-ins to store search results and update search status.

Service Broker Plugins	
Description	Each Service Broker is accessed through a dedicated plug-in in charge of issuing the searches and getting back the results.
Functionalities / services	<ul style="list-style-type: none"> <li>• Send the search to a Service Broker</li> <li>• Receive Service Broker results</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Data component: to store search results</li> </ul>
User	This module is used by the Request Handler to handle search requests.

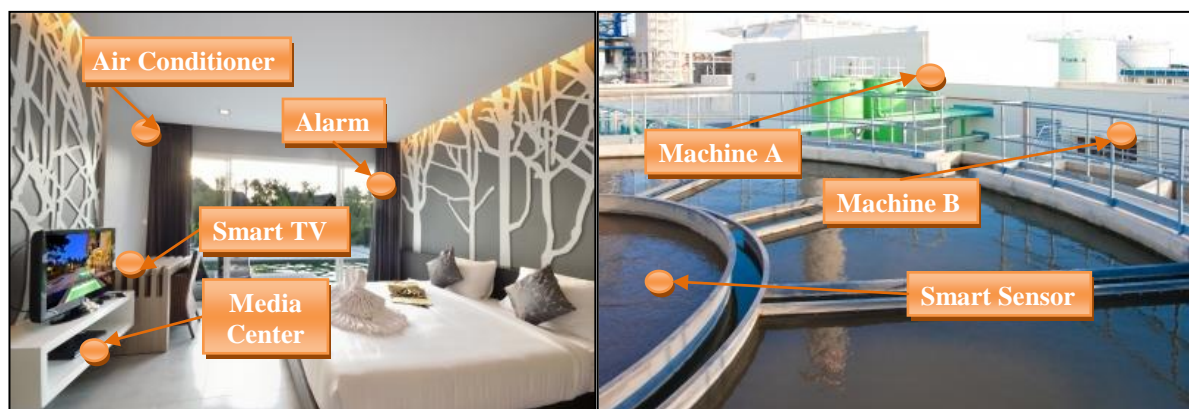
### 6.3. Ambient Intelligence Module

The Ambient Intelligence vision, advanced in the 1999 by EU IST Advisory Group, included a “human surrounded by computing and advanced networking technology which is aware of his presence, his personality, his needs and is capable of responding intelligently to spoken or gestured indications of desire, and even in engaging in intelligent dialogue”.

Even if this vision has not been achieved completely, many steps have been done in the right direction, and today we are surrounded of smart products able to interact with humans and other products.

The main goal of the AMI module is to discover and interact with the devices disseminated in the environment in order to support Ambient Intelligence and Internet of Things (IoT) scenarios and, at the same time, address the general objectives of MSEE and the concrete needs of MSEE end-users.

Figure 21 shows two heterogeneous environments where several devices are available.



**Figure 21: Home and Manufacturing Environments<sup>27</sup>**

The following table report the main requirements considered during the definition of AMI functionalities. These requirements have been collected from the MSEE Description of Work document and from the deliverable D52.1 including the initial set of MSEE end-user requirements.

<sup>27</sup> Images by arztsamui and John Kasawa (FreeDigitalPhotos.net)



Requirement	Feature
<b><i>Ambient Intelligence</i></b> <p>The original Ambient Intelligence vision included a “human surrounded by computing and advanced networking technology which is aware of his presence, his personality, his needs and is capable of responding intelligently to spoken or gestured indications of desire, and even in engaging in intelligent dialogue”.</p>	<p>The AMI module helps a product to be aware of other products present in the same environment and to interact with them in order to build a coherent, updated and shared user profile usable by each of them to be aware of user preferences, needs, personality, etc.</p>
<b><i>AMI Space</i></b> <p>The 2002 ISTAG Report “Strategic Orientations &amp; Priorities for IST in FP6” introduced the notion of ‘AmI Space’ in which there will be seamless interoperation between different environments – home, vehicle, public space, etc.</p>	<p>The AMI module enables the user to save and manage different environments, to group the devices that he uses respect to these environments and to carry on his profile from an environment to another. Moreover the AMI module is able to automatic recognize the user environments.</p>
<b>Internet of Things</b> <p>In the Internet of Things (IoT), the “things” are expected to become active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information “sensed” about the environment, while reacting autonomously to the “real/physical world” events and influencing it by running processes that trigger actions and create services with or without direct human intervention.</p>	<p>The AMI module addresses IoT topics helping things to become active participants in business, information and social processes through the access to proper services facilitating interactions with other “smart things” and making things in the same environment aware of each other.</p>
<b>MSEE Servitization Process</b> <p>One of the main objectives of MSEE is to support manufacturing enterprises to go through all servitization levels enriching and differentiating their products with new services that could appear and disappear during the products life.</p>	<p>The AMI module enables customers to find services and applications for the products around them and to link the physical available product with their virtual representation in the cloud.</p>
<b>MSEE End Users – TP-Vision</b> <p>TP-Vision pilot application focuses on the extension of the Philips Net TV services accessible from Philips Smart TVs.</p>	<p>The AMI module is able to recognize Philips Net TVs and can suggest the available services and mobile applications. Moreover the module can enable the user to carry, in his mobile devices, all the settings of his TV in order to quickly set up other Net TVs (i.e. TV in a hotel) according his preferences.</p>

<p><b>MSEE End Users - Indesit</b></p> <p>The Indesit use case focuses on the “Carefree Washing” service. The capabilities of the Indesit Washing Machines will be extended by a set of services that helping the customer to avoid caring about maintenance, machine control, soap recharge, spare parts, etc. During the product lifecycle (some years) some services will appear while other could be discontinued.</p>	<p>The AMI module can help the user to discover new services for his particular WM (model, features, production year, available firmware, etc.) and can support interactions between the WM, the user and the available services. The module can also helps the Indesit WM to be aware of other machines in the same environment in order to enhance machine-to-machine interactions.</p>
<p><b>MSEE End Users – Ibarmia</b></p> <p>Ibarmia use case focuses on the “Intelligent Maintenance Services Ecosystem” in which, for foreign countries, the machine maintenance operations will be carried out by external authorized technical assistance service providers.</p>	<p>The AMI module can help external technicians entering in unknown industrial facilities to identify Ibarmia machines (model, serial code, etc.) and can check if new services/information (i.e. technical procedures, advices, etc.) are available for the machines.</p>

**Table 4: Ambient Intelligence module requirements**

Some assets of the AMI module internal architecture and some of its functionalities have been inspired by the results achieved in past research projects (in particular PERSIST, MIMOSA<sup>28</sup> and MINAami<sup>29</sup>). These projects focused specifically on Ambient Intelligence issues applied to mobile devices. The AMI module addresses several Ambient Intelligence issues but must be also considered in the context of MSEE project where plays the innovative role of a virtual bridge between the products, the customers and the services provided by the smart manufacture enterprises, supporting these manufactory enterprises to achieve the *product+services* and *products2services* servitization phases.

The Ambient Intelligence module handles two main concepts namely devices and environments. The term *device* is used here as synonym of product (washing machine, TV, sensor, etc.) or thing or entity of interest (IoT terminology) while the term *mobile device* identifies the user mobile phone, smartphone or tablet.

<sup>28</sup> MIMOSA website, [www.mimosa-fp6.com](http://www.mimosa-fp6.com), accessed 22 May 2012

<sup>29</sup> MINAami website, [www.fp6-minami.org](http://www.fp6-minami.org), accessed 22 May 2012





**Figure 22: User, Environments and Devices**

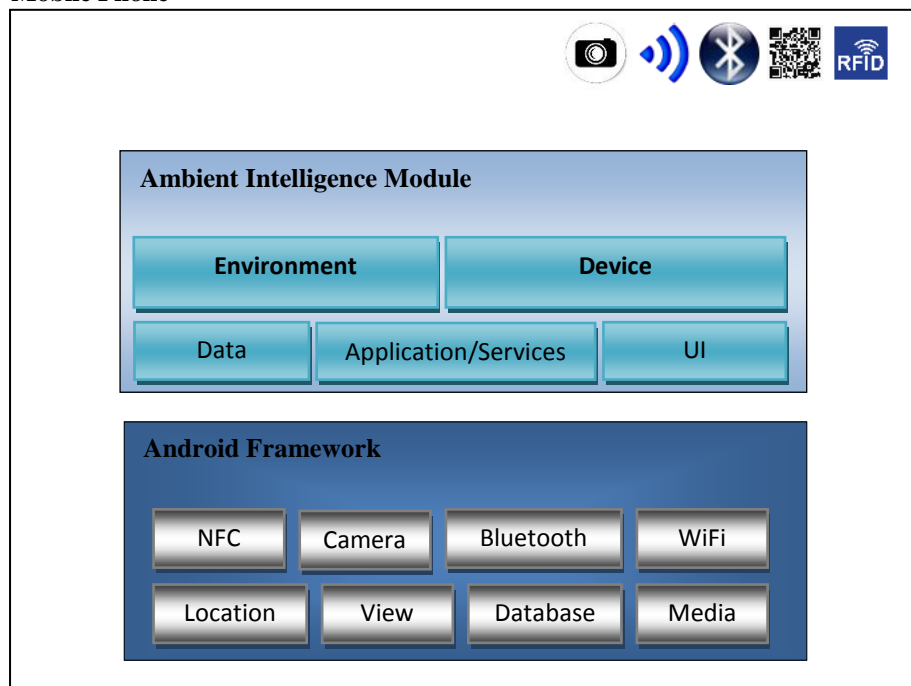
Figure 22 shows three environments (home, office, hotel room) where there are some devices (TV, satellite receiver, air conditioner, phone, etc.). The user moves between these environments carrying on his mobile device where the AMI Ambient Intelligence App is installed.

The AMI module is implemented as an Android application (Figure 23 shows the AMI app internal architecture) exploiting the Android framework building blocks and the mobile phone hardware (i.e. camera, bluetooth, network connections, etc.).

The main components of the module are the Device (dedicated to manage devices) and the Environment (dedicated to manage environments) ones.

In addition there are components devoted to handle the user interface (UI), the internal data (Data) and the discovery of applications and service on the mobile phone and on the cloud (Application/Services).

#### Mobile Phone



**Figure 23: AMI Module**

A physical device in the ambient is represented within the AMI module as an object with the following attributes:

- Name: a descriptive name assigned by the user (i.e. Home WM, Hotel Bristol TV, etc.)
- Thumbnail: a thumbnail of the device taken using the mobile device camera
- Identity Information: a generic list of key-value pairs that can include several attributes like the product model, the manufacturer name, the serial code, etc. and that is defined by the device manufacturer.

Once a device has been identified the user can save it and associate it to an environment.

Some devices are able to communicate through one or more communication channels like the Bluetooth or the WiFi technologies while other devices are not able to communicate but can provide some information about themselves through QR codes, 1D bar codes, RFID tag, etc.

In order to discover and identify a device the AMI module exploits several technologies. In some case the discovery and identification is user driven (i.e. the user see a legacy manufactory machine and input its product code using the mobile device keyboard) while in other case is driven by the module (the AMI module discovers a temperature sensor automatically using the Bluetooth technology).

The following sub-sections provide some details regarding relevant topics related to device identification, environment recognition and regarding the functionalities provided by the AMI module once a device has been identified. Finally the Section 6..3.5 presents the internal components of the module.

### **6..3.1. Device Id**

When a device has been identified it is associated with an internal identifier together with a list of attributes provided by the product manufacturer. The attribute list can include the product model, manufacturer, firmware and so on, and it is used by the AMI module to provide some features such as to find available services for a certain product model. The list of attributes can include information belonging to one or more product identification standards.

Even if today several product identification standards exist, there is not a universal standard and some standards can be applied only to particular products (i.e. books). In the following it is provided a short summary of existing product identification standards.

#### **Universal Product Code**

The Universal Product Code (UPC) is a 12 numerical digits barcode, uniquely assigned to each trade item, widely used in North America, and in countries including the UK, Australia, and New Zealand for tracking trade items in stores.

#### **International Article Number**

An EAN-13 barcode (originally European Article Number) is a 13 digit barcode standard which is a superset of the original Universal Product Code system.

**Global Trade Item Number** <sup>30</sup> is an identifier for trade items used to look up product information in a database (often by inputting the number through a bar code scanner pointed at an actual product) which may belong to a retailer, manufacturer, collector, researcher, or other entity. The uniqueness and universality of the identifier is useful in establishing which product in one database corresponds to which product in another database, especially across organizational boundaries.

<sup>30</sup> <http://www.gtinfo.info/>

## International Standard Book Number

The International Standard Book Number (ISBN) is a unique numeric commercial book identifier barcode composed by 13 digits and compatible with the International Article Number Bookland. An ISBN code is assigned to each edition and variation (except re printings) of a book.

### 6..3.2. Device Discovery and Identification

According to its physical and technological features a device can be discovered by the user directly (for example the user sees a big machine) and/or by the AMI module (for example the module discovers an hidden temperature sensor) through WiFi, RFID, Bluetooth, NFC, etc.

Once a device has been discovered its attribute list must be retrieved and stored into the AMI module. If the device is able to communicate it can send its identification attributes to the AMI module that will store them. Otherwise the user is in charge of providing these attributes using one of the available input methods provided the AMI module (i.e. using the mobile device keyboard or scanning the product QR code).

The rest of the section reports some details on the available discovery and identification options.

### Bluetooth Discovery and Identification

Bluetooth (BT) is a proprietary technology standard for exchanging data over short distances (the range depends by the power class of devices and is usually included between 5 and 100 meters) and it is today available on many devices like mobile devices, car navigation systems, sensors, printers, personal computers, game consoles, watches, health devices, etc.

BT technology is based on a packet-based protocol with a master-slave structure. The interoperability of different application exploiting the protocol is accomplished by BT profiles<sup>31</sup>.

BT profiles define the interactions, including application behaviours and exchanged data formats, between the layers of the bluetooth technology stack inside a device as well as the peer-to-peer interactions of specific layers between different devices.

The Bluetooth system defines a base profile (GAP), which all BT devices implement, focusing on device discovery, link establishment and security procedures.

The GAP profile enables the development of applications, deployed on a BT device, enabling the user to discover other BT devices, to acquire their name and address and to establish a connection following a security procedure named “pairing process”.

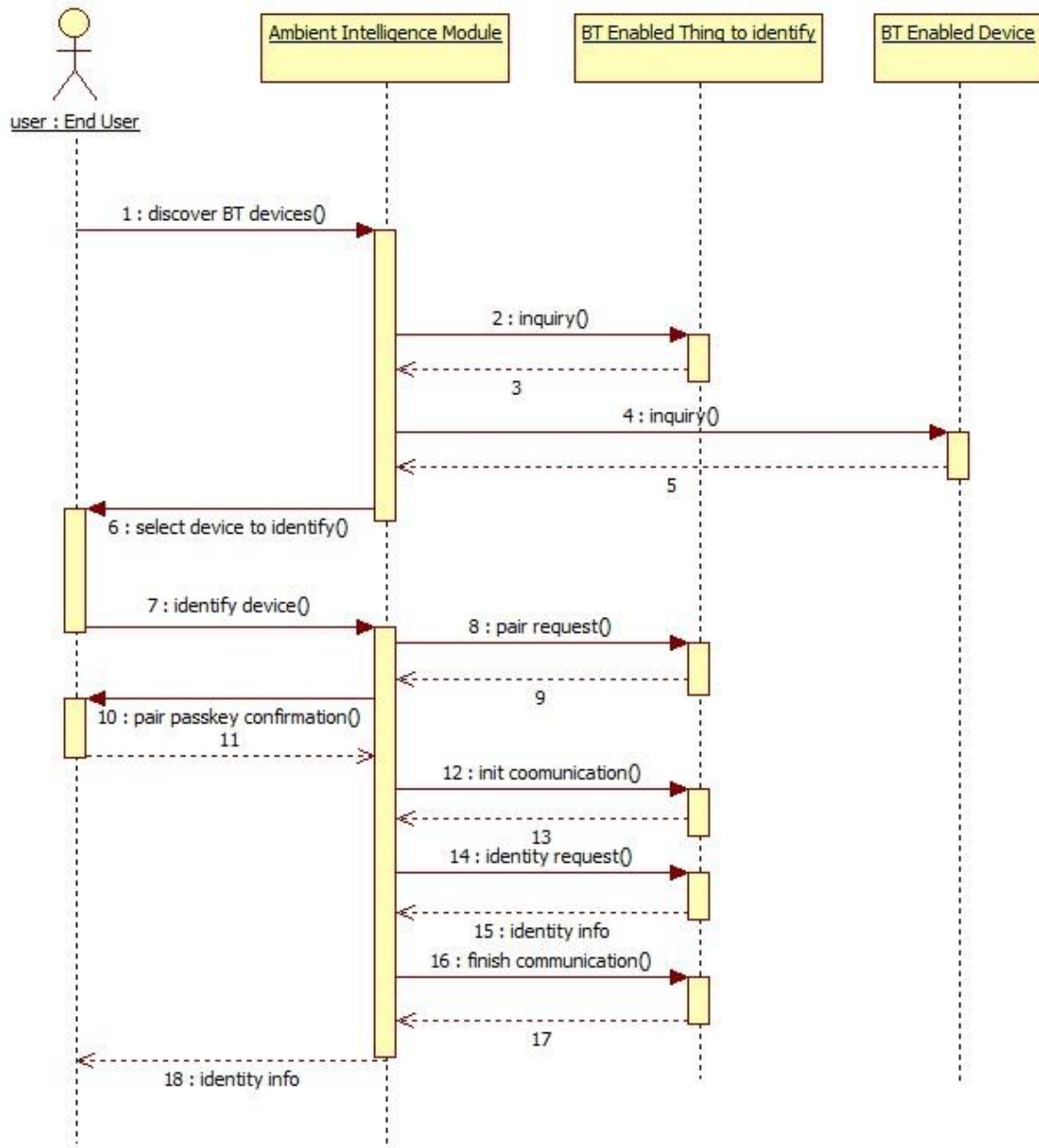
The Serial Port Profile (SDP) enables two devices to establish a communication channel like they were connected by a physical cable through a virtual serial port abstraction. Once the connection has been established the two devices can exploit a communication channel to exchange data.

The GAP and the SPP profiles are used by the AMI module to discover and identify things. In practice the user, exploiting the AMI module deployed on his mobile device, performs the following actions:

1. Search BT devices
2. Select a device of interest and follow the pairing process
3. Retrieve the device identification attributes
4. Save the device in the AMI module

The following diagram summarizes the interactions happening during the identification of a device.

<sup>31</sup> Bluetooth Core Specification V.4 <https://www.bluetooth.org/Technical/Specifications/adopted.htm>



**Figure 24: Ambient Intelligence device identification sequence diagram**

The user starts asking the AMI module to look for available BT devices in the environment. The AMI module exploits the bluetooth capabilities of the user device to accomplish the request and to find all BT devices in the user ambient showing to the user their descriptive names. Examples of names can be “Motorola Headphone X23” or “Indesit Aqualtis WM” etc. At this point the user is requested to select the device to be identified (i.e. Indesit Aqualtis WM). The device to be identified must support the SPP profile playing the Acceptor role. After the pairing process the AMI module (playing the role of Initiator) establishes a communication channel with the device and sends over the channel an identity request (request message). The device must reply to this request with a JSON string including a set of identification attributes expressed as key-value pairs (i.e. model: aqualtis aq8l-09, manufacturer: indesit, ean: xoo78554). When identity information has been received the AMI module sends a “bye” message so the two parties can release the communication channel correctly and the device can go on waiting for other identity request.

## Wi-Fi Discovery and Identification

Wi-Fi is the acronym of “*Wireless-Fidelity*”, a trademark of the Wi-Fi Alliance. Wi-Fi<sup>32</sup> networks use radio technologies to provide secure, reliable, fast wireless connectivity. A Wi-Fi Local Area Network can be used to connect electronic devices to each other, to the Internet, and to wired networks which use Ethernet technology.

Today many mobile devices, personal computers and other apparatus are equipped with a wireless network interface controller and in many area (public buildings, industrial plants, private houses, etc.) it is available a Wi-Fi LAN.

Respect to bluetooth the Wi-Fi technology has a greater range and a major power request so, even if some devices support both Wi-Fi and BT, many devices support only one of these technologies according to their features and goals.

The AMI module, exploiting the mobile phones and the Android framework functionalities, is able to discover and to connect to available LAN networks and, if needed, to authenticate to these networks. Once the AMI module is connected to a LAN networks it uses the TCP-IP (Internet protocol suite) to communicate with other devices on the same LAN.

Wi-Fi LAN do not support the discovery of other devices connected for security reasons. The approach used by the AMI module is to send a multi-cast message including an identity request and to handle the responses received by other devices.

## Barcode and QR Codes Identification

A barcode (see Figure 25) is an optical machine-readable representation of data, which shows data about the object to which it attaches. Originally barcodes represented data by varying the widths and spacing of parallel lines, and may be referred to as linear or one-dimensional (1D). Later they evolved into rectangles, dots, hexagons and other geometric patterns in two dimensions (2D). Barcodes originally were scanned by special optical scanners called barcode readers; later, scanners and interpretive software became available on devices including desktop printers and smartphones.



**Figure 25: Barcode**

QR Code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or two-dimensional code) first designed for the automotive industry. More recently, the system has become popular outside the industry due to its fast readability and large storage capacity compared to standard UPC barcodes. Due to their graphical features to scan a bar code with a mobile device is easy and fast and QR code can store a lot of text data (up to 4.296 characters).

The QR code shown in Figure 26 stores the model, the manufacturer and the serial code of a product (JSON encoded): `{"model": "aspire", "manufacturer": "acer", "serial_code": "882IIOP12"}`.

<sup>32</sup> <http://www.wi-fi.org/discover-and-learn>





**Figure 26: QRCode**

The AMI module is able to scan Bar codes and QR codes exploiting the Zxing<sup>33</sup> Android application. Even if the code of Zxing is licensed under the Apache License 2.0 it is not integrated in the module in order to take advantage of Zxing updates provided by the Android Automatic Update mechanism. Zxing supports the scanning of the several following bar code formats.

### **Keyboard Identification**

Even if a device is not able to provide identity information actively or through a Bar/QR code the user can provide some information on the device using the mobile device keyboard.

In the near future new identification and discovery methods will be considered (i.e. Near Field Communication, Radio Frequency Identification) considering the availability of these technology on mobile devices (today only few devices supports these technologies).

#### **6..3.3. Environment Recognition**

An environment is a *virtual location* where some devices are available. An environment can represent a place characterized by a geographic location (home), a *class* of places (Hilton hotels) or even an environment able to change location like the user car or a train wagon.

The AMI module enables the user to save his environments assigning them a name and a thumbnail and defining the environment characteristics.

The AMI module can recognize an environment, previously saved by the user, using:

- Geographical attributes (detected using GPS, cell phone location, WiFi location, etc.)
- Presence of a Wi-Fi network
- Presence of a connected bluetooth device (i.e. car wireless earphone)

In some case it is not possible to recognize an environment and the user can select the current environment manually. For example it is not possible to distinguish between two environments representing two adjacent rooms where GPS is not available (GPS does not work inside buildings), where the same WiFi network is available and where there are not particular BT devices connected.

When the user saves a new environment he can define on which basis the AMI module must recognize the environment. The recognition process is transparent for the user and is based on an environment recognition algorithm exploiting several technologies.

The idea behind the algorithm is that when more than an environment is eligible as *current environment* it is selected the environment associated to the available technology with the lowest range. The following ranges are taken in account:

- GSM Cell phone: some kilometres
- Phone GPS Accuracy: 3-5 meters
- Bluetooth range (class 2 and 3): 5-10 meters
- WiFi range: about 32 meters indoors and 95 m outdoor

<sup>33</sup> [code.google.com/p/zxing/](http://code.google.com/p/zxing/)

To retrieve information on the current user environment (location, available Wi-Fi networks and connected devices) is an expensive operation in terms of battery consumption. The AMI module implements a cache mechanism able to minimize the power request.

#### 6..3.4. Functionalities available for identified devices

Once a device has been identified (automatically or manually) it can be associated to an environment. At this point the AMI module is able to provide the following functionalities:

- find already installed applications to interact with the device
- find new services and applications to interact with the device
- store on the mobile device the information provided by the device (i.e. user profile, settings, preferences) and export them to devices in other environments enabling the user to bring his user context in different environments.
- receive textual and multimedia information from devices not equipped with proper displays to show them (air conditioners, washing machines, temperature sensors, humidity sensors, etc.)
- send a multicast message to all the devices in the same physical environment making them aware of each other (enabling IoT and Ambient Intelligence scenarios).

#### 6..3.5. AMI functional components

The following tables report some details on the functional components shown in Figure 23:

Environment	
Description	This component manages the saving, the modification and the recognition of environments.
Functionalities / services	<ul style="list-style-type: none"> <li>• Save Environment</li> <li>• Modify Environment</li> <li>• Delete Environment</li> <li>• Recognize Environment</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> <li>• Data component: used to save/retrieve data about environments</li> </ul>
User	This module is used by the UI when the user handles his environments and when the module suggests to the user the current environment.



Device	
Description	This component is dedicated to manage the various operations on devices (discovery, identification, saving, etc.)
Functionalities / services	<ul style="list-style-type: none"> <li>• Discover Device</li> <li>• Identify Device</li> <li>• Save Device</li> <li>• Delete Device</li> <li>• Interact with Device: settings save/export, request device messages to be shown to the user, inform the device about other devices in the same environment, etc.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> <li>• Data component: used to save/retrieve data about devices</li> <li>• Application/Services: used to find application and services for the devices</li> </ul>
User	This module is used by the UI when the user handles his devices.

Data	
Description	This component manages the internal data that are stored into a local database.
Functionalities / services	<ul style="list-style-type: none"> <li>• Save/Update/Delete device</li> <li>• Get list of devices</li> <li>• Save/Update/Delete environment</li> <li>• Get list of environments</li> <li>• Save/Retrieve user preferences</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This module is used by the UI, Device and Environment components when they need to store/retrieve data.

Application/Services	
Description	This component is in charge of finding application and services for a device.
Functionalities / services	<ul style="list-style-type: none"> <li>Find Application</li> <li>Find Services</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	This module is used by the Device component to find applications and services for a device.

GUI	
Description	This component is in charge of building the User Interface (UI) and manage the interactions with the user.
Functionalities / services	<ul style="list-style-type: none"> <li>UI graphical building</li> <li>UI event management and UI update in response of user interactions</li> <li>UI update in response to notification of other components</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	The user.

## 6.4. Mobile Collaboration Module

The goal of the Mobile Collaboration module is to improve the collaboration between the members of Manufacturing Service Ecosystems extending the services provided by the Innovation Ecosystem Platform (IEC) described in the deliverable D26.2.

The IEC supports the management of the ecosystem and of its members (Manufacturing Service Ecosystem Governance), fosters and stimulates open innovation actions among the ecosystem actors (Manufacturing Service Ecosystem Innovation), and allows Virtual Manufacturing Enterprises to deliver new value proposition to customers (Virtual Manufacturing Enterprise Operations). In all these processes the collaboration between ecosystems members plays a central role and mobile devices can facilitate and enhance this collaboration.

Ecosystems are flexible communities where the people of ecosystems companies, universities and research-centers can appear and disappear. To find the right person at the right time anywhere and to be able to collaborate with him in a fast way through several communication channels makes the difference when facing business challenges where the reaction time and the collaboration are crucial aspects.

Thanks to the Mobile Collaboration module users on the move (i.e. not only out of their offices but in situations where a suitable working environment is not available like in a station or an airport) are able to check the ecosystem message board and the shared calendar, to browse searching ecosystems companies and people and to find the best way to interact with them using the available contact details and the mobile applications which support mobile communication and collaboration.

To maximize the usability of the module and its integration with existing applications (i.e. facebook, skype, google talk, calendars, mail applications, etc.) the Mobile Collaboration module is implemented as a native mobile application exploiting a dedicated web service to interact with the IEC.

The Innovation Ecosystem Platform front-end is based on the Liferay Portal<sup>34</sup> and offers several collaboration tools:

- Ecosystem Directory: provides information on all the entities (name, address, web site, etc.) belonging to the ecosystem and on all the members (name, mail address, skype id, facebook id, etc.) of these entities.
- Ecosystem Calendar: enables ecosystem members to advice other members on relevant events related to the ecosystem.
- Ecosystem Message Board: provides several functionalities to manage message threads on a shared board that members use to share information or discuss topics.

These tools, like any other part of the Innovation Ecosystem Platform, are designed to be accessed with a normal browser. When accessed with a mobile device browser the user experience is subjective to the availability of Internet bandwidth and to the mobile device display size. Figure 27 shows how the IEC deployed for the Indesit ecosystem appears when accessed with a smartphone.

It is possible that, in order to find the phone number of a person, the user is forced to navigate the portal carrying on a long sequence of page visits and zooming and, finally, he needs to copy the phone number in the phone display manually to be able to call it. If the person does not answer to the call the user must look for other communication channels and use external mobile applications to exploit them.

There are also aspects not related to the GUI preventing the usage of the IEC when the user is “on the road” like the need of a reliable Internet connection and the amount of network data exchanged between the browser and the IEC that is a problem in case of slow connections and roaming rates.

<sup>34</sup> [www.liferay.com](http://www.liferay.com)



**Figure 27: IEC access from a mobile device**

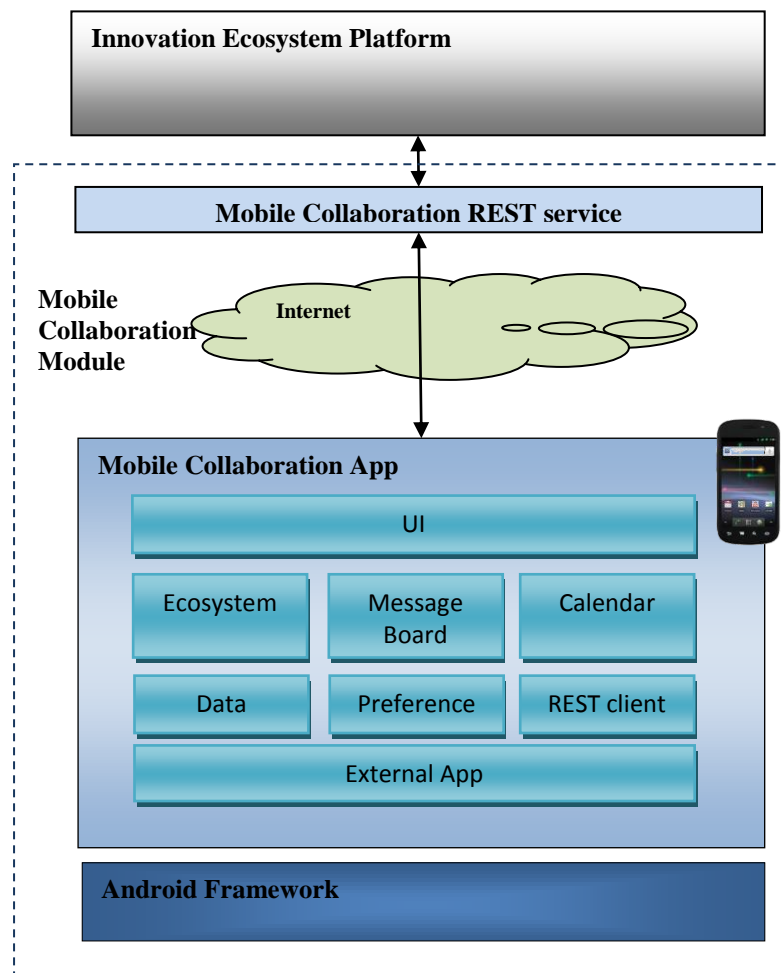
The Mobile Collaboration module goes beyond these limitations guarantying an engaging usage experience to users in mobility regardless of the size of the owned mobile device or of the characteristics of the available Internet connection. This result is achieved thanks to the presence of a user interface able to adapt to the mobile device where it is presented, to the availability of a specific web service in charge of sending to the mobile devices only relevant data encoded with a lightweight format and to the presence of caching strategies enabling the Mobile Collaboration module to provide several functionalities even in absence of an Internet connection. Finally the user is able to fully exploit the powerful communication features of modern smartphone and tablets as well as available apps in an integrated way with the Innovation Ecosystem Portal data.

#### 6..4.1. Mobile Collaboration module architecture and components

The Mobile Collaboration module is composed by:

- A REST service in charge of interacting with the IEC to exchange and optimize data in order to minimize the required bandwidth and to enhance the presentation of the information on a mobile application.
- An Android mobile application interacting with the REST service enabling the user to exploit all the functionalities of the module.

Figure 28 shows the internal architecture of the module. The Mobile Collaboration REST service is deployed on a server where it is accessible from mobile devices through an Internet connection. This server has full access to the server where the IEC platform is installed (eventually the IEC and the Mobile Collaboration REST service can be installed on the same server). The Mobile Collaboration App is deployed on a mobile device (tablet or smartphone).



**Figure 28: Mobile Collaboration Module Architecture**

The mobile application has an internal modular architecture composed by the following parts.

<b>UI</b>	
Description	This component is in charge of building the User Interface and manages the interactions with the user.
Functionalities / services	<ul style="list-style-type: none"> <li>• UI graphical building</li> <li>• UI event management and update in response of user interactions</li> <li>• UI update in response to notification of other components</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Ecosystem: used to provide ecosystem related functionalities.</li> <li>• Message Board: used to provide message board functionalities.</li> <li>• Calendar: used to provide calendar functionalities.</li> <li>• Preference: used to get/set user preferences.</li> <li>• Android Framework</li> </ul>
User	The user.

<b>Ecosystem</b>	
Description	Provides all the functionalities needed to view the information related to the ecosystem members like companies, members, details, etc.
Functionalities / services	<ul style="list-style-type: none"> <li>• View Ecosystem Members (companies, universities, etc.)</li> <li>• View Ecosystem Members People</li> <li>• View People Contact Details</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Data component: used to save/retrieve local data.</li> <li>• Preference: used to retrieve data about user preferences.</li> <li>• REST Client: used to retrieve data from the IEC.</li> <li>• External App: used to launch external mobile applications to support collaboration and communication.</li> <li>• Android Framework</li> </ul>
User	This module is used by the UI when the user browses the ecosystem information and needs to interact with other members.

Message Board	
Description	Provides all the functionalities needed to view the messages available on the ecosystem message board and to add new messages.
Functionalities / services	<ul style="list-style-type: none"> <li>• View ecosystem board messages</li> <li>• Add new messages</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Preference: used to retrieve data about user preferences.</li> <li>• REST Client: used to retrieve data from the IEC.</li> <li>• Android Framework.</li> </ul>
User	This module is used by the UI when the user checks the ecosystem message board and adds new messages.

Calendar	
Description	Provides all the functionalities needed to view the events available on the Ecosystem shared calendar and to add new events.
Functionalities / services	<ul style="list-style-type: none"> <li>• View ecosystem calendar events</li> <li>• Add new events</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Preference: used to retrieve user preferences.</li> <li>• REST Client: used to retrieve data from the IEC.</li> <li>• Android Framework.</li> </ul>
User	This module is used by the UI when the user checks the ecosystem calendar and adds new events.

Data	
Description	This component manages a local database used when an Internet connection is not available.
Functionalities / services	<ul style="list-style-type: none"> <li>• Create, Read, Update, and Delete local data items.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This module is used by the UI to show local data when an Internet connection is not available and by the REST client to create a local cache.



Preference	
Description	This component manages the user preferences (i.e. the Mobile Collaboration REST service URL or the data synchronization policies).
Functionalities / services	<ul style="list-style-type: none"> <li>Create, Read, Update, and Delete user preferences.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	This module is used by the UI when the user sets his preferences and by all the other components to retrieve user preferences.

External Applications	
Description	This component is in charge of finding external mobile applications to enable the user to perform some actions like to contact a person by skype or to send him an email.
Functionalities / services	<ul style="list-style-type: none"> <li>Find application providing specific functionalities</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	This module is used by the UI when the user needs external applications to carry on his/her activities.

REST Client	
Description	This component is in charge of interacting with the REST service.
Functionalities / services	<ul style="list-style-type: none"> <li>Retrieve ecosystem members (companies, universities, research centres, etc.) data.</li> <li>Retrieve people associated to ecosystem members.</li> <li>Retrieve people contact details.</li> <li>Retrieve ecosystem calendar events.</li> <li>Add ecosystem calendar events.</li> <li>Retrieve ecosystem board messages.</li> <li>Add messages to the ecosystem board.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	This module is used by the other components to interact with the

	IEC through the Mobile Collaboration REST service.
--	--

The application UI is designed to enhance the application usability and to adapt it to the user context. In particular the module exploits some functionalities provided by the multimodal module to sense the user context and to adapt the UI to it (for example the UI is able to adapt to the environment light changing the application colors).

## 6.5. Multimodal Module

### 6.5.1. Introduction

Several studies on Human Computer Interaction focus on innovative user interfaces going beyond the traditional keyboard/mouse input and display based output. Oviatt<sup>35</sup> defines multimodal interfaces as follows: “*Multimodal interfaces process two or more combined user input modes (such as speech, pen, touch, manual gesture, gaze, and head and body movements) in a coordinated manner with multimedia system output. They are a new class of interfaces that aim to recognize naturally occurring forms of human language and behavior, and which incorporate one or more recognition-based technologies (e.g. speech, pen, vision)*”.

In other words multimodal interfaces are based on additional and alternative input/output modes or combined input/output modes (i.e. display + text-to-speech, etc.).

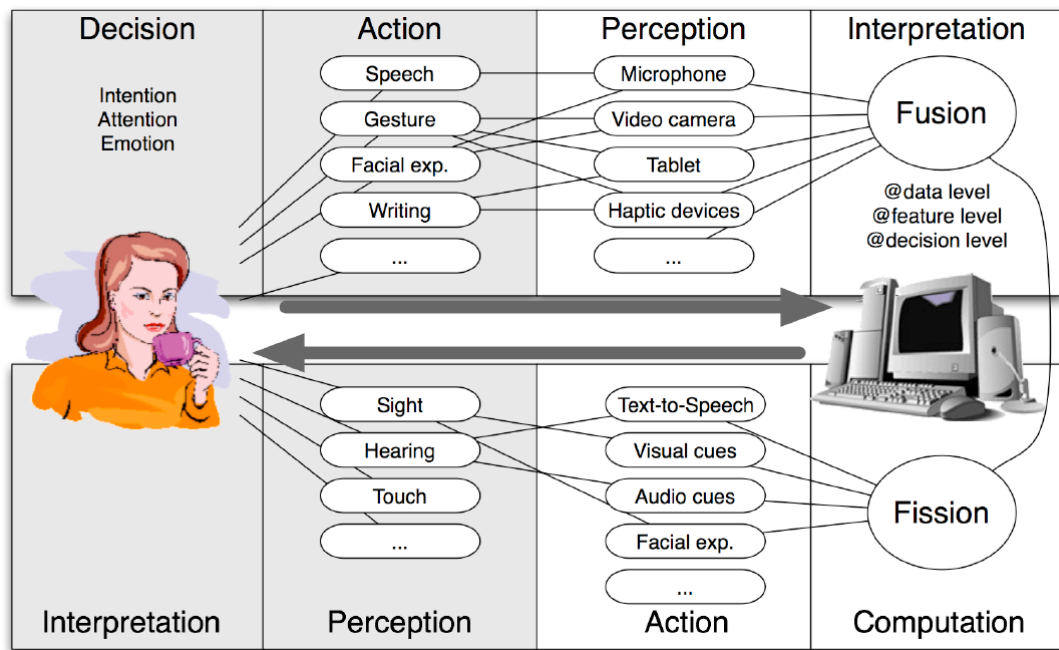
Dumas et al.<sup>36</sup> provide an extended overview of the principles, models and frameworks related to multi-modal interfaces. The goal of multimodal interface is to offer easier and more expressively powerful and more intuitive ways to use computers, enhancing human/computer interaction in a number of ways:

- Enhancing robustness due to combining different partial information input/output sources.
- Ensuring flexible UI personalization based on user and context.
- Providing new functionality involving multi-user and mobile interaction.

Figure 29 shows a representation of multimodal man machine interaction loop. The figure shows all the machine modules used to “perceive” the user input and provide the machine output.

<sup>35</sup> Oviatt, S.L., Advances in Robust Multimodal Interface Design. In: IEEE Computer Graphics and Applications, vol. 23, september 2003 (2003).

<sup>36</sup> Dumas, Lalanne, Oviatt: Multimodal Interfaces: A Survey of Principles, Models and Frameworks Book Human Machine Interaction Pages 3-26 Springer-Verlag Berlin, Heidelberg 2009



**Figure 29: A representation of multimodal man machine interaction loop.**

Not all the modules are present in a specific implementation due to hardware or technical limitations or to the specific application usage. For example a video camera could not be available or could not be used in a specific usage scenario.

Multi-modal interfaces can be applied in many application domains. In the entertainment field the availability of multi-modal interfaces able to acquire the inputs and provide the outputs on multiple channels in parallel can provide an engaging user experience. For example a fighting game could acquire input and react to the user facial expressions, to the user speeches, to the user body movements and so on. At the same time the game could provide output through a display, a sound surrounding system, some dedicated lights in the room, a vibrating gamepad etc. The goal in this case is to involve the user in the game experience by capturing his attention as much as possible.

In the manufactory domain the requirements are different because users move in industrial environments which may be dangerous and are often involved in manual activities. Software applications must be designed to not distract the user attention from the task he is performing. In this context a real improvement of the system usability would be a hands-free (and eyes-free) communication approach. This rationale has been confirmed by some interviews to MSEE end-users and to technological partners having customers in the manufactory domain.

Let's consider a concrete example of a mobile application supporting an operator in a maintenance procedure for a certain industrial machine. With a traditional interface the user must read step-by-step instructions on the display and apply these instructions performing specific actions on the machine. To read the instructions can be difficult due to the display size and/or to the ambient light and the user is requested to look alternatively the mobile device screen and the machine (attentional shift). After the completion of each step the user must use the device keyboard to input some notes and must press the "next" button to go on to the next step.

With a multimodal user interface the user can listen instructions from the device speakers (or using a wireless earphone), then he can dictate notes instead of writing them and finally he can go on to the next step with a simple gesture.

These considerations are very relevant for mobile devices, like smartphones, having limited interaction capabilities due to the available space for screens and (virtual) keyboards. Up to now the lack of computational power has restricted the implementation of multimodal interfaces on the cell phones. For example it was not possible to deploy a speech recognition engine on a cell phone and it was slow to connect to online services. Several research papers

describes multimodal distributed systems that in few cases have been developed in commercial products due to the lack of fast and reliable Internet connections.

Today these limitations have been overcome thanks to the availability of powerful mobile CPUs and of reliable fast internet connections on modern smartphones. Moreover smartphones thanks to their sensors are able to better exploit contextual information and to allow new user interactions ways.

The following table reports some relevant features of mobile devices for the development of multimodal user interfaces.

Feature	Usage
Capacitive Touch Screen	Capacitive touch screens enable to acquire user inputs easily. On these screen it is possible to present traditional QWERTY keyboards, but also keyboards customized to accomplish specific tasks and numeric keyboards. Moreover touch screens can be used to recognize user gestures on the screen. Finally these screens are very bright and their brightness can be adapted to the environment light. Finally the resolution of these screens enables to provide high quality images that can be zoomed.
Microphone	The microphone can be used to detect the ambient noise or to perform speech recognition.
Built-in speaker	The built-in speaker can be used to drive user actions by vocal commands (when the user cannot look at the device) or to play meaningful sounds.
Camera	The camera can be used to “look around” applying image recognition algorithms to the user environment.
Ambient light sensor	The ambient light sensor can be used to adapt the behavior and look & feel of the application to the ambient brightness.
Accelerator	Using this sensor it is possible to sense some user actions like if the user shakes the phone or rotate it. Moreover it is possible to understand if the user is moving or he is standing still.
Rotation motor	Several messages can be sent to the user through well-defined vibrating patterns (haptic feedback).

**Table 5: Mobile devices relevant features for multimodal applications**

As described in the following Section the multimodal module exploits several features presented in the table above to support the implementation of mobile applications that can be used in a hands-free/eyes-free way thanks to the availability of a multimodal interface.

### 6..5.2. Multimodal Module Description

The goal of the Multimodal module is to support the development of applications in the manufactory domain that can be used in hands-free (and eyes-free) way. To achieve this goal some mobile device features play a primary role (i.e. microphone and rotation motor) while others play a secondary role (i.e. capacitive touch screen).

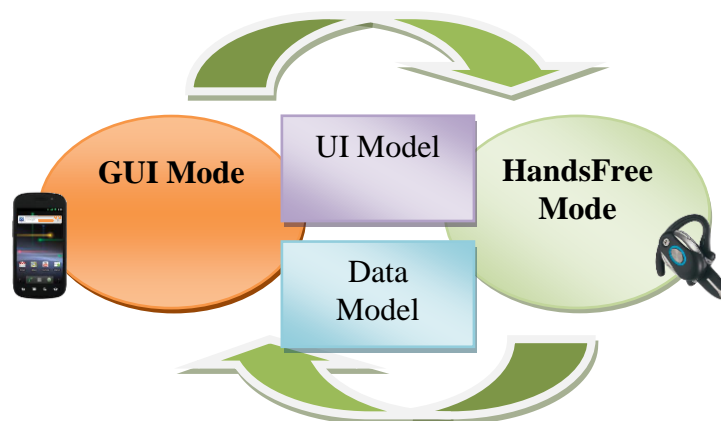
The concepts behind the module can be implemented with any modern mobile programming language. However, we have decided to provide a reference implementation for the Android devices (see Section 2.2).

The module is provided with a sample application, exploiting and demonstrating it, namely Handsfree Maintenance App. The application enables technician to follow maintenance procedures and to save some notes. For example an operator can follow a procedure for the ordinary maintenance of an engine and can save several values like the general aspect of the engine (i.e. if it has blighted parts), the current temperature and the max temperature recorded during the maintenance procedure or the minimum round for minutes (RPM). This application is interesting for MSEE end users (in particular Indesit and Ibarmia) as well as for other partners (i.e. Engineering, Softeco, Hardis, etc.) that can consolidate it in a commercial product.

The multimodal module supports the development of applications providing two user interfaces (UI):

- Graphical User Interface (GUI): the user exploits the mobile device screen and keyboard to interact with the application.
- Hands-Free User Interface (HF): the user interacts with the application in a hands-free and eyes-free way.

To handle these two UIs the module provides suitable functionalities to maintain a UI model that is rendered in two different ways according the application mode (GUI or HF). Moreover the module offers suitable mechanisms to synchronize the UI model with the application data model.



**Figure 30: Application modes.**

For example let's suppose that the application enables the user to input a "serial number" and a product "model".

In the GUI mode two editable text fields are shown on the phone screen. When the user provides some values they are saved by the application as attributes of the "product" bean.

In the HF mode the module enables the user to listen a description of the value that he must provide (i.e. serial number and model), to select the value that he wants to provide at a certain point, to input the value by voice and to have a feedback from the application on the

recognized value (i.e. the *application* has understood “ZV Classic 45”). Things become more complex as the number of fields increases and their type is heterogeneous (text, numbers, dates, etc.).

We have decided to use an earphone and the media control button available on this device (see Figure 31) in order to:

- Start the voice recognition process. The speech recognition engine used in the module, as almost all recognition engines, does not recognize speech in a continuous way but it needs a user input to understand when to start to recognize speech. Then the recognition process is stopped when the user stops to speech.
- Enable the user to navigate into the application UI (i.e. to change field, area, etc.)
- Listen application output provided by the a text-to-speech engine.

Figure 31 shows wired and wireless earphones where the media control button has been highlighted. The last picture shows the Sony Smart Watch <sup>37</sup> providing an additional UI to Android devices and that can be used to show the media button while an earphone is still required to listen to the application audio and provide vocal input.



**Figure 31: Media button control.**

The multi-modal module interprets a certain event from the number of presses of the media button according the following semantic:

Number of consecutive presses	Event
<b>1</b>	<b>Next</b>
<b>2</b>	<b>Activate</b>
<b>3+</b>	<b>Previous</b>

The effect of the event can be customized by the application exploiting the module. By default the Next/Previous events causes the application in the HF mode to go to the next/previous item like for example the next/previous maintenance procedure step or the next/previous input field (the opposite for the Previous). The Activate event starts the speech recognition engine enabling the user to input a value if an input field is selected. The events are ignored if the user is using the application GUI.

As said before the module in the HF mode exploits the Android Speech Recognition (ASR) service and the media button control to acquire the user input. The Android TextToSpeech engine (TTS) and the phone vibrator motor (Haptic feedback) are used to provide the user output.

The module handles the GUI mode using the phone screen and (virtual) keyboard to provide output and acquire input. The module is able to sense the user context (i.e. noise, light, if the user is moving, etc.) to suggest to the user the best mode to use at a certain time.

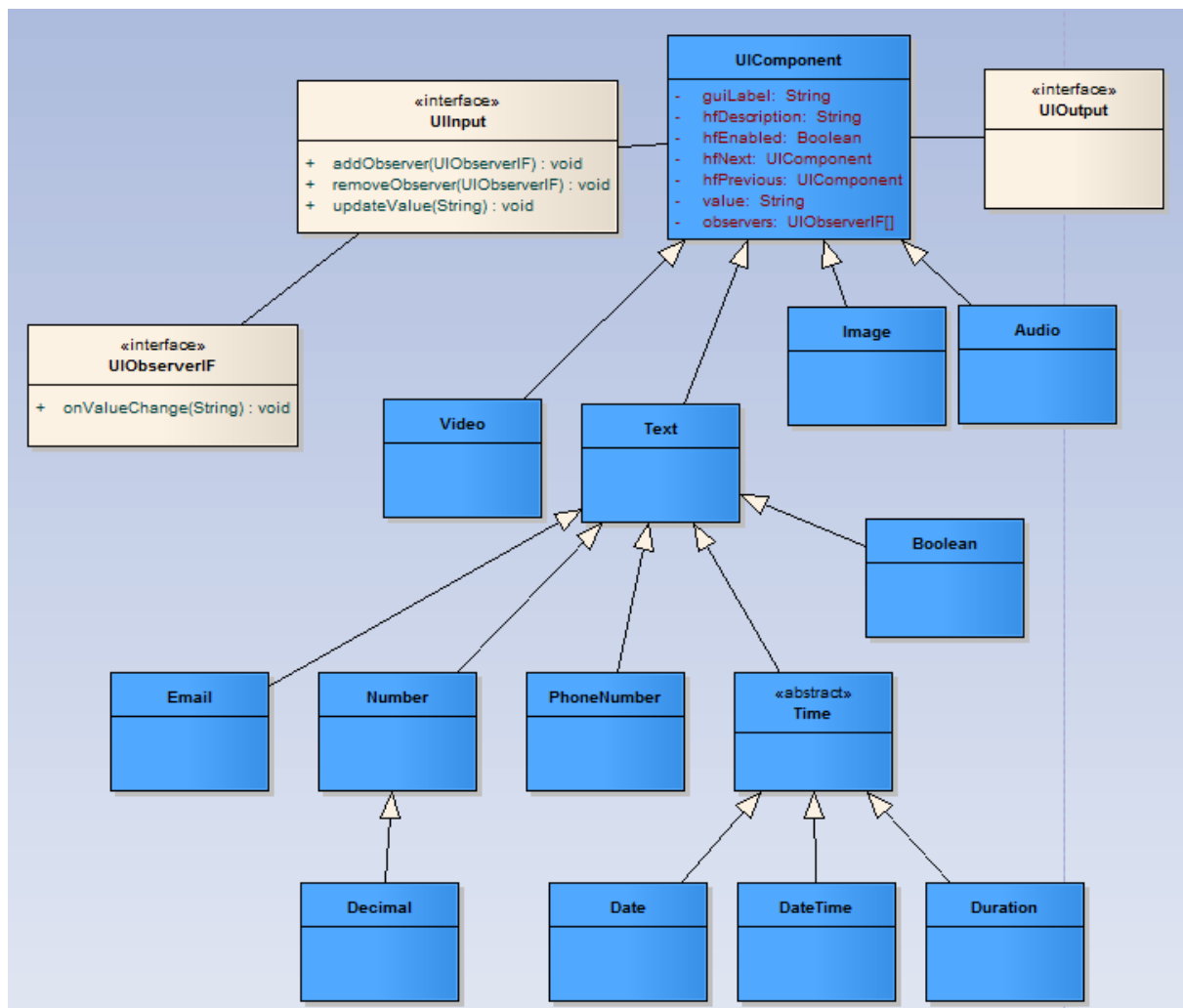
Figure 32 depicts the UI model enabling the module to render the user interface in the GUI and in the HF mode. All the components belonging to the UI inherit from the UIComponent

<sup>37</sup> <http://www.sonymobile.com/global-en/products/accessories/smartwatch/>



base class and share the following set of attributes (peculiar attributes of the subclasses are not reported in the figure for simplicity):



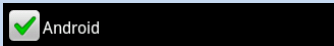

- guiLabel: label of the component in the GUI mode (i.e. “model:” ).
- hfEnabled: is true if the component must be managed in the HF mode (i.e. a long description can be disabled when in HF mode while the user can read it on the GUI if needed).
- hfDescription: description of the component in the HFmode (i.e. “ product model”).
- value: value of the UIComponent. The value can be modified or inserted by the user.
- observers: objects that must be notified when the value changes.
- hfNext: next UI component to be used in the HFmode.
- hfPrevious: previous UI component to be used in the HFmode.

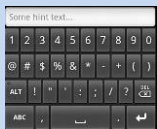


**Figure 32: UI model.**

A UIComponent can implement the UIInput interface. In this case the user is able to provide a value for the component and each time the value is modified a set of observers, in charge of modifying the application data model, are notified. In alternative a UIComponent can implement the UIOutput interface. In this case the value of the component is provided to the user that can not modify it. The UIComponent sub classes characterize various kinds of input/output components rendered and managed in different ways as summarized in the following table.

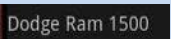
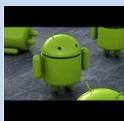




UIInput Component	GUI	HANDS-FREE
Text Email PhoneNumber	<p>The component is rendered as an <i>Edit Text</i> customized to show a virtual keyboard enabling the insertion of text (the keyboard is also customized for Email and Phone Numbers).</p> 	<p>The HF description is readed to the user exploiting the TTS engine and the input is acquired exploiting the ASR service. The recognized value is repeated to the user exploiting the TTS for confirmation. Validation rules are checked and eventual errors reported to the user using the TTS.</p> <p>This procedure, namely Text HF procedure, is the same for all the UI components and is not reported in each row.</p>
Number Decimal	<p>The component is rendered as an <i>Edit Text</i> customized to show a virtual keyboard enabling the insertion of numbers including a decimal part (only for decimal).</p> 	<p>In addition to the Text HF procedure the recognized string is processed to check if it represents a number. If a number is not recognized the TTS is used to suggest to the user how to pronounce a number correctly.</p>
Boolean	<p>The component is rendered as a Check Box.</p> 	<p>In addition to the Text HF procedure the recognized string is processed to check if it represents a Boolean value. If a Boolean value is not recognized the TTS is used to suggest to the user how to pronounce a Boolean correctly (yes/no). The module improves user input if needed (i.e. “ye” is transformed in “yes”).</p>
Date DateTime	<p>The component is rendered as a Date Picker and a Time Picker.</p> 	<p>In addition to the Text HF procedure the recognized string is processed to check if it represents a date/time value.</p> <p>If a date/time is not recognized the TTS is used to suggest to the user how to pronounce a date/time. The module improves user input if needed (i.e. “today/yesterday” are transformed in the right dates, “now” is transformed in the current date/time, etc.).</p>

Duration	<p>The component is rendered as an <i>Edit Text</i> customized to show a virtual keyboard enabling the insertion of text. When a value is inserted it is checked if it is in the format “hour:minutes” (i.e. 03:22 means 3 hours and 22 minutes).</p> 	<p>In addition to the Text HF procedure the recognized string is processed to check if it represents duration of time. If a duration is not recognized the TTS is used to suggest to the user how to pronounce it.</p>
Image	<p>The photo camera application of Android (or another app installed on the phone) is used to enable the user to take a photo.</p>	<p>The input cannot be acquired in this mode. The TTS is used to suggest to the user to switch to the GUI mode.</p>
Video	<p>The video camera app of Android (or another app installed on the phone) is used to enable the user to record a video.</p>	<p>The input cannot be acquired in this mode. The TTS is used to suggest to the user to exploit the GUI interface.</p>
Audio	<p>The Audio Record app of android (or another app installed on the phone) is used to enable the user to take a video.</p>	<p>The TTS is used to read the hf_description of the field. When the user activates the field its voice is recorded (not recognized) until he stops speaking.</p>

**Table 6: Multimodal input UI components**

The following table summarizes how the components are rendered if they implement the UIOutput interface.

UIOutput Component	GUI	HANDS-FREE
Text Email PhoneNumber Number Decimal Date DateTime Duration Boolean	<p>The component is rendered as an Android <i>TextView</i>.</p> 	<p>The hf_description is readed to the user exploiting the TTS engine.</p>
Image	<p>The component is rendered as an Android <i>ImageView</i>.</p> 	<p>The hf_description is readed to the user exploiting the TTS engine.</p>

Video	<p>The Android Media Controller is used to reproduce the video.</p> 	The hf_description is readed to the user exploiting the TTS engine.
Audio	<p>The Android Media Controller is used to play the audio.</p> 	The audio is reproduced using the earphones.

**Table 7: Multimodal output UI components**

The Multimodal Module offers the following set of functionalities:

- UI Model Building: the module enables to build the specific UIModel of an application (or part of it).
- GUI Building and Management: the module provides all the functionalities to build a graphical user interface rendering the application UI Model to be used when the application is in the GUI mode. The built GUI is able to check some input validation rules.
- HandsFree Navigation: the module enables the user to navigate between the UI components using the media button control. The developer can also implement and enable other kinds of inputs (i.e. to shake the phone to go to the next UI component, etc.).
- HandsFree Input Management: the module enables the user to provide input by voice, checks validation rules (i.e. checks that the user has pronounced a number for a Number UI component) and improves the user input (i.e. translates the value “today” in the appropriate date for a Date UI component)
- HandsFree Output Management: the module uses the TTS engine to provide vocally the descriptions contained in the hfDescription attribute.
- Validation Rules Check: the module is able to check the following validation rules on user input (not all the rules can be applied to all the types): required input, minimum/maximum length, value greater/lesser than.

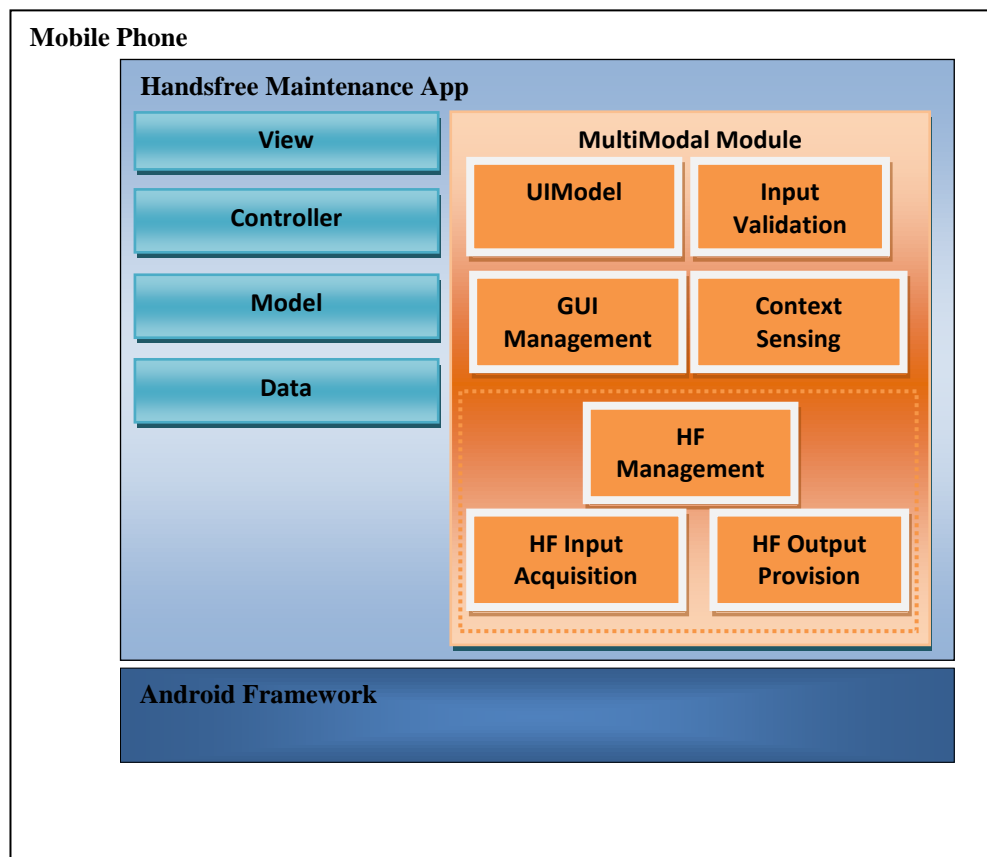
### 6..5.3. Multimodal Module Internal Architecture

Figure 33 shows the internal architecture of the Hands Free Maintenance App including its main components and the internal structure of the Multimodal module.

The scope of the Hands Free Maintenance App is to show how it is possible to apply the Multimodal Module to a real mobile application. The application enables maintainers to follow maintenance procedures in a hands-free way. In particular the maintainer is able to select the maintenance procedure to be followed, to retrieve information on each step and to save some data to provide a report on the results of the maintenance activity performed.

The Hands Free Maintenance App architecture is based on the MVC pattern:

- **Model:** includes the classes representing the application model like for example the classes representing a maintenance procedure, a maintenance procedure step, etc.
- **View:** includes the Android layout files representing the UI interface of the applications for the parts not managed with the Multimodal module (i.e. home page, preference page, etc.).
- **Controller:** includes all the classes managing user interactions and exploiting the Multimodal Module.
- **Data:** includes the classes in charge of saving and retrieving data from the internal database and or from external data sources like web services.



**Figure 33: Multimodal Module Internal Architecture**

The following tables present a high level description of the Multimodal Module components. In the table the terms “*application*” refers to an Android application, like the Handsfree Maintenance App, exploiting the multimodal module to provide a handfree user interface.

UI Model	
Description	This component provides all the classes and interfaces to represent the user interface both in the GUI and HF modes.
Functionalities / services	<ul style="list-style-type: none"> <li>• Enable <i>application</i> to model the user interface.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This component is used by the <i>application</i> to build a model of user interface that can be managed in a graphical (GUI) and in a non-graphical way (HF).

Input Validation	
Description	This component is in charge of checking the user input applying some validation rules that can be assigned to user input components.
Functionalities / services	<ul style="list-style-type: none"> <li>• Check validation rules.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This component is used by the GUI Management component and by the HF Input Acquisition component to validate user input.

Context Sensing	
Description	Sense the user context (noise, light, user moving, etc. ) in order to adapt and suggest the best user interface. Additional details are provided in Section 6..5.4
Functionalities / services	<ul style="list-style-type: none"> <li>• Sense Brightness</li> <li>• Sense Noise</li> <li>• Sense Location</li> <li>• Sense Motion</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This module is used by the <i>application</i> , by the GUI Management and the HF management to adapt the UI and/or to suggest to the user the best mode (HF/GUI) to use.

GUI Management	
Description	Builds and manages the graphical user interface of the <i>application</i> starting from the model provided by the UIModel component.
Functionalities / services	<ul style="list-style-type: none"> <li>• Build the user interface</li> <li>• Manages user interaction in the GUI mode</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• UIModel: used as underground UI model.</li> <li>• Input Validation: used to check user input.</li> </ul>
User	This module is used by the <i>application</i> when the user uses it in the GUI mode.

HF Management	
Description	Builds and manages the user interface of the <i>application</i> when in HF mode starting from the model provided by the UIModel component. Includes all the classes to provide the navigation of the UI in the HF mode.
Functionalities / services	<ul style="list-style-type: none"> <li>• UIModel: used as underground UI model.</li> <li>• Input Validation: used to check user input.</li> <li>• HF Input Acquisition: used to acquire user input.</li> <li>• HF Output Provision: used to provide output.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• Android Framework</li> </ul>
User	This module is used by the <i>application</i> when in HF mode.

HF Input Acquisition	
Description	This component is in charge of acquiring the user input, improve it (i.e. transform “today” into “15/06/2013”) and provide to the user a feedback on the recognized speeches.
Functionalities / services	<ul style="list-style-type: none"> <li>• Acquire user input</li> <li>• Improve user input.</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>• UI Model: to manage input data.</li> <li>• Android Framework</li> </ul>
User	This module is used by the HF Management component to acquire user input.

HF Output Provision	
Description	Provides output to the user exploiting the Android TextToSpeech engine and vibration motor.
Functionalities / services	<ul style="list-style-type: none"> <li>Provide user output</li> </ul>
Dependencies	<ul style="list-style-type: none"> <li>Android Framework</li> </ul>
User	This module is used by the HF Management component to provide user output.

The Context Sensing component offers several features, which are described in the following Section, usable in many applications to adapt the UI to the user context.

#### 6..5.4. Context Sensing

The context sensing functionalities are used by the multimodal module to suggest to the user the best UI to use. For example if the user is in a noisy environment the module suggests to the user to exploit the GUI model because the Android Speech Recognition could be inaccurate. However the context sensing functionalities can be used in a wide range of applications for different purposes (i.e. to estimate the user location or to interpret user behavior, etc.). The module offers the following main features:

Feature	High level description
<b>Sense Brightness</b>	The device light sensor is used to estimate the environment brightness.
<b>Sense Noise</b>	The device microphone is used to estimate the environment noise.
<b>Sense Location</b>	The device position features (i.e. GPS, network information, etc.) are used to estimate the location of the device.
<b>Sense Motion</b>	The device accelerometer is used to estimate if the device is standing or not.

Several features are based on the use of the device sensors. Despite the usefulness of these sensors to use them is a power-wasting activity so it is not possible to use them in a continuous way to get real time information. For example to get information on the environment noise it is possible to activate the microphone for some seconds but not to use it continuously.

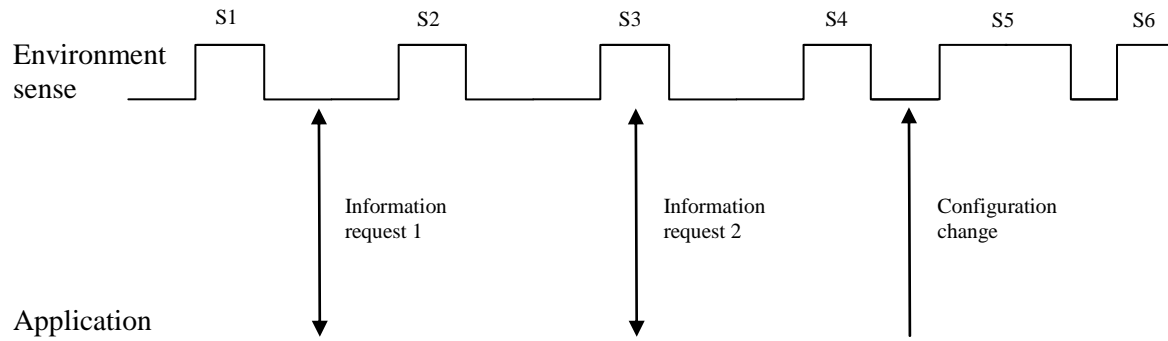
On the other side to retrieve information on the user context is needed only at a certain time and the accuracy of information depends on this period. In the example before to activate the microphone for 10 seconds could enable the component to understand that the user is in a quiet environment where sometime there is some noise, while to activate the microphone for only 1 second when there is some noise could get to a misleading result.

To summarize, the quality of achieved results and the required battery power are influenced by two factors:

- Sense Frequency: how often sensors are used to perceive the environment and the user actions.
- Sense Duration: how long sensors are used each time the user context must be sensed.




Another thing to consider is that different applications require different accuracies and the same application can require different accuracies at different times according the user task. The context sense component enables applications to modify the component configuration at run-time optimizing the trade-off between result accuracy and power consumption.



**Figure 34: Application and environment sensing**

Figure 34 shows an example of interactions between an application and the Multimodal module. The module senses the environment with a given frequency and for a certain period (s1, s2, s3, s4).

The application requires the information on the environment each time it needs them (for example in response to certain user actions). The module always returns the most updated available information (i.e. at the information request 2 are returned the information collected during s2 since the information that the module is collecting in s3 are not still available). Finally the application can change the module configuration. In the figure the application improves the accuracy of information increasing the frequency and the sensing period (S5 is longer than S4 and the time between S6 and S5 is lower than the time between S4 and S3) .

Project ID <b>284860</b>	MSEE – Manufacturing Services Ecosystem	
Date: <b>30/06/2013</b>	Deliverable D44.2 Mobile Business platform specifications and architecture – M21 issue	

## 7. Conclusions

This deliverable consists in an overview of the Mobile Platform architecture and presents the final specification of the platform and of its modules.

As specified in the MSEE description of work and to achieve its goals, the Mobile Platform addresses several heterogeneous topics starting from mobile development and delivery to ambient intelligence, multimodal interfaces and mobile collaboration. Due to the heterogeneity of the topics and goals addressed, the platform has been designed as a set of modules. Each module has a particular internal architecture and exploits some state-of-art technologies in order to achieve its own goal and enhance the communication and interoperability with other modules, other MSEE software systems and external applications (all the modules are re-usable by third party external applications and use standard data formats to exchange data).

The Mobile Platform addresses smartphones and tablets already available on the market and exploits their appealing features and sensors. However the platform is designed and implemented to be extensible as soon as new devices appear on the market introducing new features. For example the Ambient Intelligence module is designed to exploit the Near Field Communication technology, which is expected to significantly grow in the coming months.

The specification of the modules will be used to develop the final prototype of the Mobile Platform. While early prototypes of the modules will be shown during MSEE meetings and other events to collect end-user feedback, the Mobile Platform final prototype will be tested extensively by MSEE end-users in the framework of MSEE pilotings (WP61, WP62, WP63 and WP64).