

ALMA MATER STUDIORUM  
UNIVERSITY OF BOLOGNA

---

ARCES - ADVANCED RESEARCH CENTRE ON ELECTRONIC SYSTEMS  
FOR INFORMATION AND COMMUNICATION TECHNOLOGIES E. DE CASTRO

# Architectures for Context Aware Services in Smart Environments

Daniele Manzaroli

*PhD Coordinator*

Prof. Claudio Fiegna

---

*Supervisor*

Prof. Tullio Salmon Cinotti

---

PHD. THESIS  
January, 2007 - December, 2009

PHD PROGRAM IN INFORMATION TECHNOLOGY

---

CYCLE XXII - ING-INF/05

With a special feeling of gratitude, I dedicate this work to my family for all the love given to me



## **Keywords**

Context Awareness

Smart Environment

Interoperability

Architectures for cross-domain applications

Platforms for interoperable services



# I – TABLE OF CONTENTS

II – INDEX OF FIGURES .....	8
1.1    THESIS STRUCTURE .....	11
2 AN ARCHITECTURE FOR INTEROPERABILITY: REQUIREMENTS AND PRINCIPLES.....	12
3 RELATED WORK.....	18
3.1    COMPARISON OF INTEROPERABILITY MODELS.....	18
3.2    APPLICATION DOMAINS .....	20
3.3    CONTEXT.....	22
3.4    CONTEXT-AWARE ARCHITECTURES .....	24
4 RESEARCH FRAMEWORKS .....	26
4.1    EUROPEAN NETWORK OF EXCELLENCE ON CULTURAL HERITAGE (EPOCH) .....	27
4.2    JOINT RESEARCH TELECOM ITALIA LAB .....	28
4.3    EUROPEAN TECHNOLOGY PLATFORM ON EMBEDDED SYSTEM (SOFIA).....	28
5 ARCHITECTURES AND PLATFORMS .....	30
5.1    MULTIMEDIA GUIDE FOR MUSEUMS AND ARCHAEOLOGICAL SITES.....	31
5.2    A CONTEXT-AWARE PLATFORM FOR CULTURAL HERITAGE APPLICATIONS AND SERVICES..	33
5.2.1 <i>MobiComp: a context management solution for Cultural Heritage</i> .....	35
5.2.2 <i>Device and language interoperability for MobiComp</i> .....	38
5.2.3 <i>Multi programming languages: MobiComp Enabler Library</i> .....	40
5.2.4 <i>Context-aware application broker</i> .....	42
5.2.5 <i>FEDORA content management system</i> .....	44
5.2.6 <i>Tool chain for Cultural Heritage sites</i> .....	45
5.2.6.1    Monitoring, Tracking guiding and data collection service .....	46
5.2.6.2    Access detection service .....	47
5.2.6.3    Presence monitoring service.....	49
5.2.6.4    Statistics on visitors flow .....	51
5.2.6.5    People tracking service.....	52
5.2.6.6    Pedestrian navigation support .....	53
5.3    USER PREFERENCES IN SERVICE PLATFORMS.....	55
5.3.1 <i>Preference based information processing model</i> .....	57
5.4    AN INTEROPERABILITY PLATFORM FOR SMART SPACES.....	59
5.4.1 <i>Smart-M3 and interoperable context management solution</i> .....	61
5.4.2 <i>Integrated service and information interoperability</i> .....	63
5.4.2.1    Application of Integrated Service and Information Interoperability .....	67
5.4.3 <i>Service discovery and access control proposal</i> .....	70
6 CONCLUSIONS .....	74
PUBLICATIONS .....	80
BIBLIOGRAPHY .....	82
TOOLS .....	90
ACRONYMS .....	92
ACKNOWLEDGEMENTS .....	94



## II – Index of Figures

FIGURE 1: INTEROPERABILITY ENABLING ARCHITECTURE COMPONENTS. APPLICATIONS OR SERVICES FOR SE COULD BE MADE BY EXPLOITING ONE OR MORE OF THESE COMPONENTS .....	13
FIGURE 2: THE CLASSIFICATION FRAMEWORK OF CONTEXT-AWARE SYSTEMS [21] .....	25
FIGURE 3: TWO WHYRE VIEWS (LEFT AND CENTRE). ON THE RIGHT A MUSEUM USE CASE WHERE A VISITOR WEARING THE WHYRE CAN ENJOY MULTIMEDIA CONTENTS WHILE SENSORS LOCALIZE HIM IN TERMS OF POSITION AND ORIENTATION .....	31
FIGURE 4: MINIMALISTIC ARCHITECTURE FOR AN APPLICATION SPECIFIC SYSTEM .....	32
FIGURE 5: SCHEMATIC VIEW OF THE ARCHITECTURE ADOPTED IN EPOCH. IT PROVIDES CONTEXT, PREFERENCES, PROFILE AND CONTENT MANAGEMENT. PART OF THE ARCHITECTURE WAS DEDICATED TO SERVICE MANAGEMENT FOR THE END USER .....	34
FIGURE 6: MOBIComp ARCHITECTURE .....	35
FIGURE 7: ABSTRACT VIEW OF A MOBIComp APPLICATION: AN AGGREGATOR PRODUCES NEW CONTEXT DATA BY PROCESSING ALREADY AVAILABLE INFORMATION; A LISTENER GETS CONTEXT INFORMATION FROM THE CONTEXT STORE, E.G. FOR MONITORING PURPOSES.....	37
FIGURE 8: MMPI ARCHITECTURE.....	38
FIGURE 9: MMPI MESSAGE TRACKING UML SEQUENCE DIAGRAM .....	39
FIGURE 10: MOBIComp WRAPPER MODEL. NON-JAVA APPLICATIONS CAN USE MOBIComp WITH THIS METHOD .....	40
FIGURE 11: LIST OF MOBIComp ACTORS CLASSES .....	41
FIGURE 12: CAB ARCHITECTURE ENHANCED BY THE CONTEXT ACCESS SERVICE PROVIDED BY MMPI. THE CAB SERVICE IS CUSTOMIZED ON THE CLIENT SIDE BY USING USER PREFERENCES, PROFILE AND CONTEXT. IT IS BASED ON THE HTTP COMMUNICATION PROTOCOL .....	42
FIGURE 13: SCHEMATIC VIEW OF THE FINAL ARCHITECTURE DEVELOPED UNDER THE EPOCH PROJECT	44
FIGURE 14: STRUCTURE OF A GENERIC MOBIComp APPLICATION. ACTORS MEANS MOBIComp JAVA CLASSES THAT IMPLEMENT A LISTENER, A TRACKER OR AN AGGREGATOR.....	45
FIGURE 15: LAYERED STRUCTURE OF A “VISITOR GUIDE” CIMAD APPLICATION. THIS IS A GENERIC EXAMPLE ON HOW ALL COMPONENTS CAN BE PUT TOGETHER TO BUILD A CIMAD APPLICATION FOLLOWING THE MVC PATTERN.....	46
FIGURE 16: ON THE LEFT THE STEREO CAMERA FIELD-OF-VIEW. ON THE RIGHT THE STEREO CAMERA INSTALLED IN A MUSEUM ACCESS GATE (STH-MDCS2-C VIDERE DESIGN).....	47
FIGURE 17: VTS PROGRAM SCREENSHOTS. PEOPLE UNDER THE STEREO CAMERA ARE TAGGED WITH A COLORED CUBE. THE VTS COUNTS THE PEOPLE CROSSING THE GREEN VIRTUAL LINE.....	48
FIGURE 18: INTEGRATION OF THE STEREO VISION BASED TRACKING SYSTEM.....	49
FIGURE 19: REAL TIME VISITORS FLOW INDICATOR.....	49
FIGURE 21: PEDESTRIAN NAVIGATION SUPPORT: A USER WITH A MOBILE DEVICE - ENRICHED WITH A WIRELESS INERTIAL MOTION SENSOR BOARD –ENJOYS THE PEDESTRIAN NAVIGATION SERVICE . THE PATH TO THE TARGET AND THE DIRECTION TO FOLLOW ARE SHOWN (SCREENSHOT ON THE RIGHT).....	54
FIGURE 22: ARCHITECTURE COMPONENTS FOR A PREFERENCE AND CONTEXT BASED RECOMMENDATION SERVICE.....	55
FIGURE 23. CONTEXT MANAGEMENT ARCHITECTURE [75] .....	56
FIGURE 25: AN ARCHITECTURE EXAMPLE WHERE INFORMATION SEMANTICS IS HANDLED UNDER ONTOLOGIES CONSTRAINTS. THIS IMPROVES THE INTEROPERABILITY AT INFORMATION LEVEL .....	60
FIGURE 26: SMART-M3 FUNCTIONAL AND LOGICAL ARCHITECTURE.....	61



FIGURE 27: OVERVIEW OF THE OSGi AND SMART-M3 INTEGRATION .....	65
FIGURE 28: INTEGRATED SMART-M3/OSGi SOLUTION .....	66
FIGURE 29: SYSTEM ARCHITECTURE OF THE MAINTENANCE SCENARIO DEMONSTRATION .....	68
FIGURE 30: SOFTWARE ARCHITECTURE OF THE MAINTENANCE SCENARIO DEMONSTRATION IN TERMS OF KPS .....	69
FIGURE 31: SNAPSHOT OF THE ONTOLOGY USED FOR THE MAINTENANCE SCENARIO.....	69
FIGURE 32: THE ENVIRONMENT CAN BE SPLIT IN MANY SEs. EVERY SE CAN CONTAIN ONE OR MORE SS, DEPENDING ON THE USE CASE. ....	70
FIGURE 33: AN EXAMPLE ON HOW TO MODEL THE TOPOLOGY OF THE AVAILABLE SSS AND CORE SERVICE .....	71
FIGURE 34: ARCHITECTURE COMPONENTS FOR EXTENDED INTEROPERABILITY .....	75
FIGURE 33: FROM DEDICATED CONTEXT-AWARE PLATFORMS TO INTEROPERABLE SMART SPACES FOR MULTI-DOMAIN CONTEXT AWARE SERVICES .....	76

# Chapter 1

## Introduction

Technology provides power to modern devices. Power needs control and control requires an *infrastructure*.

Examples of modern devices relying on an infrastructure include smart phones, PDAs, RFID tags and readers but also sensing devices spread in the environment such as temperature, humidity and light sensors, inertial sensors, camera and many others. They are integrated through an *infrastructure* to impact people behaviour and life style.

The infrastructures are architecture implementations; they enable the just mentioned technology products to interact and co-operate, share and exchange information and they behave like a Service Oriented Architecture (SOA); the services provided have the ability to adapt to the user situation, profile and preferences; in other words they are context-aware services.

In my thesis I will review the history of my research activity on interoperable context-aware computing. At the beginning interoperability was not considered an issue, as the focus was on using sensors in a specific application with a specific device, namely guiding museum and archaeological sites visitors with a context-aware multimedia guide.

Then the context-awareness concept was extended to the cultural heritage domain aiming to multiple services for many user profiles in cultural heritage applications. This led to two requirements: the need to share among multiple services information originating from sensors and the devices spread in the environment, and the need for service interoperability across multivendor devices.

Eventually the focus was further extended beyond cultural heritage towards multi-domain and also cross-domain applications (such as for example, personal healthcare, city and domestic smart services), bringing in a clear requirement for information level interoperability, which is considered the enabling factor for a new industry of context-aware multi-domain services. In fact, an architecture that supports interoperability of information originating from the environment will allow to develop applications easily without the constrain of knowing everything about the new technologies involved in any new project.

A lot of research worldwide is currently investigating architecture solutions to support inter-communication and data sharing between multi-vendor devices. Interoperability is usually considered the ability of heterogeneous devices to interoperate – i.e. to interact and exchange semantically meaningful information – without the need to a-priori know each other communication protocols and information representation models.

My research activity in this area was carried out at the University of Bologna and partially at VTT (Technical Research Centre of Finland) in Oulu, mostly within the framework of European Projects and Networks of Excellence, but also within joint research projects between the University of Bologna and the Italian industry.

## **1.1 Thesis structure**

The rest of this document is organized as follows. In chapter 2 research are and the principles that drive the research are shown. Related work and the state of the art are described in chapter 3. Chapter 4 is dedicated to detail projects and networks associated to my work. All work done is detailed in chapter 5. In chapter 6 conclusion are drawn. Chapter 7 is dedicated to my publications. The list of references is reported in chapter 8. Design time tools and acronyms are explained respectively in chapter 9 and 10.

# Chapter 2

## An architecture for interoperability: requirements and principles

Provide a zero-effort interface between machine and human is one of the key aspects of this research. The goal is to achieve a smooth interaction – for both end-user and developers – based on human desires and not on the machine world rules for a novel form of Human-Computer Interaction (HCI)<sup>1</sup>.

Smart Environment (SE) is intended as an environment with an associated *digital representation* called Smart Space (SS) [1] that is a named search extent of information. In a SE it should be possible – by using an own personal mobile device – to share context information<sup>2</sup>. Both content and service fruition should be possible and they can be both context and device performance driven. Innovative applications could be developed because developers have more freedom by operating on ready to use infrastructure. These applications will be called Smart Applications (SA).

---

<sup>1</sup> **Definition:** *Human-Computer Interaction* (HCI) is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. -- 1992, SIGCHI Curriculum Development Group

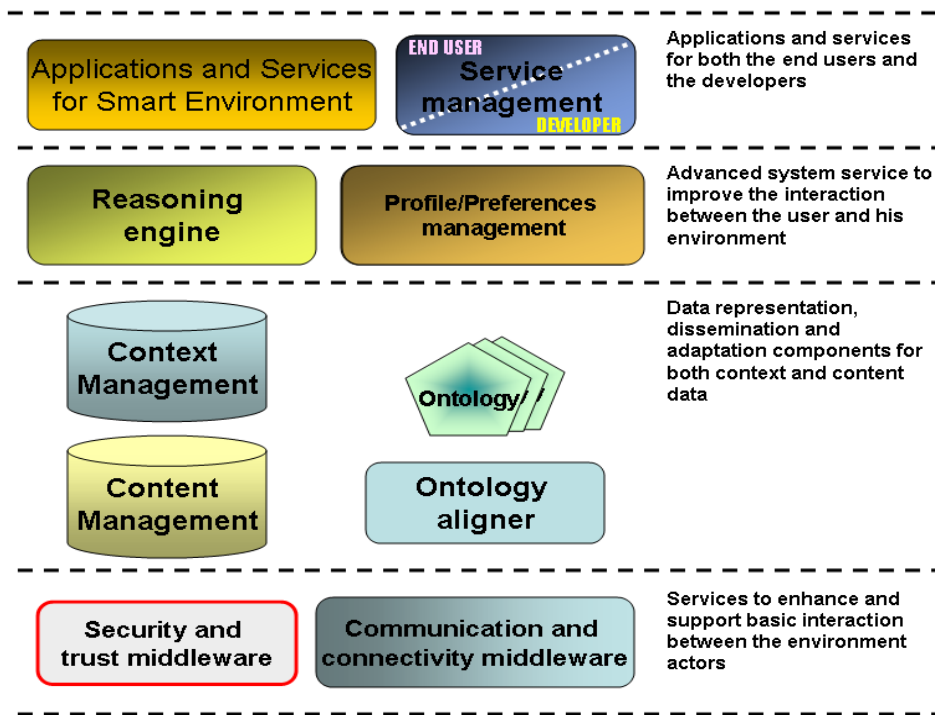
<sup>2</sup> Context Definition by Dey, Abowd & Salber (2001): "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*"

So far no context management system has been standardized yet, nor there are solutions that can be considered de-facto standards but there are many approaches and many architecture hypothesis, which depend on the inspiring principles.

Interoperability at different levels could be reachable by using an architecture that is able to support modules shown in Fig.1 that can be involved depending on the use case. So, not all of these components are always required.

With reference to Fig. 1, a *Communication and Connectivity Infrastructure* is needed to be able to interact with other actors on user or device level. An advanced middleware is necessary to overcome issues like:

- which communication channel is available?
- which is the best one?
- How can a Bluetooth device exchange information with another device with a WiFi interface?



**Figure 1: Interoperability enabling architecture components.**  
Applications or services for SE could be made by exploiting one or more of these components

In the other levels you may find:

A *Context Management System* is the infrastructure component to keep track of a digital representation of the environment and of the interaction between the user and the environment itself.

A *Content Management System* can be involved generally when content must be distribute – and/or store outside the user device – and in every case where the original content is not compatible with the addressed device hardware performance but must be adapted to. Because of that, a content management system should be a pro-active and dynamic module rather than a simple content store.

A *User Preferences Manager* can be useful to drive the services selection and to adapt service behaviour.

The final user could be interested to a new or particular service or to a new application. A *Service Management System* can help to manage service discovery and service dissemination. This can be done by regarding user preferences, profile and context. A portion of this theme should be dedicated to developers because some services can be design to improve and reduce the effort needed to make smart application and new services.

It is possible to reach the *Information Interoperability* by specifying an ontology that add semantics to the set of data. Since it is quite impossible to represent of the extra human knowledge into only one ontology it becomes more efficient to provide the system with *Ontology Aligners* that could work like translators or like a bridge between different ontologies.

A *Reasoning Engine* – by working on user preferences, context and profile information – could improve the quality of a service provider by finding the target point with less user interaction and reducing the working time.

*Security and trust* are a hot discussion theme in many application fields. A module to handle data encryption or to manage an access control list is often required to ensure protected and secure information exchange and service access.

Multi-vendor devices offer an increasing number of services and end-user applications that base their value on the ability to exploit the information originating from the surrounding environment by means of an increasing number of embedded sensors, e.g. GPS, compass, RFID reader, camera and so on.

<b>Interoperability level</b>	<b>Related to</b>
Communication and connectivity	Interoperability at the protocol and access control levels. Different applications/services must be able to communicate with each other, despite of heterogeneous programming interfaces and/or implementation languages
Devices	Capabilities sharing –e.g. temperature sensors or speakers – , smooth exchanging and replacing
Information	User preferences, context and location. Event data exchange capabilities. Ontology driven data management
Services	Service adaptation and reasoning based on available resources. Dynamic service discovery situation based
Content	Content adaptation based on device features

**Table 1: Smart Environment Interoperability levels**

However, usually such devices are not able to exchange information because of a lack of a shared data storage and common information exchange methods. A large number of standards and domain specific building blocks are available and are heavily used in today's products. However, the use of these solutions based on ready-to-use modules is not without problems. The integration and cooperation of different kinds of modules can be daunting because of growing complexity and dependency. In this scenarios it is necessary to have an infrastructure that makes the coexistence of multi-vendor devices easy, able to allow a low cost developing and a simple fruition of services for

both developers and final use. This sort of *technologies glue* should reduce both software and hardware integration issue by removing troubles of interoperability. The result should be also speed up and simplify the design, development, and deployment of cross-domain applications.

<b>The shared information principle</b>	Information ontology based model should drive the information interoperability and semantics. Information should include all data involved with the context.
<b>The simplicity principle</b>	The information level should not include knowledge about the use case and the architecture should be composed by few, general and simple components.
<b>The Service principle</b>	The SE behaviour should be close to the SOA one, that is, exporting functionalities like discovering and accessibility.
<b>The Agnostics principle</b>	Knowledge about ontology, application programming language, service, communication layer and hosting device/system should be not included into the architecture.
<b>The extensibility principle</b>	Functionality to manipulate information are not provided a-priori. Domain ontologies and information manipulation applications can extend the set of architecture functionality
<b>The notification principle</b>	Publish-subscribe functionality should be available for application to receive notification when context data changes.
<b>The security and trust principle</b>	Both the service level and information level should implement access control functionality if security and trust are required.
<b>The RAS (Reliability, Availability and Serviceability) principle</b>	At development time Reliability, Availability and Serviceability should be evaluated for both localization and application .They should be also measurable at execution time.

**Table 2. Interoperability principles**

Interoperability is intended as “*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*” [2]. According to this definition interoperability can be located on different abstraction levels as shown in Tab. 1.



To evaluate an interoperability enabled architecture some architecture principles should be outlined to define a priori capabilities and requirements. The following table summarizes some of the principles, taken from the set of principles defined in SOFIA project (Smart Object For Intelligent Application) [4] that will be discussed later.

# Chapter 3

## Related work

In the previous paragraph the view on interoperability platforms adopted in the project where I'm doing my activity was introduced. Interoperability platforms enable different kind of objects to interact. Objects can be e.g. sensors, devices, appliances, and embedded systems. These interacting objects form different kind of smart environments. The idea is to connect physical world with information world by enabling the sharing of information in digital format in physical spaces. This, sharing of information, enables novel and possibly cross-domain applications.

Here some related works are summarized.

### 3.1 Comparison of Interoperability Models

Interoperability models proposed or adopted in existing platforms vary depending on which interoperability levels are considered, on how interoperability is conceived and on the technical solution adopted. The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C4IF) exploits the concepts of language theories such as the language form, syntax, meaning and use of symbols and interpretations [5]. C4IF maps the linguistics concepts to the interoperability levels as follows:

- Connection interoperability is i) an ability to exchange signals, ii) a channel as an object of integration, and iii) out of content. Connection interoperability can be easily mapped to the device interoperability in our model.
- Communication interoperability is i) an ability to exchange data, ii) information as an object of integration, i.e. format and syntax of data, iii) out of context. Communication interoperability is mapped to the device interoperability in our definition.
- Consolidation interoperability is i) an ability to understand data, ii) information as an object of integration, and iii) out of usage. It is mapped to the service level interoperability in our model.
- Collaboration interoperability addresses i) an ability to act together, and ii) process as an object of integration. The definition is similar to the information interoperability in the sense that it concentrates on usage of information, i.e. actions/modelling. However, dynamism, cross-domain and cross-business interoperability are not considered. The main reason is the difference in application fields; smart spaces vs. information systems.

One of the definitions of interoperability levels widely used is the reference model of interoperability [6], where levels are defined as integrability, interoperability and composability. Integrability is achieved by technical and syntactical interoperability (related to network and connectivity). The interoperability level focuses on semantics and pragmatic interoperability that are related to simulation and implementation. Composability tries to manage dynamic and conceptual interoperability with abstraction and modelling. This is very similar to the ontology oriented modelling adopted in our approach and used for achieving cross-domain interoperability through abstraction and modelling.

In [7], a different approach has been taken for interoperability: Conflict Resolution Environment for Autonomous Mediation (CREAM). CREAM has similarities with our ontology oriented application development approach that gives support for modelling and abstracting concepts and their relations at the information interoperability level by means of a core ontology, domain ontologies and application specific ontologies. CREAM has three levels:

- Semantic conflict resolution ontology that provide a dynamic mechanism for comparing and manipulating contextual knowledge about each information source.
- Ontology relationship knowledge that forms the core of the reasoning process for semantic reconciliation.
- Semantic mediation service layer that manages schema mapping and ontology-schema mappings.

This kind of support is still missing from our approach although we are working for it. For example, a semantic information broker could acts as a conflict resolver, the core ontology as a mediator, and SmartModeller, a tool intended for smart space application development, could handle schema mapping and ontology-schema mappings. However, there is still a lot of work to be done in order to get the approach to work in practice.

Interoperability maturity models are compared in [8]. In summary, none of the compared models are based on system theories but one has strong basis on computer science. All of them have potential because of support for standards. However, less support was for flexibility to adapt, agility to react, openness and re-configurability. Inter-system interoperability was addressed in one model. Thus, we conclude that there are still much work to be done in order to put interoperability into practice in smart spaces that are made of heterogeneous systems, devices, and services.

### **3.2 Application domains**

The recently published papers on information interoperability highlight issues and topics related to the following fields:

- Health care systems
- Cross-border public services and
- Networked manufacturing systems.

Interoperability is seen as a fundamental requirement of a health care system to derive the societal benefits promised by the adoption of electronic medical records [9]. However, its achievement is difficult because interoperability benefits are highly

dispersed across many stakeholders, and early adopters are penalized by negative network externalities and first-mover disadvantages, e.g. faced barriers and challenges that have resulted in partial success, slow progress and outright failure. However, Extensible Markup Language (XML) is considered as an integration glue for biomedical information interoperability among disparate and dispersed systems – a common constellation in the fragmented world of healthcare [10]. The Open Health Tools [11] is an open source community with a vision of enabling an ecosystem, where members of Health and IT professions collaborate to build interoperable systems that enable patients and their care providers to have access to vital and reliable information at the time and place it is needed. Tools are really needed in order to tackle all interoperability challenges of heterogeneous environments, practices and cultures [12].

In [13], four types of Pan-European Public Services (PEPS) have been introduced and analyzed, and thereafter, a typology of semantic conflicts in PEPSs is defined; evidences, i.e. a piece of information that is used to activate the service, evidence placeholders, pre-conditions, service providers, public services, effects and service versions. In SOFIA, we focus on three types of Smart spaces, personal spaces, indoor spaces and smart cities. We use a set of ontologies to handle differences between spaces, domains, applications and businesses but so far we have not taken into account the differences between countries on the information interoperability level.

In [14], a product ontology based on standards about product data representation and exchange is proposed as a vehicle for achieving interoperability between networked production processes. In practice, PDM STEP Schema and IEC 62264 models were renormalized, conceptualized and represented by Unified Modeling Language (UML) class diagrams in order to have a common minimum denominator which allows the matching and mapping between the two standards. Although the context is far away of smart spaces, the solution is analogous to ours; minimal common ontology shared with two worlds of standards and rules for mapping the concepts of two standards. The work was made in a bottom-up fashion, similar to our approach.

### 3.3 Context

Designing context-aware applications requires an adequate context representation model. One of the most popular context definitions was provided by Dey and Abowd [16]. They describe context as "any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves" [17].

Context modelling may be categorized by considering the method adopted to represent and share context information. The following approaches should be mentioned:

- **Key-value pairs.** This is the simplest way to model context data used by Schilit et al. [22] to model context information like the location and export this as an environmental variable. Because its simplicity the key-value modeling approach is widely used in distributed service frameworks, such as discovery frameworks, e.g. Jini [23] or SLP [24], where a list of simple attribute-value pairs describe service functionalities – as well as system context information. The discovery procedure is realized by a matching mechanism on these attributes. In some cases Key-value pairs can not be adopted because a lack of structured data to enable context retrieval algorithms.
- **Markup scheme.** Markup tags with attribute and content are organized in a hierarchical data structure. ISO 8879 Information Processing Standard Generic Markup Language (SGML) [25] is a standard technology to define generalized markup languages. "eXtensible Markup Language" (XML) is the most commonly adopted language – or one of its vocabularies [26].
- **Graphical model.** UML is one of the widely used graphical oriented general purpose modeling instrument appropriate also to model the context. Henriksen et al. in [27] introduce a significant example of the graphic-

oriented context model which is a context extension of the Object-Role Modeling (ORM) approach [28]. Database-oriented applications are a prolific field for a graphical approach to model context for example to derive Entity-Relationship (ER) context models.

- **Object-oriented model.** Object-oriented approaches export encapsulation and reusability concept often useful to model, to represent and to access to multi level/multi actors environment context information. The *object* abstraction permit to encapsulate and hide low level details on the context data while providing contextual information by means of interfaces. It is an example the TEA project [29] with its concept of *cues*. It provides an abstraction for physical and logical sensors and provide a symbolic representation of a certain context data starting from the value of single physical or logical sensors data.
- **Logic-based models.** The means of facts, expressions and rules – in a logic-based context model – are used to represent and process the context. Generally a reasoning process is applied on conditional expressions and facts to obtain a derived set of new expressions and/or facts. Conditions usually are a set of rules. Contextual information is represented by means of logical expressions. All logic based models adopt an high degree of formality to represent the context and processing it. For example, the Sensed Context Model proposed by Gray and Salber [30] is based on a first-order predicate logic as a formal representation of contextual propositions and relations. Another approach within this category is the framework GAIA [31]. However others solutions can be adopted – e.g. fuzzy logic – to represent and reason about uncertain context information or to determine the quality of context information [32]. A logic context representation allows an automated reasoning improving the quality of the service.

- **Ontology-based models.** An ontology can be defined as "a formalization of a conceptualization" in according with the Gruber's definition [33]. Ontologies allow the description of context within specific knowledge domains. Semantic-based context models represent an emerging approach in context representation especially if it is coupled with an ontology – that permit to check the consistency of context information. Examples can be the CONON context modeling approach by Wang et al. [34] and the SOUPA ontology developed within the CoBrA system [35].

Many other approaches to context modelling are available in the literature. For a more extended analysis of this topic refer to [18].

### 3.4 Context-Aware architectures

According to [19] a smart environment is a dynamic system that can change its behaviours based on context-awareness mechanisms implemented as core elements of its architecture. A context-aware architecture collects, uses and interprets context information and changes its functionality to the current context of use.

In the survey on context-aware web service systems presented in [19], the following questions were considered:

1. *Context information and context representation:* which techniques should be adopted to modeling context information?
2. *Context sensor techniques:* how to measure and sense context information?
3. *Context storage techniques:* how context information is stored and how the information can be accessed from its storage?
4. *Context distribution techniques:* how application and services can access and retrieve context information? How to disseminate context information to different components?
5. *Security and privacy techniques:* how to protect context information and the

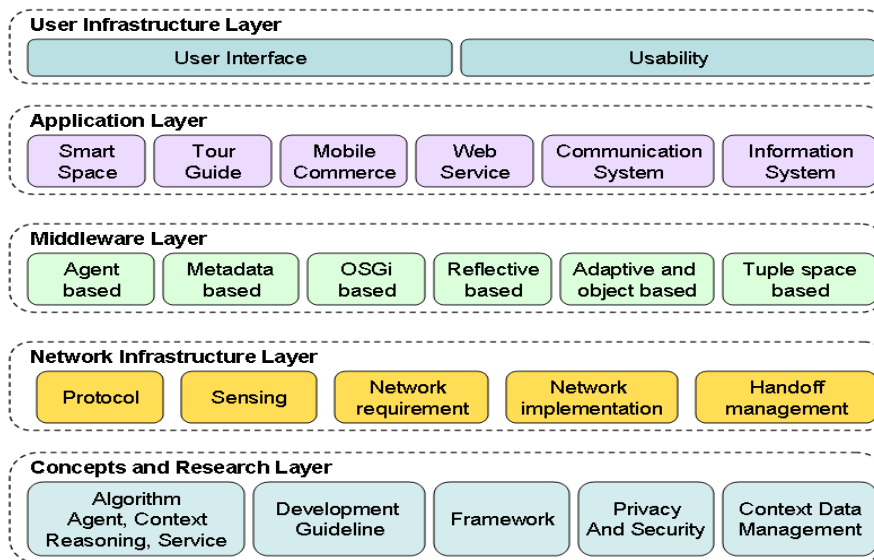


access to it? Which authentication and/or authorization mechanism should be adopted? How to reach privacy on connection channel to the context shared store?

6. *Context adaptation techniques*: In which context information are actually used?

According to [16] context-aware systems have typical layered architecture like distributed systems. According to [21] the classification framework of context-aware systems can be decomposed in five layers (Fig. 2):

- end user infrastructure
- application and service
- middleware
- network infrastructure
- concepts and research



**Figure 2: The classification framework of context-aware systems [21]**

My contribution was mainly is in the research layer and particularly on:

- context reasoning (see section 5.3)
- the use of agent and service technologies for context-awareness (see section 5.4.1)
- context-awareness design and evaluation (see sections 5.2.4, 5.2.5)
- security and privacy (see section 5.4.3)
- context data management (see sections 5.2.1, 5.4.1)

# Chapter 4

## Research frameworks

My research program was carried out within the framework of three main research initiatives: a European Network of excellence on cultural heritage EPOCH (2004-2008), a joint research project with Telecom Italia Lab on social advertising and a European project on interoperable smart environments named SOFIA.

Common thread within these projects was the goal to study system architectures that can handle context information and services and support the interaction between user and context-aware applications.

Acting in different scenarios, ranging from cultural heritage to social applications to smart buildings, my research concerned:

- Profiling

- Tools able to dynamically model the behaviour of the system by monitoring and know the user context and actions

- Recommendation

- Special purpose engine able to automatically recommend services and content by combining context, profile, domain and group information

-Content adaptation

Special purpose engine able to adapt user interface and content based on the user preferences, profile and context

-Domain ontology definition

Formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.

With reference to the list of my publication – section 7 - the following topics were mainly covered:

- User preference service adaptation [pub. 2]
- Context management [pub.8]
- Content management [pub.3]
- Information interoperability [pub.1]
- Multivendor device interoperability [pub.1],[ pub.8]
- Communication/connectivity interoperability [pub.6],[ pub.1]
- Service discovery [pub.6],[ pub.3]

## **4.1 European Network of Excellence on Cultural Heritage (EPOCH)**

*EPOCH [36] is a network of about a hundred European cultural institutions joining their efforts to improve the quality and effectiveness of the use of Information and Communication Technology for Cultural Heritage. Participants include university departments, research centres, heritage institutions, such as museums or national heritage agencies, and commercial enterprises, together endeavouring to overcome the fragmentation of current research in this field. [37].*

The goal of the network is to define a framework to overcome fragmentation on cultural heritage applications. EPOCH is focused on all processes and information involved in cultural heritage domain, from archaeological discovery to education and dissemination.

Within EPOCH I worked on a project called CIMAD "[Common Infrastructure|Context Influenced] Mobile Acquisition and Delivery of cultural heritage data" [38][39]. CIMAD provided a framework supporting the development of cultural heritage "services". CIMAD results were shown in many European museums as an itinerant exhibition called *Interactive Salon* [12].

## **4.2 Joint research Telecom Italia Lab**

Within a collaboration with TiLab-TELECOM Italia [40], I was involved in a research project on social advertising. The goal was to develop and implement an algorithm and a web based service to recommend activities to the end-users based on their preferences and context. The proposed solution was based on the Telecom CAP – Context Aware Platform –,an implementation of Telecom context management architecture [41],[42]. Research goal was to improve the CAP by introducing a mechanism to discover and recommend new social activities. The project was called CaPUA (Context-aware Preferences, Users, and Activities) and it was carried out under the supervision of professor Paolo Ciaccia, at the University of Bologna.

## **4.3 European technology platform on embedded system (SOFIA)**

Sofia (2009-2011) is an European research project carried out by 19 academic and industrial partners led by Nokia. Partners joined their effort to enable people to benefit from smart environments, by “*making information in the physical world available to users while maintaining existing legacy*” [44].

SOFIA is funded through the European ARTEMIS programme [45] under the subprogramme SP3 “*Smart environments and scalable digital services*”, within the 7th European Framework Programme, and it aims to provide a shared information search extent for cross-domain end-to-end applications executed by multivendor

devices. The primary SOFIA project contribution is an Inter-Operability Platform (IOP). Sofia IOP is an infrastructure to assist developers with added-value interoperable information about objects/sensors/devices spread within the surrounding environment. The information repository is active and it can trigger external entities to react to relevant and selected environmental changes. The IOP was designed starting from sixteen principles [4] that originate from vertical application domains, i.e. personal spaces, smart housing and smart city.

# Chapter 5

## Architectures and platforms

This section will review the context-aware platforms I was involved with along my PhD program, starting from a dedicated solution, moving to the Interoperability Platform developed within SOFIA Project (see par. 4.3).

These solutions are intended to support a wide range of applications, addressing domain-specific as well as multi-domain scenarios. Methods, architecture components and SW agents developed to support context-aware services will be described.

The following platform and frameworks will be considered:

- *WHIRE*: Application specific, context-aware mobile client with server based multimedia content distribution [49][50]
- *MobiComp*: a context management framework for Cultural Heritage applications [38]
- *CAB*: a context, preference and profile based application broker to support service and application distribution [12]
- *Smart-M3*: "Semantic Web based" information sharing infrastructure for smart spaces designed by Nokia within the European project SOFIA
- *NoTa*: a service and transport independent connectivity framework designed by Nokia [47]
- *OSGi*: the well known Java based service support framework

## 5.1 Multimedia guide for museums and archaeological sites

*WHYRE® is a hands-free, sensory augmented, wearable computer designed to turn museums and archaeological sites into communicating machines. It offers a unified interface to multiple format contents, including interactive 3D, sensors driven QTVRs, and streamed animations. It is based on an IA32 mobile platform with a 3D graphics accelerator. Its operating system is Windows XP Embedded [12]. WHYRE was developed by Ducati Energia [48] in cooperation with the University of Bologna and other partners within an Italian research project – Parnaso (2000-2003).*

WHYRE[49],[50] was used as a dedicated context aware platform where the context (basically the user current place and field-of-view in a Museum or archeological site) was evaluated based on on-board sensors, i.e. a compass, a gyroscope and two accelerometers [51]. The WIFI interface was also considered a sensor as it was used not only for wireless communication but also for coarse location detection [76].

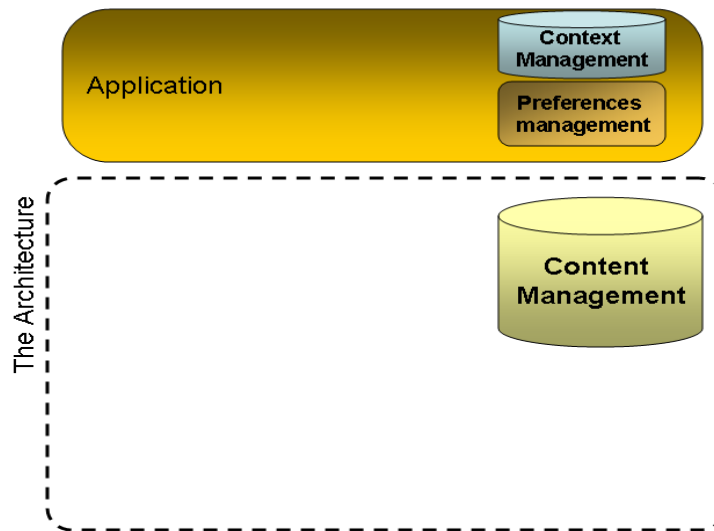


**Figure 3: Two WHYRE views (left and center). On the right a museum use case where a visitor wearing the WHYRE can enjoy multimedia contents while sensors localize him in terms of position and orientation**

WHYRE tracks the user position and direction when he is visiting a museum or an archaeological site. A specific application running on WHYRE exploits the user's context to dynamically select the relevant multimedia content associated to each exhibit to be shown to the visitor.

The architecture is sketched in Fig.4. Multimedia contents are centralized, shared and distributed through the network. This is a simple architecture for a specific use case. Context – e.g. sensors data – is managed locally at application level; it is not shared

and it does not have any impact on the environment. This reduces the interoperability level and it prevents to develop services or applications that exploit the user context.



**Figure 4: Minimalistic architecture for an application specific system**

Only adopting a context management system - as show in the next section – it becomes possible to share context data opening the way to additional services like visitor monitoring and object localization services inside the museum – i.e. to define for every object or exhibit of interest some context information like their position and/or location to be used in multiple services. By knowing exhibits and user exhibits position, it is possible to collect some important information about the visit – e.g. amount of time spent on each exhibit, which exhibits were visited, the path followed by the user – and then build new interesting services upon this information, such as, for example, pedestrian navigation support.

Next section will show the way to “extend” the range of services through a shared context management system. This will demonstrate the requirement for and the implementation of the *extensibility principle*, i.e. one of the principles of a target interoperability platform (see section 2).

To summarize this section, WHYRE relies on a centralized dedicated architecture which is unable to establish interoperability at information and device level. The evolution towards a smart environment scenario should start from context sharing.



Next section will address a context management platform supporting context sharing among a set of context-aware services and applications. Issues like multi-vendor device support, device usability, service distribution, information sharing and content interoperability will be faced.

## **5.2 A Context-aware platform for Cultural Heritage applications and services**

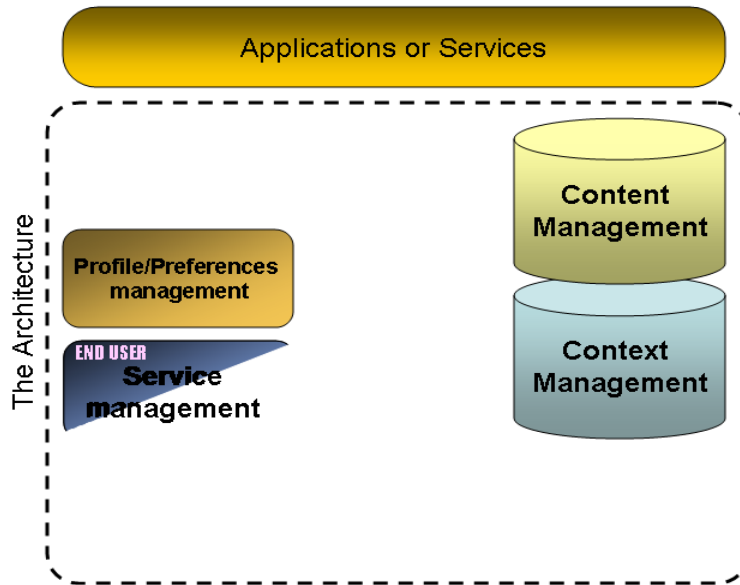
This section is dedicated to context-aware solutions in the Cultural Heritage domain. Here not only the visitor guide is considered, but also several additional services that we want to provide with the same platform.

The goal is to provide an infrastructure supporting access to customized services and to focus on the following properties [12]:

- Scalability
- Interoperability of Context Information
- Context-awareness
- Developers facilities
- Support to heterogeneous devices
- Content adaptation

This activity was carried out within the framework of the EPOCH Network of Excellence.

The focus was to discover pros and cons of architecture components implementations and extend them to meet new requirements.



**Figure 5: Schematic view of the architecture adopted in EPOCH. It provides context, preferences, profile and content management. Part of the architecture was dedicated to service management for the end user**

Services and applications for both end-users and museum curators were also considered. Fig. 5 shows a schematic view of the architecture. Here a more complex architecture was adopted compared with the one shown in paragraph 5.1. Very important was to introduce a shared context management system. This was the first step towards a interoperability SE. Next paragraphs are dedicated to the work done on this architecture.

### 5.2.1 MobiComp: a context management solution for Cultural Heritage

Mobicomp is a context management framework to store, retrieve and aggregate *context elements*, i.e. heterogeneous static and dynamic information about the entities involved (e.g. people, exhibits, devices, sensors). A *context-element* includes: a subject-predicate-object triple – relating an entity identifier to a named context value – a time-stamp, and additional specifiers (e.g. validity period and privacy level). MobiComp also provide a data store features to maintain history of entities data.

MobiComp is written in Java. Hence, multiplatform applications can be easily written. To develop a MobiComp based custom application, MobiComp client source code needs to be compiled with the appropriate client dependent JDK. Standard PCs, PDAs and mobile phones are supported. The compatibility is guarantee from the JVM 1.4 to the 1.6 at the moment of writing.

Three components exist allowing the interaction with MobiComp: trackers, listeners and aggregators. The Fig. 6 report a top view of the MobiComp components:

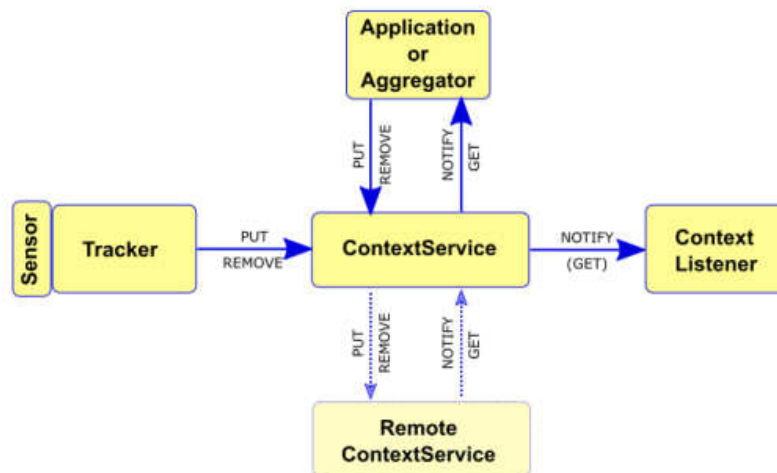


Figure 6: MobiComp architecture

- A tracker is a MobiComp component that acts as a context producer. Trackers register their availability and capabilities by sending appropriate information to the ContextService. Their purpose is to collect raw context data from sensors, such as GPS receivers, and other dynamic or static sources, including configuration files for

device capabilities and user-preferences. Trackers transform their input into context elements which are then put into the MobiComp repository. It sends to MobiComp a subject-predicate-object triple, e.g. the tracker ID and the pair (“predicate-ID”, “value”). Trackers can provide multi level data like sensor data (e.g. the user position inside a room), higher level information (e.g. the place ID) or other context information made by a reasoning engine.

- The second component that interacts with MobiComp is a listener. It is a Java object waiting for events. Listeners receive notifications of ContextEvents from the ContextService and perform some actions based on the context element carried by the event object. They receive event notifications whenever a context element is put into the store or modified by a tracker that sends a Context Element to MobiComp (*put* action). On receiving a notification, the listener may get the element from the store and use it as required.

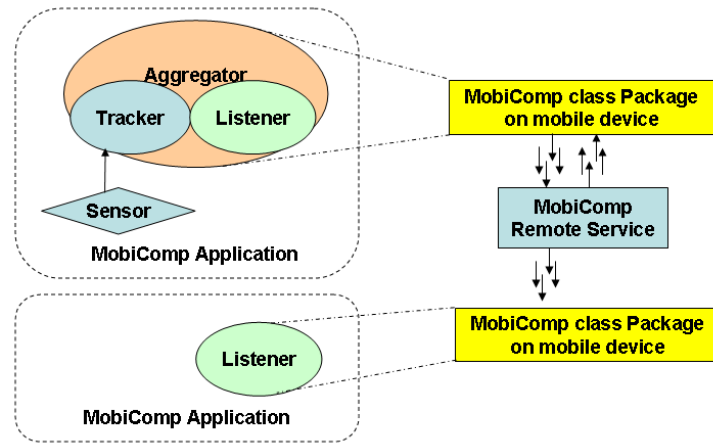
- The third MobiComp component is the “Aggregator”. The Aggregator combines the behaviour of both a tracker and a listener. Aggregators monitor events from the ContextService, rather than a sensor device, and apply a transformation before returning a new element to the repository. For example, aggregators can combine several low-level sensor elements to produce an element at a higher level of abstraction. Also, aggregators may perform transformation services, i.e. converting latitude and longitude coordinates from a GPS sensor to coordinates on an appropriate local or national grid. Many non-trivial context-aware applications utilise a number of complex context aggregators, e.g. the FieldMap [52] application.

MobiComp has a local context data store – *ContextService*, see Fig. 6 – automatically aligned with the remote one. It can prevent loss of data in case of remote MobiComp service not reachable and can speed up both data store and retrieve. XPath query are also supported [65].

The widely reused model of a MobiComp application is shown in Fig. 7.

Usually each use case involves at least two services that use MobiComp components. The first service is a context producer, based on a tracker that collects some

interesting context data and sends them to MobiComp. The second one is a context consumer, based on a listener that reacts to context change events reading from MobiComp repository information originated by the producer. Producers and consumers may run on different machines. An aggregator can be used rather than a listener or a tracker whenever deemed useful.

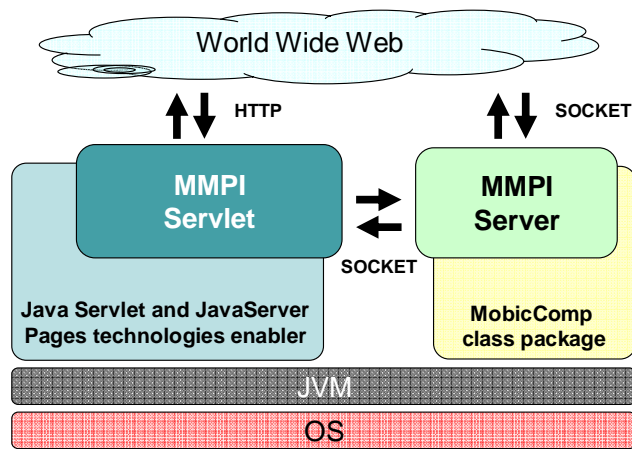


**Figure 7: abstract view of a MobiComp application: an aggregator produces new context data by processing already available information; a listener gets context information from the context store, e.g. for monitoring purposes**

MobiComp was one of the first context management frameworks ever developed by the academic world and had the following main limitations experienced in some use case. There was not an event notify mechanism therefore to implement an event driven interaction model it was necessary to use polling loop method to discover data changes. This reduced performance and it took developers time. A first solution should be to integrate this mechanism inside the MobiComp middleware and then to change the interaction model from *Web Service* to *Publish Subscribe* model. Furthermore MobiComp should be enhanced by adopting an ontology to give semantics to the data store and make possible the information interoperability between different vertical use cases and different domains. MobiComp was used to track people position inside a museum site. At the moment of writing MobiComp was not intended to support *real-time* use . Therefore MobiComp it can not be used in scenarios where information changes quickly or when many clients call for services at the same time. This was demonstrated by many experiments done in cultural heritage scenarios aiming to evaluate the scalability of the framework.

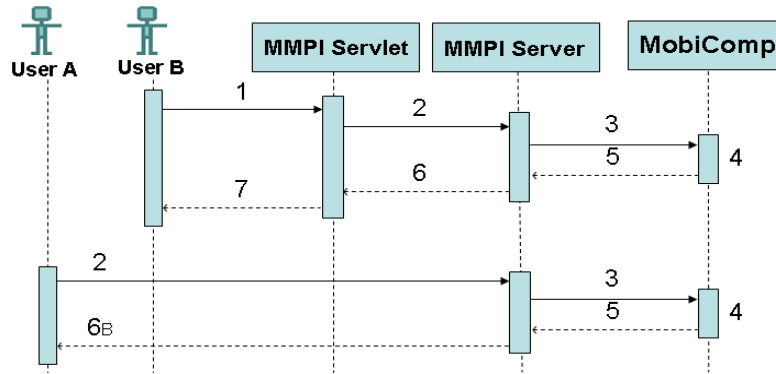
## 5.2.2 Device and language interoperability for MobiComp

Another issue with MobiComp was its Java dependency: programming languages interoperability was not supported. My contribution here was to design and to develop a service called MMPI (MobiComp Multi Platform Interface). MMPI service is an alternative – servlet/socket based – interface to MobiComp that removes both the requirement to have a JVM installed on the client side and the need to use the MobiComp java classes. With MMPI, MobiComp can be accessed from any platform because MMPI removes constrains in term of programming languages and OS.



**Figure 8: MMPI architecture**

MMPI consists of two components (Fig. 8): a java servlet and a stand alone server. They can work in conjunction or separately. The former works as a web interface resolving HTTP client request and enabling web application developers to access to MobiComp through a browser. The latter is a multi-thread Mobicomp aggregator Java program that wait for client connection through socket .



User can have different type of device  
 User can use both the web service interface (servlet) or the socket one (server)

1. A request for context data is sent through the LAN to the MMPI Servlet
2. The servlet receive the request and move it to the MMPI server
3. The server move the user request to MobiComp
4. MopiComp elaborate the request and then reply to the request
5. Send back the answer to the server that...
6. Send back the answer to the servlet (B:to the user) that...
7. Send back the query response to the client

**Figure 9: MMPI message tracking UML sequence diagram**

Both the servlet and the server can handle request like:

- publish - *put* - data on MobiComp:

The *put* command allow clients to publish new subject-predicate-object data triple on MobioComp. It is required that the client send him MobiComp ID (MID) and a list of papameters: the predicate and the object value.

- retrieve - *get* - data from MobiComp:

The *get* command is used to search the context repository for specific context elements. A context elements include triple as subject-predicate-object. With this command is possible to specify the triple. One element can be a wildcard with the meaning of *any element* - the wildcard adopted was the “\*”. If the subject is the “\*” wildcard, a list of all subjects with relative values is retrieved. It is required that the client send his MID and a list of parameters: the subject MID, the predicate and the object value.

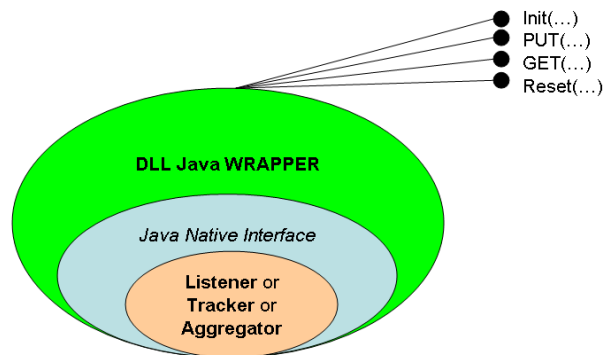
- register - *reg* - a new entity in MobiComp:

The *reg* command is used to register a new entity on Mobicomp. It is required that the client send him MID and a list of parameters: the entity name, the entity type and an entity description.

Fig. 9 shows how a MMPI request can evolve during the time. The user A sends his request to MobiComp through a socket connection to the server and receives the answer directly from the MMPI server. The user B uses the web interface so his request goes first through the servlet (steps 6 and 7 in Fig.9).

### 5.2.3 Multi programming languages: MobiComp Enabler Library

The MMPI could not be a scalable solution in some case, so devices involved need to be MobiComp enabled by adopting the MobiComp's standard library. However to reduce programming languages constrains a set of C/C++ DLL libraries was developed to wrap the MobiComp Java library supporting different programming languages interoperability - MobiComp Enabler Library (MEL). The Java Native Interface [53] was used to encapsulate Java classes. It is quite simple to use these C/C++ DLL from many different programming languages – e.g. Visual Basic, Java, C/C++, C# and so on. These modules allow the use of trackers, aggregators and listeners from programming languages other than Java enlarging developers expression possibilities. These library wrap not only the basic functionality MobiComp class but also some specific use case SW modules. The Fig. 10 show the generic wrapper layered model.



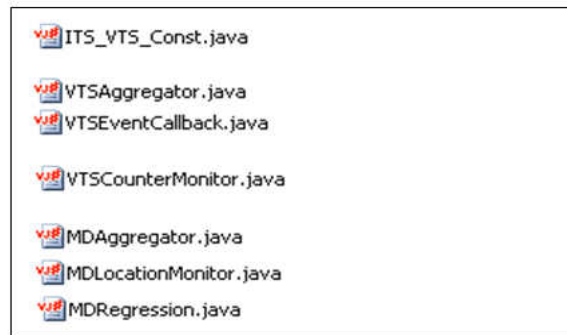
**Figure 10: MobiComp wrapper model. Non-java applications can use MobiComp with this method**



Not all methods and functions of a generic MobiComp module are provided by the wrapper to the top level user. The wrapper provides only high level methods useful to communicate with MobiComp and some methods to handle the state of the DLL - e.g. for the latter INIT and RESET methods and for the former the PUT and GET methods and an event notifier method used in case of context data changing. To support the experimental CIMAD services a wrapper for a visitor guide and people tracking services were developed.

The following rules were adopted for class naming:

- MD stands for “Mobile Device” (WHYRE® in our case)
- VTS stands for “Video Tracking System” (VTS) (described below in par. 5.2.6.2) [56]
- ITS stands for “inertial tracking system”, i.e. the navigation system available on WHYRE®



**Figure 11: List of MobiComp actors classes**

With reference to Fig. 11, which shows my MobiComp classes:

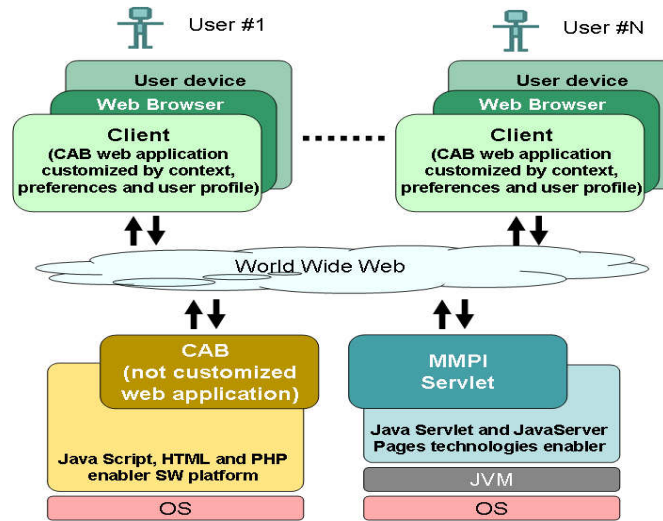
- The VTSAggregator class supports mechanism to publish and to retrieve context information provided by the VTS itself to maintains the people counter service working.
- The VTSCounterMonitor class is a listener reacting to events generated by the VTS and it provides functions supporting Monitoring Services.
- The MDLocationMonitor class is a listener. It is activated by context change events originated by the mobile devices – e.g. new position or new Point Of Interest (POI). It is the base component for user on-site tracking service.

Other classes provide auxiliary functions. For example VTS EventCallback supports the functionality to receive and handle context data changes event. MDRegression

supports reference coordinate system changes (e.g. from stereo camera vision system coordinates to museum coordinates). ITS\_VTS\_Const is a class containing useful naming constants.

### 5.2.4 Context-aware application broker




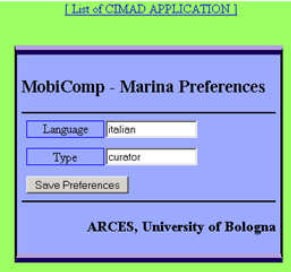
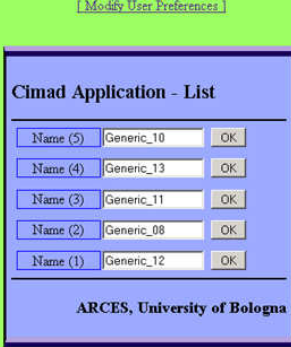
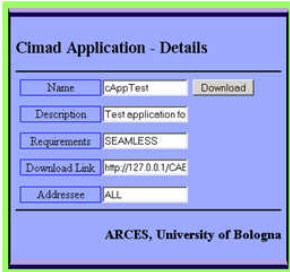
A first prototype of a service/application management service was developed within a project named CIMAD (section 4.1) and was therefore called “CIMAD Application Broker” (CAB). CAB is a web service, it is based on MMPI service and it was developed in PHP and Java-Script. CAB enables the discovery of context-aware applications reachable on the web (*CIMAD applications*). It enables the user to choose a CIMAD application from a list. The list is made by considering user profile, preferences, context and device features.



**Figure 12: CAB architecture enhanced by the context access service provided by MMPI. The CAB service is customized on the client side by using user preferences, profile and context. It is based on the HTTP communication protocol**

Fig. 12 shows the system architecture and the interaction between modules. CAB requires that both the users and their own device are provided with a MID (MobiComp Identifier). The hypothesis is that a device is owned by just one user. A CAB service understands automatically if the device is registered or not and if it has

an owner. This is done by using cookies to save some information on the device – e.g. device and user MIDs. A CAB’s feature allows to reset this information at any time.

<p style="text-align: center;"><b>1</b></p>  <p>The user device is not recognized as a valid MobiComp device. It means that the device doesn't have a MID</p>	<p style="text-align: center;"><b>2</b></p>  <p>The user device is not recognized as a valid MobiComp device. It means that the device doesn't have a MID</p>	<p style="text-align: center;"><b>3</b></p>  <p>CAB recognizes the user and the device. It is possible to change the own preferences or to search for a CIMAD Application</p>
<p style="text-align: center;"><b>4</b></p>  <p>Preferences example. The user language and the user role</p>	<p style="text-align: center;"><b>5</b></p>  <p>List of available CIMAD applications based on user preferences, profile and context</p>	<p style="text-align: center;"><b>6</b></p>  <p>Details on the selected CIMAD Application . It can be executed or downloaded.</p>

**Table 3: Steps to find and select a CIMAD application**

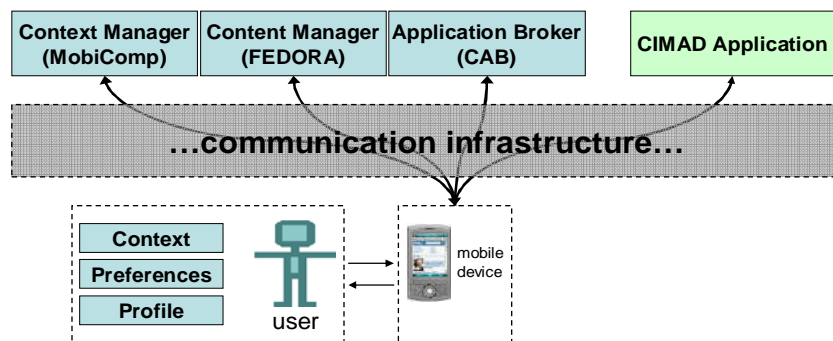
This is useful when a device may be time-shared by multiple users who became temporary owners; this is the case, for example, of museum guides for hire. Context can include also information about the device e.g. the screen resolution and list of sensors available on-board. To reach a CIMAD Application, the CAB service requires

that the user follows some steps. These are explained in Tab. 3. Future work may include the definition of services and users through an ontology. This could made interoperable the CAB service and the way to publish and associate a new service to a set of users. At the moment the CAB has only a web interface for user interaction purposes, but the service could be made available through a socket or HTTP interface in the future.

Next paragraph is dedicated to the content management system adopted within EPOCH and to its integration in the platform with the purpose to achieve content and device interoperability.

### 5.2.5 FEDORA content management system

FEDORA (Flexible Extensible Digital Object Repository Architecture)[54] was adopted for its scalability and ability to adapt multimedia contents to heterogeneous devices. *Fedora was originally developed by researchers at Cornell University as an architecture for storing, managing, and accessing digital content in the form of digital objects inspired by the Kahn and Wilensky Framework. Fedora defines a set of abstractions for expressing digital objects, asserting relationships among digital objects, and linking "behaviours" (i.e., services) to digital objects [55].*



**Figure 13: Schematic view of the final architecture developed under the EPOCH project**

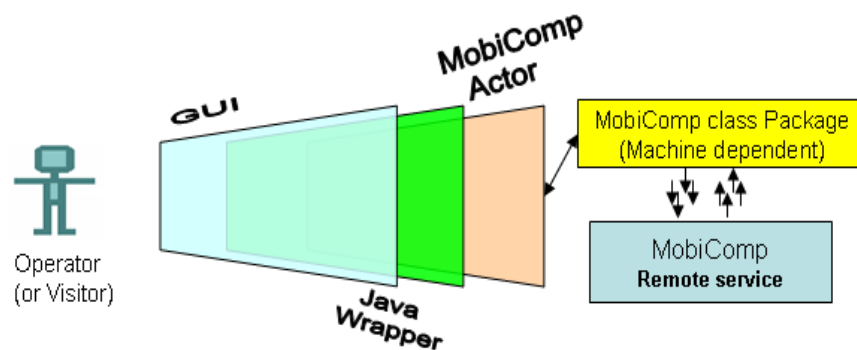
To reach device interoperability it is necessary to reduce technologies constrains. In CH scenario, multimedia content needs to be adapted to device performance and technologies features – e.g. screen resolution. A solution can be to adopt a content management system able to adapt content on-demand with the respect to device features.

My target was to study how to integrate FEDORA with the other architecture components. Fig. 13 shows a schematic view of the final architecture whereby it is possible to handle context information, adapt content based on device performance and find the appropriate context-aware application by using user preferences, profile and context.

The integration of FEDORA was simple because its service can be reached through the HTTP protocol, reducing communication and developing issues.

### 5.2.6 Tool chain for Cultural Heritage sites

A set of applications and services was developed to enrich a cultural heritage environment. Some tools were integrated to support user-system interaction, the user being the visitors, the museum curator and also the developers.

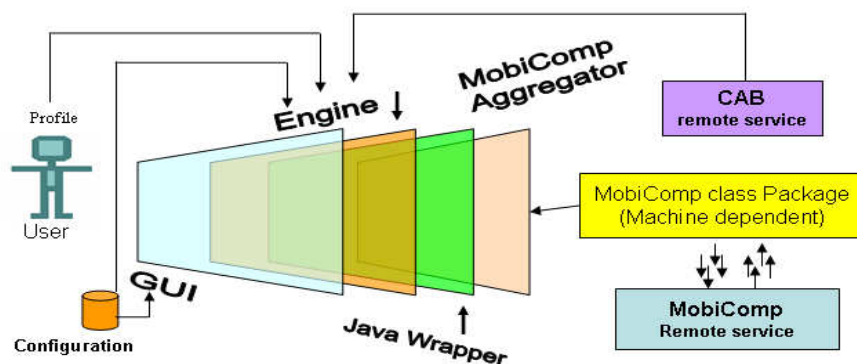


**Figure 14: Structure of a generic MobiComp application. Actors means MobiComp Java classes that implement a listener, a tracker or an aggregator**

Fig. 14 shows a layered structure of a generic application that exploits context information provided by MobiComp. It can be a very simple application. Thanks to architecture and the related MobiComp library its implementation required a minimal effort i.e. the development of a Graphic User Interface (GUI), because most of the work was done previously and it could be reused. Examples of this type of applications were Visitor Monitoring and Tracking, Museum Guide and a Site Data Collection service detailed below (see sections 5.2.6.1 to 5.2.6.5).

### 5.2.6.1 Monitoring, Tracking guiding and data collection service

With reference to fig. 15 this paragraph describes the structure of a Multimedia Museum Guide and a Site Data Collection service. The former is addressed to museum visitors, the latter is addressed to museum operators who need to collect information about exhibits spread into the museum. Information may be additional detail including multimedia content for the exhibit, and context information like the exhibit “place” inside the museum and/or its position. Looking from the user perspective, we first find a GUI that can change according to the user profile and preferences. Configuration info – e.g. connection information or specific application setting – can be stored locally on the device. Next to the GUI there is the “engine” level acting as coordinator between the user, MobiComp and, the CAB service.



**Figure 15: Layered structure of a “visitor guide” CIMAD Application.**  
**This is a generic example on how all components can be put together to build a CIMAD application following the MVC pattern**

The GUI and the Engine are two separate components of the same client-side application. In the prototype the engine and the GUI were developed in C# and C++ respectively.

As MobiComp is java-based, a java wrapper allows the engine to communicate with MobiComp directly. Information exchange and event notifications are managed by a MobiComp aggregator.

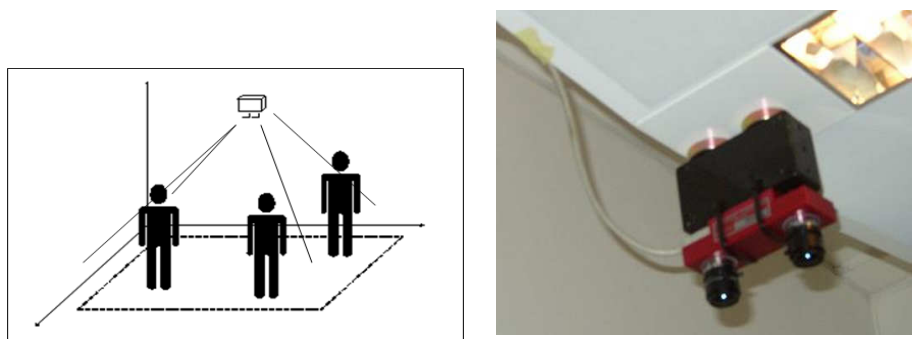
The behaviour of the application is customizable. For example, by changing some configuration data, such as the user profile and/or preferences, the scenario changes and the guide behaviour may be turned into a data collection behaviour.

### 5.2.6.2 Access detection service

Some of the MobiComp based services developed within EPOCH rely on a computer vision based tracking system named VTS [56] developed by third parties ([56] ).

The VTS accurately tracks people walking under a stereo camera and counts the people crossing a virtual line located in its Field-of-View.

From these counting events, higher level dynamic information about the museum (e.g. Day visitors, Average Occupancy, Average Visit Time and others) can be estimated and exploited by a CIMAD application.



**Figure 16: On the left the stereo camera field-of-view. On the right the stereo camera installed in a museum access gate (STH-MDCS2-C Videre Design)**

Fig. 16 shows the camera installation set up. In Fig. 17 some VTS program screenshots are reported, in order to show the camera view and the counter of the virtual line crossings.



**Figure 17: VTS program screenshots. People under the stereo camera are tagged with a colored cube. The VTS counts the people crossing the green virtual line**

The application was modified to become a MobiComp data producer; in this way the stereo camera was turned into a smart sensor.



### 5.2.6.3 Presence monitoring service

A service to remotely monitor the presence of people inside the museum was implemented starting from the VTS coupled to the CAB service, the MMPI service and MobiComp. Fig. 18 shows a schematic view of the use case where appropriate users can enjoy the monitoring application on their mobile device. A user – with curator profile – can join the CAB, discover the monitoring application and then run it on his personal mobile device.

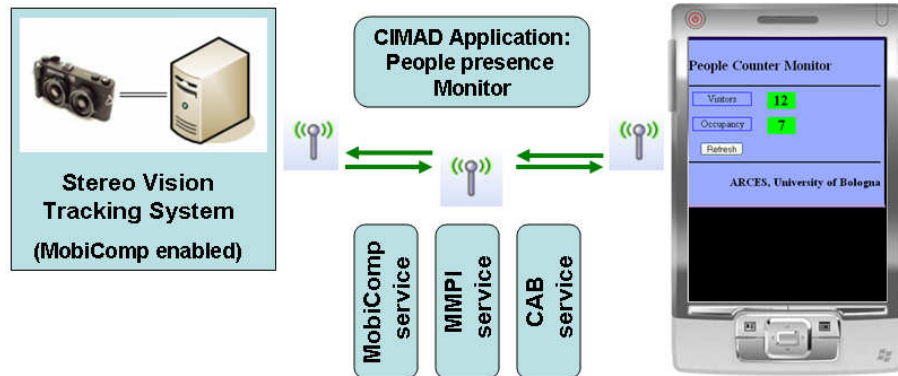


Figure 18: Integration of the stereo vision based tracking system

The monitoring application is a web application developed with PHP and Java-Script technologies. It is a data consumer. Thanks to the MMPI service it is possible to reach the VTS context data stored in MobiComp.

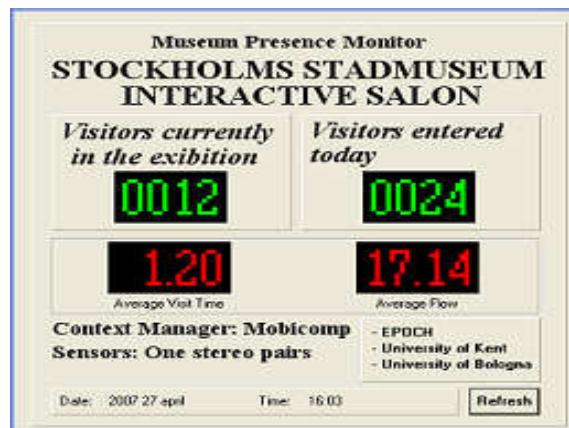


Figure 19: Real time visitors flow indicator

Fig. 19 shows the output of the presence monitoring application developed in C++ on Windows XP. The application is based on a “Mobicomp Enabler Library” (MEL) and it demonstrates that heterogeneous applications developed in different programming languages can coexist and use the same data with possible different purposes. Development time and effort are reduced by using the services and the ready-to-use SW packages provided by the infrastructure. This application demonstrates is a tiny step in the direction of interoperability at platform level in context-aware computing.

The VTS was installed on the access gate of several museum exhibitions – please refer to *The Interactive Salon* [40] - In order to make the best use of the information produced by the VTS a Smart Application was developed to gather statistics about the visitor flow over long periods of time. Tab. 4 shows screenshots of charts produced by the service.

### 5.2.6.4 Statistics on visitors flow

The VTS was installed on the access gate of several museum exhibitions – please refer to *The Interactive Salon* [40] - In order to make the best use of the information produced by the VTS a Smart Application was developed to gather statistics about the visitor flow over long periods of time. Tab. 4 shows screenshots of charts produced by the service.

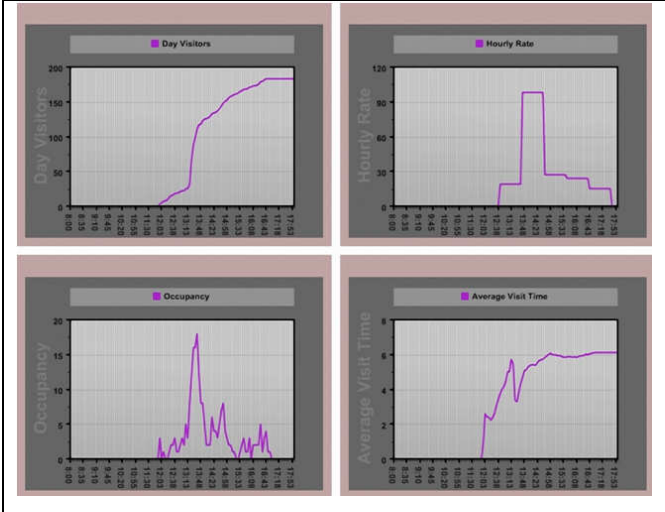
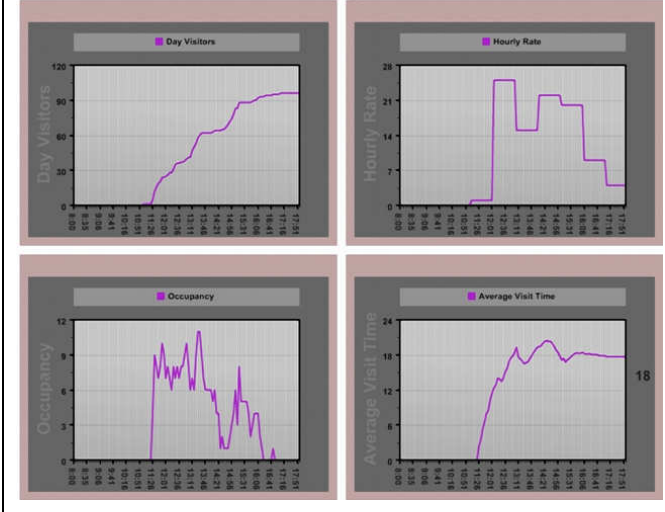

	<p>Two set of four charts showing presence information during the day in two visit days. Charts show: visitors total count during the day, visitors hourly rate, occupancy (i.e. the average amount of people inside the exhibition in every hour) and average visit time (i.e. the average amount of time spent inside the exhibition in every hour).</p>
	
	<p>Monthly reports show amount of visitors and average visit time day by day over a month</p>

Table 4. Screenshot of statistics produced by a web-based visitor flow monitoring service

### 5.2.6.5 People tracking service

Here a MobiComp application called “People Tracking Application” (PTA) is summarized. The PTA is based on positioning information gathered by WHYRE’s on board sensors (section 5.1) and it is a data consumer providing a mobile device tracking service to museum curators (screenshot in Fig. 20). It is developed in C++ on Windows XP and it uses the MEL. It is a general purpose smart application and can be adopted when position information is provided to the Context Management Platform. All kind of device that share the same context information in the same way – e.g. the user position – can be tracked by this application. Under semantics constrains all kind of information can be shared and used smoothly in cross domain applications.

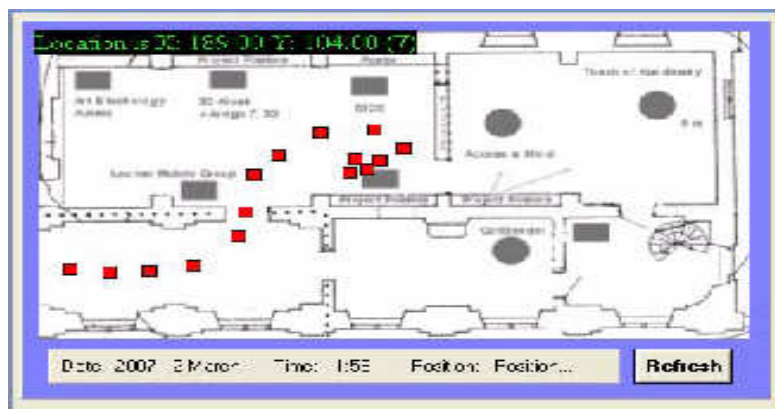


Figure 20: User tracking inside a museum site

Within EPOCH information interoperability was assured by a standard XML schema. However, while XML schemas is most useful to describe the structure and to validate documents [57] an ontology should be used to ensure information interoperability. With the help of an ontology and its related aligner each context-aware application can produce compatible and machine understandable context information, as it will be shown in section 5.4.

Additionally, as proof of concept, a context aware service that exploit this shared information is shown in next paragraph.

### 5.2.6.6 Pedestrian navigation support

An important principle for context-aware platforms is the extensibility principle introduced in section 2. As proof of concept a new service was introduced to add more features to the platform. This was a pedestrian navigation support service. It was developed both as a stand-alone C++ DLL library for Windows OS and as a web service to achieve device interoperability and also for reliability and serviceability purposes (see RAS principle in section 2). The service simply finds the shortest path between a source point and a target point. In cultural heritage applications the source point often coincides with the user position and the target point with an exhibit to reach. The service can also be used whenever a path solver is required.

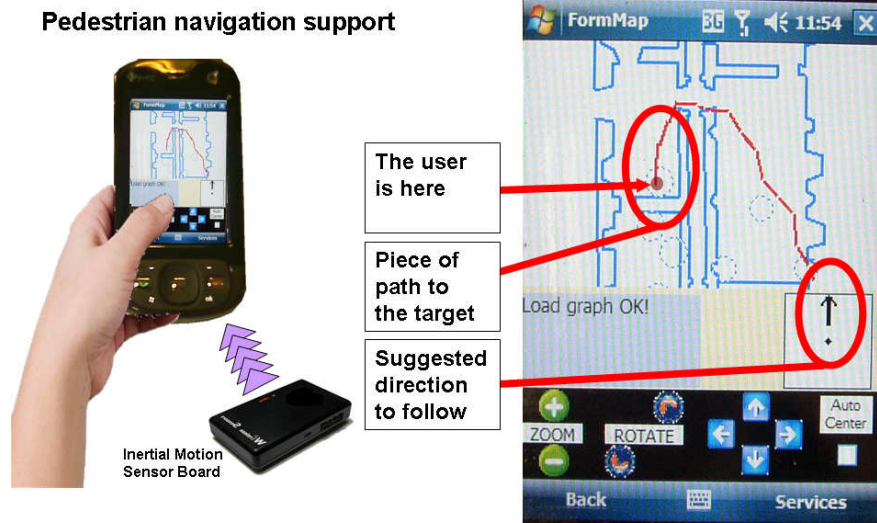
A complete tool chain was made to support the following functions:

- add a new map to the service
- starting from a 32bit BMP raster map image create a structured map graph representation
- change the map reference system
- simulate people walking on the area represented by the map

The service was enhanced by a dedicated and still unpublished image manipulation algorithm called *Bubbles* which automatically creates an optimized graph representation of the map and:

- reduces the memory amount occupation
- speeds up the path solving computation
- optimizes and makes “pedestrian-friendly” the path to follow

The service was integrated into a multimedia museum visitor guide. This drives the user to an exhibit by showing him the path to the target and the direction to follow. Incremental position information is provided by an inertial sensor platform.



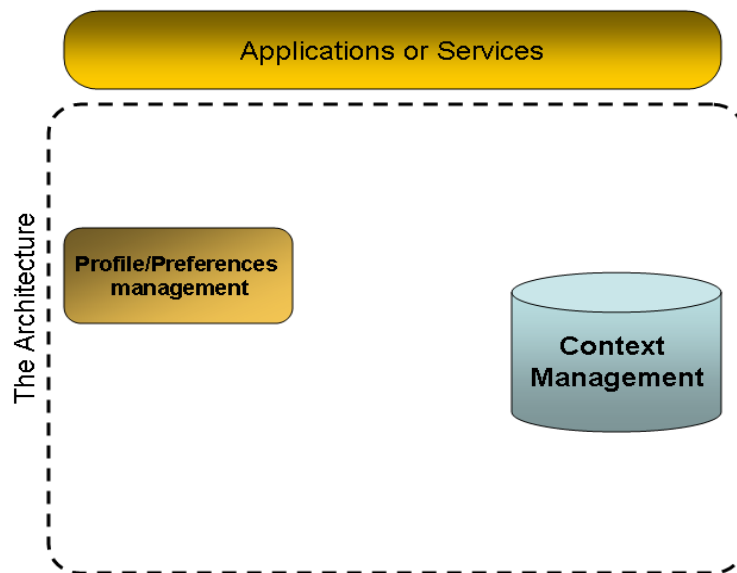
**Figure 21: Pedestrian navigation support: a user with a mobile device - enriched with a wireless inertial motion sensor board –enjoys the pedestrian navigation service . The path to the target and the direction to follow are shown (screenshot on the right)**

Fig.21 shows a screenshot of an application exploiting the service. In this paragraph a mechanism to guide a user to a target was shown. But, how is the target selected? Some support to implement user preference based selection policies is required. Next paragraph introduces an architecture module to handle context-dependent user preferences.

### 5.3 User preferences in service platforms

The user preferences are an important context coordinate. They complete the information related to the environment and they allow to model a service or application behaviour. The use case presented in this paragraph concerns a preference management system developed during a joint research program with Telecom Italia Lab (par. 4.2).

The idea was to extend Telecom Context-Aware-Platform (CAP) with an “activity suggestion service”, being the suggestion achieved by aggregating context and user profile and preferences. Main architecture components are shown in Fig. 22.



**Figure 22: Architecture components for a preference and context based recommendation service**

The CAP is a software platform for the management of *context* information. The term *context* indicates the collection of information available from the environment surrounding an entity, the terminal capabilities, the network connectivity, the preferences and the available services. Thus, this is a large amount of information that grows with time. A description of the CAP architecture follows.

The CAP is a comprehensive and distributed context-aware platform capable of aggregating and processing a variety of context information [58]. Fig. 23 shows how this architecture facilitates re-use and support for many applications from context acquisition to context usage or consumption.

The CAP has been designed according to the producer/consumer paradigm where some entities produce context, *i.e.* Context Providers (CP), while other entities consume context, *i.e.* Context Consumers (CC). These entities communicate with each other through a central entity named Context Broker (CB), which also provides some additional functions within the system (see Fig. 23). The main characteristics of the above mentioned entities follow.

**Context Broker (CB)** creates and manages the relationship between CPs and CCs. The CB performs *Context Source* discovery and management and it offers subject-based lookup services.

**Context Provider (CP)** is the logical point where context is detected and acquired or extracted from *Context Sources*. A CP supplies mechanisms for on-demand queries and it may optionally support subscription based notifications depending on defined event occurrences, *e.g.* general context change or timer expiration. Along with the acquisition of context data, a CP performs data aggregation, fusion and inference.

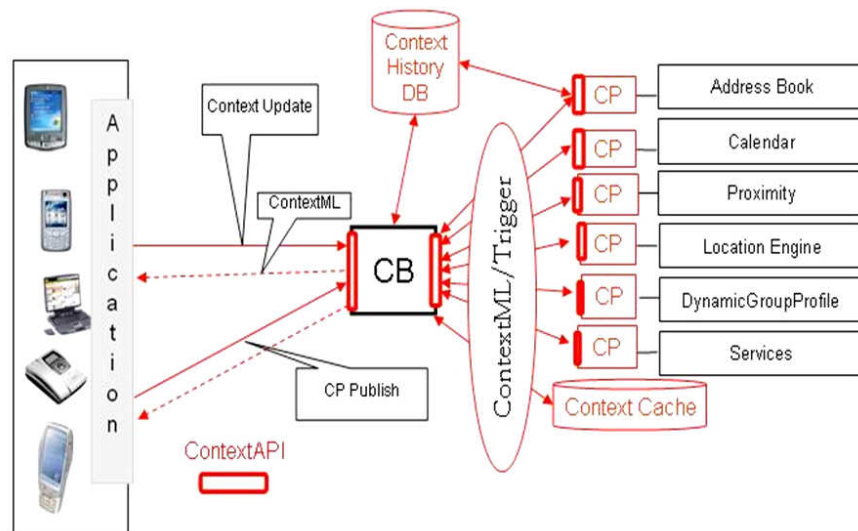


Figure 23. Context Management Architecture [75]



**Context Consumer (CC)** is the logical point where context is used (or consumed) accordingly to the context-aware service logic. CCs implement an event-based model to support a query-based publish-subscribe mechanism for context data notification.

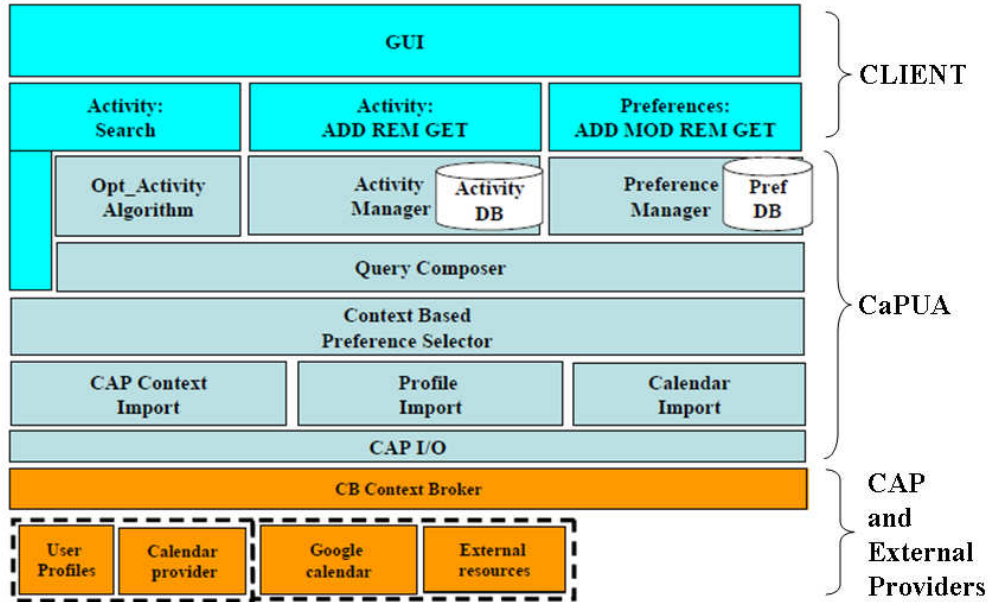
### 5.3.1 Preference based information processing model

The interoperability between CPs from different domains is based on a common language for context information representation named *ContextML* [59]. It is an XML-based language that all CPs need to comply with, in order to be registered in the CB and to enable potential CCs to discover the context information they need. To simplify the data management, context has been subdivided into *scopes*, namely a simple a priori aggregation of data with a semantic coherence, grouped together and identified by a name. Such names could easily be mapped to concepts within an ontology. Scopes can be atomic or aggregated, as the union of different atomic scopes.

Any information given by a CP is characterized by an entity and a specific scope. When a CP is queried, it returns the required data in a *ContextML* document, which contains the following XML elements:

- **contextProvider**: a unique identifier for the CP of the data;
- **entity**: the identifier of the entity which the data are related to;
- **scope**: the scope which the context data belongs to;
- **timestamp and expires**: the time in which the response has been created and the expiration time of the dataPart;
- **dataPart**: part of the document which contains actual information data which are represented by a list of name-value pairs through the *<par>* element (“parameter”). Parameters can be grouped through the *<parS>* (“parameter structure”) or *<parA>* (“parameter array”) elements if necessary.
- CAPUA was developed on top of the CAP. CaPUA is a Java Servlet that exploits user preferences and context. CaPUA exports a web GUI to allow user interaction – e.g. from a smart phone – but can be used directly from other applications that need to use the service directly through the HTTP protocol. My

job here was to participate to the development of the CaPUA architecture and to integrate an activity discovery service that exploits an optimized algorithm – called Opt\_Activity – able to handle qualitative preferences instead of the common quantitative preferences often adopted.



**Figure 24: Context-aware and user preferences activities suggestion service architecture**

This algorithm searches on the activities database to find all the activities most close to the user preferences and context.

The following service interface methods were implemented:

- add, remove, update activity
  - activity search
  - add, remove, modify preference
- update context information

Fig. 24 depicts the architecture split in client side, CaPUA server side and external services side – that include the CAP service. The communication protocol between a client and CaPUA is based on a set of XML messages.

So far, in this thesis, all services that exchange information need to know the meaning of the information exchanged. Thus, information interoperability was based on a

“service-to-service” agreement. Rising the level of information interoperability from service to platform would dramatically extend the scope of context-aware platforms, bringing in an inherent ability to support multi-domain applications.

Next section is dedicated to the project SOFIA and to the related impact on interoperability and multi-domain context aware services.

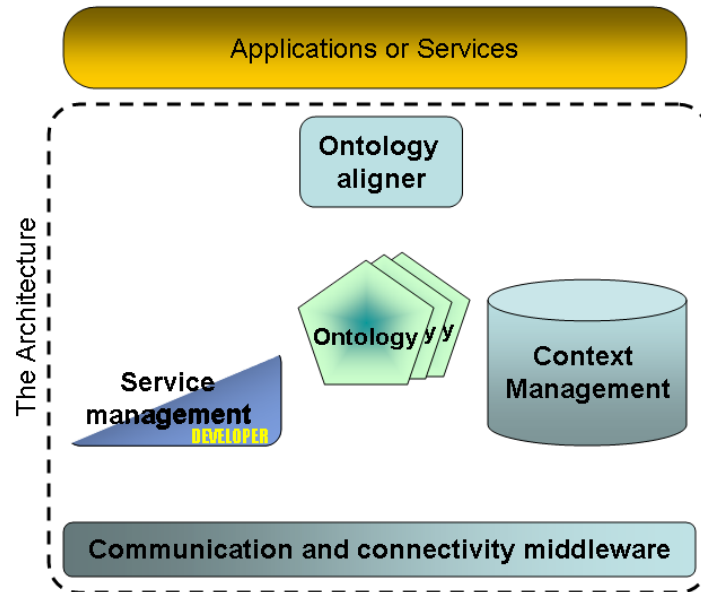
## 5.4 An Interoperability Platform for Smart Spaces

As already introduced in section 4.3, SOFIA goal is to provide an interoperability platform (IOP) to make the information originating from the physical world available to smart services and then to the users. This information is stored in Smart Spaces (SSs). Smart Space interoperability is based on information sharing and on adaptation to existing legacy devices and systems. Ontologies are adopted to share the semantics of the information and to assure a complete understandable interaction. Because the fulcrum is information, context management service is a central to SOFIA. On top of this service it is possible to build others services or applications that consist of a collection of small SW modules. These modules are independent each other i.e. they can be developed separately based on a previously defined and shared ontology. These SW modules are called *Knowledge Processors* (KPs) and are mostly developed in Python, even if they may also be coded in other programming languages.

In order to help developers to interact with the SS without the need to be expert on ontologies and related description languages some tool was developed. An example of these tools is the “Ontology to Python Tool” from Åbo Akademi University [66] which creates an ontology dependent library that “wraps” the ontology and takes the developer to a higher programming level, adopting a model based approach.

Within SOFIA, my contribution was to add side by side to the already available Python library a new library to enable Java KPs to interact with the Interoperability platform. This library is independent from development support tools. It is a SOFIA’s core SW module that interacts with the IOP directly. It can be used to implement data providers and data consumers.

Furthermore, as the architecture originally lacked of service oriented facilities, an OSGi (Open Service Gateway initiative) based interface to the IOP was introduced [60].



**Figure 25: an architecture example where information semantics is handled under ontologies constrains. This improves the interoperability at information level**

In the following paragraph I will introduce the project, the architecture, its actors and how they were integrated together. The last paragraph is dedicated to describe a proposal to add access security and service discovery mechanism.

### 5.4.1 Smart-M3 and interoperable context management solution

Smart-M3 [61] is an open source architecture [62] supporting a tuple-space and agent-based model of computation upon a semantic web substrate providing integration of different kinds of devices at the information level and thereby facilitating interoperability between applications and devices. It provides cross domain search extent of information. As an example, Smart-M3 allows an application developer who works on a specific mobile platform to access simultaneously and in a uniform way contextual information of a car, home, office, football stadium, etc. Smart-M3 stores information in RDF (Resource Description Language) [63] format, which is a W3C standard.

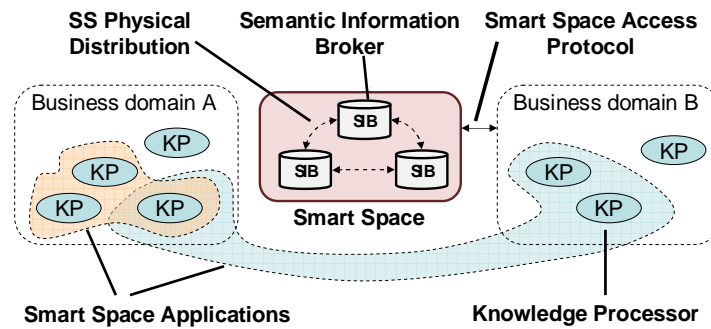


Figure 26: Smart-M3 Functional and Logical Architecture

First, RDF provides the ability to join data from vocabularies belonging to different business domains, without having to negotiate structural differences between the vocabularies. Second, adopting RDF allows to benefit of the Ontology Based Reasoning theory, practice and tools developed by the semantic web community. Wilbur query language [64] is also supported to enhance data retrieving. The Smart-M3 functional and logical architecture is shown in Fig. 26.

The following elements can be underlined:

- Smart Space (SS): a named search extent of information.
- Semantic Information Broker (SIB): an entity for storing, sharing and governing the information of one smart space as RDF triples, i.e. ontology.
- Triple governance transactions using an XML based communication protocol, namely Smart Space Access Protocol (SSAP). The SSAP primitives are: join, leave, insert, remove, update, query, subscribe, unsubscribe.
- Knowledge Processor (KP): any entity contributing to produce (*insert*, *remove*, *update*) and/or consume (*query*, *subscribe*) information in a SS. Common functionalities to access and use SS information are made available to KP by Knowledge Processor Interfaces (KPIs), i.e. KPIs hide SSAP details.
- SS Physical Distribution: allow the formation of SS by using multiple SIBs. SIBs physical distribution is hidden to KPs.
- Smart Space Application (SSA): a subset of KPs that use one (or more) SS as resource to perform the desired functionality.

As shown in Fig. 26, Smart-M3 architecture allows KPs running in different business domains to share interoperable information. Smart-M3 will also provide a set of common *convenience libraries* to facilitate the access and the usage of ontology based information, i.e. RDF triples are hidden. Smart-M3 is still under development and its access protocol, i.e. at device level, can be implemented by using any transport protocols, e.g. TCP/IP or NoTA.

## 5.4.2 Integrated service and information interoperability

A SOA approach can be a promising solution in order to optimize costs, provide ICT agility and guarantee software/systems evolution [20]. However, it is not enough: i) applications should be able to integrate easily and dynamically, ii) without writing custom specific glue code and iii) without extra costs. The philosophy behind OSGi [60] is an answer to these requirements. OSGi proposes a dynamic modular SOA application platform for Java™ that has been adopted by many enterprises in different application fields, e.g. home and office environment. OSGi specifications have been implemented in several certified platforms that have been adopted by a wide range of device vendors to empower their products with OSGi as a hosting platform for wired and wireless SOA oriented applications. Examples of these platforms are: Equinox [67], Knopflerfish [68], Apache Felix [69] and Concierge OSGi [70]. They define a standardized, component-oriented computing environment for general networked services that is intended to significantly increase the quality of the produced software. In OSGi the application emerges from a set of dynamic modules – called *bundles* – that collaborate with each other. Dependencies between bundles and associated consistency are automatically managed. Explicit sharing, i.e. importing and exporting, and automatic management of code dependencies are available for bundles, and their associated services, that may appear or disappear at any time. OSGi Service Platform specification adds in a networked device the capability to manage through the network the lifecycle of the software services exposed by the device itself [71]. Software services can be installed, updated, or removed in a controlled manner without having to disrupt the operations the device is doing [72], increasing maintainability, scalability and evolvability. The OSGi service oriented component model enables networked services to dynamically discover other services and work together to achieve the desired functionality [72]. It is also possible to receive event notification when a service emerges or changes its properties [73]. Services could be linked to devices. If a device is no more available, the services it offers are automatically unregistered. OSGi has a component-based architecture that enables device-side applications to load required components from the management server at run time [74]. In this sense, application components can be customized on demand,

e.g. if a new sensor is available or a new service appears, then specific module can be loaded and used smoothly.

The OSGi framework provides the following benefits:

- The modular approach based on bundles reduces the complexity, in terms of bundle development and in terms of system architecture. It contributes to the rationalization of applications and exploits reuse e relying on a huge community of bundles developer.
- OSGi framework is simple, because the core API is composed only of 30 classes and the entire framework consists of a JAR file of about 300KB. With this footprint it can be used on a large range of devices, with the only requirement of a minimal JVM.
- OSGi is a dynamic framework, where bundles can be updated on the fly and the associated services come and go dynamically. This dynamicity is fully supported by the framework and it is transparent to the developer. Bundles can be installed, started, stopped, updated, and uninstalled without bringing down the whole system.
- The framework is adaptive, because bundles can find out what capabilities are available on the system through a service registry and can adapt consequently the functionality they can provide.
- OSGi based solutions are easy to deploy, because the standard specifies how components are installed and managed. It supports a native versioning system, solving the big issue of JAR version management.
- It provides a new security model that leverages and hardens the Java fine grained security model but improves the usability by introducing a simple way to specify the security details of the bundles.
- It is not intrusive because it can run potentially in any existing facility and can be easily ported to almost any software environment. It can run everywhere there is a JVM, because it is entirely written in Java and the API uses only standard Java classes.
- It is available since 1998 and has been extensively used in several application contexts (automotive, mobile and fixed telephony, industrial automation, gateways & routers, private branch exchanges, etc.). It is supported in many development environments (IBM Websphere, SpringSource Application

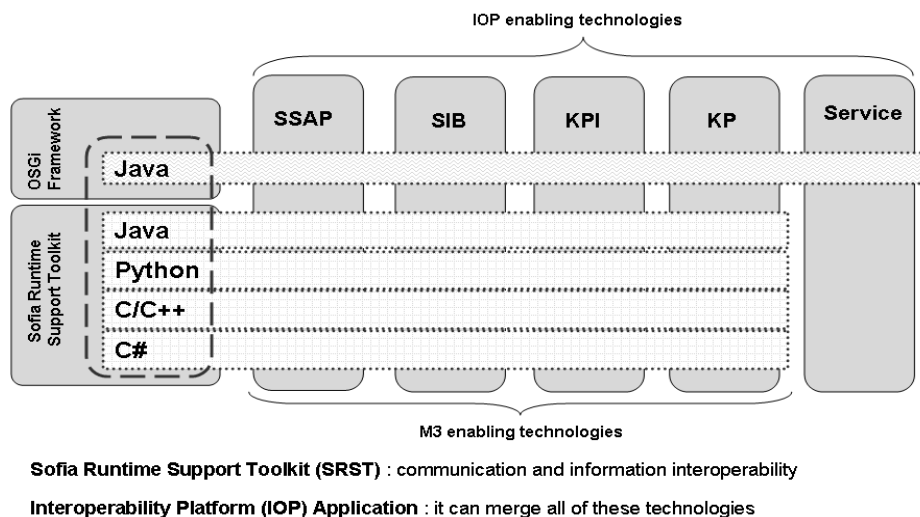


Server, Oracle Weblogic, Sun's GlassFish, Eclipse, and Redhat's JBoss) and by key companies (Oracle, IBM, Samsung, Nokia, IONA, Motorola, NTT, Siemens, Hitachi, Ericsson, etc.).

OSGi specifications do not concern with interoperability at the information level so the introduction of elements that allow information interoperability would represent an important improvement for OSGi platform.

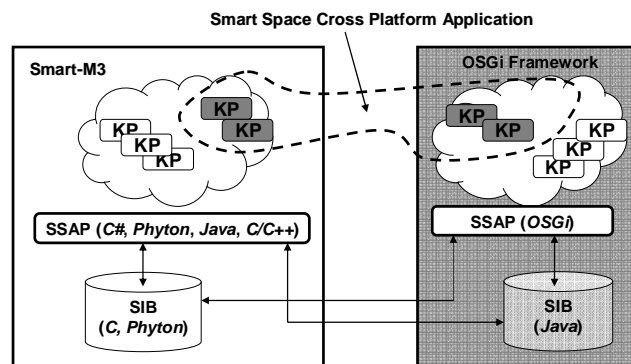
The functionalities of the information interoperability level in Smart-M3 are based on the interaction between the actors involved in a smart environment, i.e. KPs and SIBs, and on the common access protocol, i.e. SSAP. Porting these elements into OSGi, i.e. OSGi bundles, introduce the information interoperability level into OSGi allowing OSGi Smart Environments applications to exchange information and use services in a unified and simple way. From Smart-M3 side, it benefits from the entire OSGi Framework, both in terms of architecture and functionalities and in terms of software engineering – by introducing a modular service oriented architecture.

The result of this integration (see Fig. 27) is a dynamic interoperability service architecture where it is possible to publish a new service at runtime, discover and use services, share both raw data and high level information obtained from devices and sensors.



**Figure 27: Overview of the OSGi and SMART-M3 Integration**

The most important parts of the integration between Smart-M3 and OSGi are the SIB and the SSAP. The availability of a native OSGi SIB provides complete autonomy to OSGi by introducing the possibility to implement a complete OSGi based SSA. At the same time, from a communication point of view, the implementation of SSAP on OSGi allows legacy Smart-M3 application to interact with the OSGi SIB localization. With respect to the service oriented architecture of OSGi, a specific bundle for SSAP has been developed. This bundle provides an SSAP services for all the bundles that want to access a SS and it allows each OSGi based KP to communicate with every SIB in the SS. Fig. 27 describes the current status of Smart-M3 implementations, specifying the development language/platform and related support tools. The frame related to OSGi explains how the integration of Smart-M3 has been performed and which components have been involved.

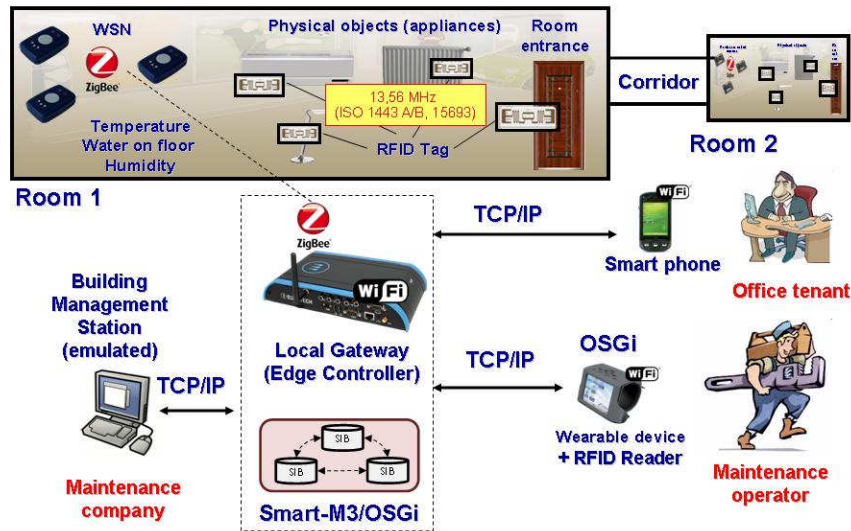


**Figure 28: Integrated Smart-M3/OSGi Solution**

The integration is fully compatible with existing Smart-M3 implementations, allowing developers to create multi-platform SSA as shown in Fig. 28: on the left side – the Smart-M3 KPs written in Python, Java, C#, C/C++ and – on the right side – the same modules implemented in Java as bundles running on the OSGi framework. Through the SSAP these modules can communicate and exchange information in both directions, i.e. Smart-M3 to OSGi, OSGi to Smart-M3.

### 5.4.2.1 Application of Integrated Service and Information Interoperability

In this paragraph, an application of the proposed solution to the maintenance of buildings and, in general, maintenance of indoor spaces (Fig. 29) is presented. In this scenario, maintenance needs should be *identified*, *specified* and *communicated* to the reference maintenance company, as early as possible. For flexibility reasons any authorized maintenance company may subscribe to any maintenance action needed. As shown in Fig. 29, the main actors involved in the demonstration are: the maintenance company, maintenance operators and office tenants. In the demonstration, maintenance needs are represented by faults related to environments. Faults are automatically detected on the base of the environmental conditions, i.e. temperature, humidity, presence of water. Environmental conditions are gathered from a set of EUROTECH ZigBee sensors spread across the environment. A EUROTECH local gateway acts as a controller of the Wireless Sensor Network (WSN) and it hosts Smart-M3/OSGi. The local gateway provides two different interfaces: SSAP through a TCP/IP connection and OSGi. The former is used by non OSGi compliant devices, i.e. the office tenant device and the building management station, and the latter by OSGi compliant devices, i.e. a wearable device used by maintenance operators. Relevant items that need to be identified are tagged using RFID tags, e.g. Heating Ventilation Air Conditioning (HVAC) system, light, window (Fig. 29). RFID tags are also placed on the entrance door of each environment. In the first instance, this information will be used to locate maintenance operators.



**Figure 29: System Architecture of the Maintenance Scenario Demonstration**

According to the Smart-M3 vision, the application is composed by a set of KP – see Fig. 30 - , each of them with a specific function:

*WSN Manager* acts as a legacy adapter. It gathers data from wireless sensors and provides information to the Smart-M3/OSGi in terms of RDF triples.

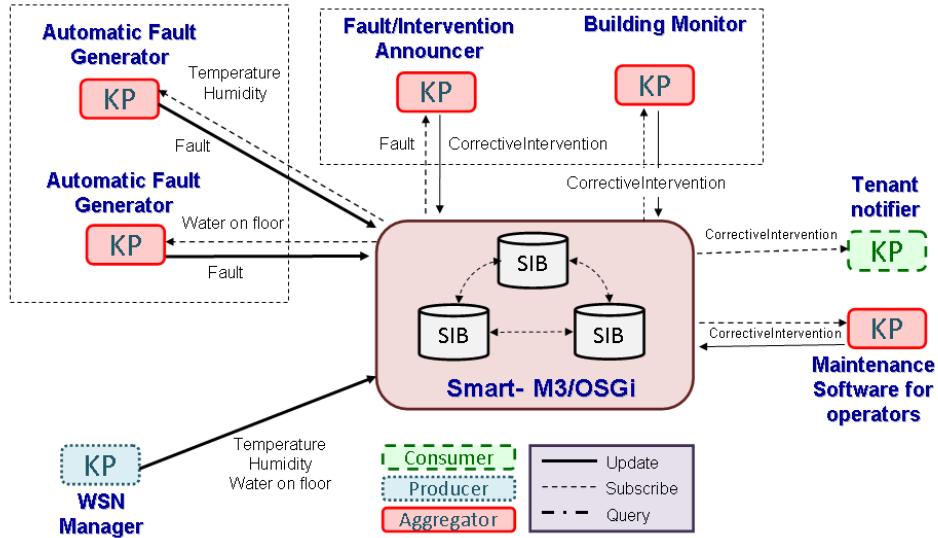
*Automatic Fault Generators* starting with environmental information, i.e. temperature, humidity, presence of water, and a set of rules, detect the presence of an anomalous condition, i.e. a fault. In the demonstration the rules consist of simple threshold-based conditions.

*Fault/Intervention Announcer* is subscribed to the presence of fault instances. When it receives a notification of a new fault, it updates Smart-M3/OSGi with a request of a corrective intervention related to the detected fault.

*Tenant Notifier* is subscribed to corrective interventions related to the environment, i.e. an office, of a tenant. It notifies the tenant of the time and time expected for the intervention. The tenant is also notified at the end of the intervention.

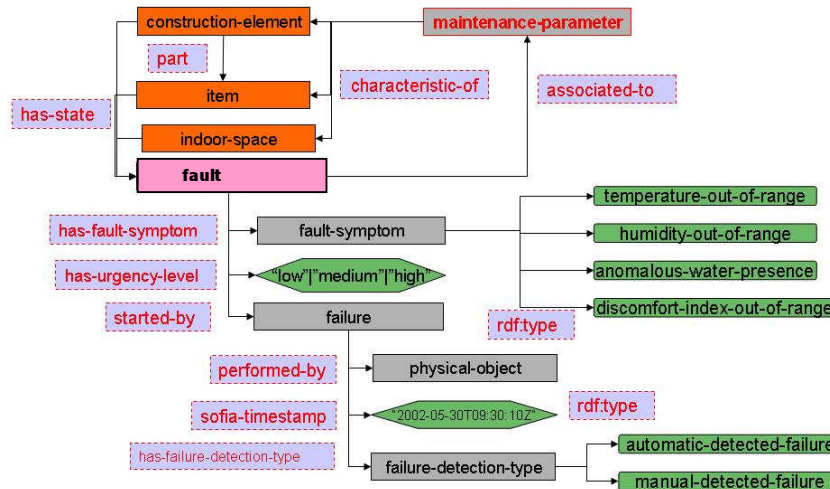
*Maintenance Software for Operators* is subscribed to corrective interventions. When it receives a notification of a new intervention request it asks the user, i.e. maintenance operator, if he wants to take care of the intervention. When the operator reaches the place of intervention, he verifies if being in the right place by reading the RFID on the entrance door. This operation also sets the intervention start time.

*Building Monitor* is subscribed to corrective intervention. This KP presents a user interface, where all relevant information are shown, e.g. fault location, fault time, relevant parameters, scheduled intervention date and time.



**Figure 30: Software Architecture of the Maintenance Scenario Demonstration in Terms of KPs**

Information interoperability is achieved by using Smart-M3/OSGi with a domain specific ontology. In Fig.31, as an example, a snapshot of the ontology used to describe a fault is depicted.

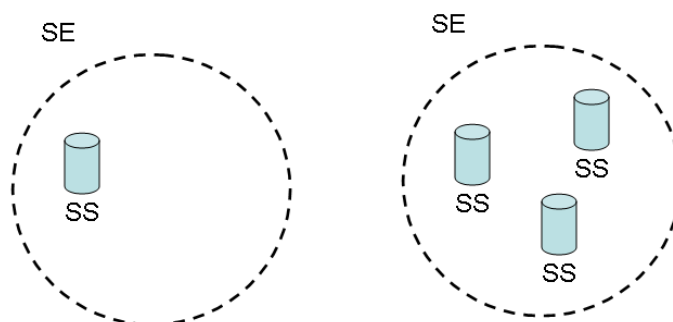


**Figure 31: Snapshot of the Ontology Used for the Maintenance Scenario**

When creating a SE becomes a concrete opportunity for a real implementation, challenging issues like authenticated security access and service discovery must be considered. The next paragraph is dedicated to discuss about a possible solution to these challenges.

### 5.4.3 Service discovery and access control proposal

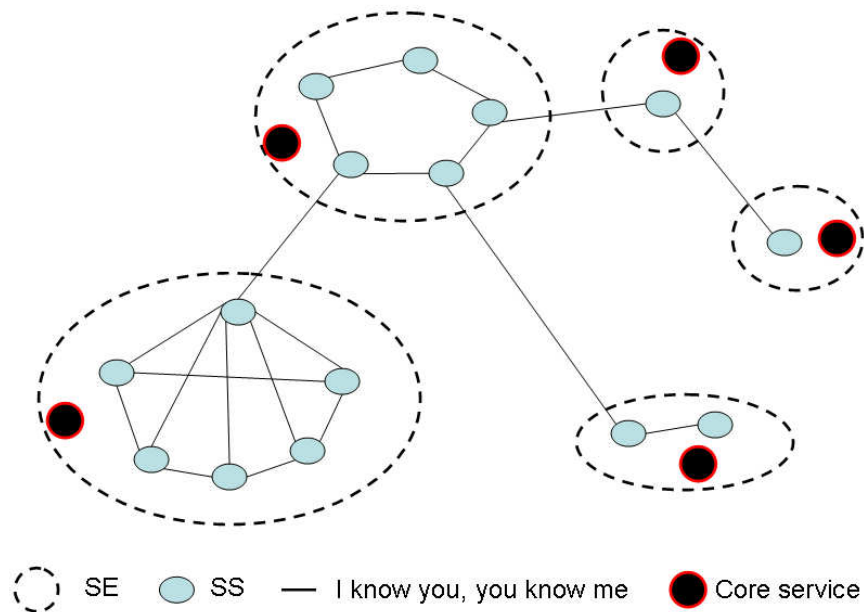
*Security-and-trust* and *service discovery* are fundamental topics in many use cases. Often it is important to ensure that only a subset of users may discover and access a service. [3] is a comprehensive survey on service discovery protocols. This paragraph introduces an idea for a future implementation of service discovery and access control mechanisms on top of the architecture presented in 5.4.1. Basically the idea is to put side by side to the SIB a core service. All the information useful to reach this core service should be available on the relative SIB in the relative SE. Every KP should know how to obtain this information from the *local* SIB – because this is a core service. Some information handled by the core service may be useful to the same service in another SE so this information should be able to migrate from one SS to another one. The scenario consists of a set of SSs that can be related to a set of SEs through an N-to-M relation, i.e. one SE can include more than one SS, as shown in Fig. 32.



**Figure 32: the environment can be split in many SEs. Every SE can contain one or more SS, depending on the use case**

Several system topologies are suitable to link together every SS and to put a core service next to the SS. One of these topologies could be a network where every node is a core service linked to one or more SSs. This requires that every SIB contains information to reach all other SIBs. This could be done with a hierarchical topology where not all the SSs know the other SSs but only one special candidate can be used as a gate to the others (Fig.33).

This topology could be similar to a distributed database. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network where a distributed database management system (DDBMS) manages the distributed databases and makes this distribution transparent to the user. This helps when the information to handle could be put all together. In other cases – when it is necessary to keep apart every set of information especially if they refer to different ontologies – a scalable solution that can evolve during the time should be a better solution.



**Figure 33: an example on how to model the topology of the available SSs and core service**

Fig. 33 shows a SE distribution example with the relative SS and core service. More than one core service can be present into the same SE, the concept does not change.

In this scenario a special purpose KP can be designed to *feel* when a new user enters the SE and to retrieve the user sensitive information from the other SSs by browsing the network. The user can carry very little information like his/her identifier and the last SE visited. The user can enjoy the smooth propagation of his sensitive data and does not need to provide it again when enters a new SE, as the platform recognizes the user from the above mentioned minimum information. All services in the new SE can exploit the user sensitive data propagated through the network.

Likewise an *access control service*, should offer functionalities to subscribe to the service and to check if a user is subscribed and recognized by the system. When a user tries to use a service, this can understand if the user is able to use it and his role, by using the access control service. In this way unauthorized use is prevented. Security-free service can coexist with the secure ones. The latter will be more complex, while the former will keep unmodified. SOFIA Architecture doesn't change but it requires a shared ontology to handle the user/service interaction.

The *service discovery service* should offer functionalities to subscribe to a new service and to search for a service. This could be done, for example, with a reasoning engine based on the user preferences and profile.

The way to interact with core services should be standardized and it should be known from all KPs. An ontology that describes services and user types should be defined to fix which user type can use a service.





# Chapter 6

## Conclusions

My research intended to investigate which architecture components can be devised and combined to enable multi-level interoperability in context-aware computing.

The following interoperability levels were considered:

- Communication and connectivity
- User Device
- Information
- Service (discovery and composition)
- Content (adaptation)

Some solutions were proposed in terms of architecture and systems components. Considering both the developer and the user point of views, their aim was to implement an extended interoperability concept first, and then to provide a mature architecture for context-aware services.

With reference to fig. 1 (pg. 9), rearranged into fig. 34 and starting from the device/communication level, I am now going to summarize the results of my experience.

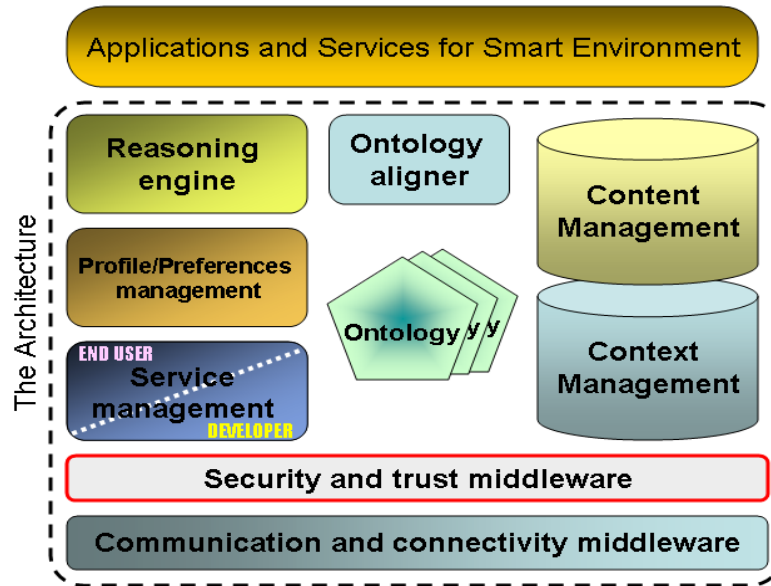


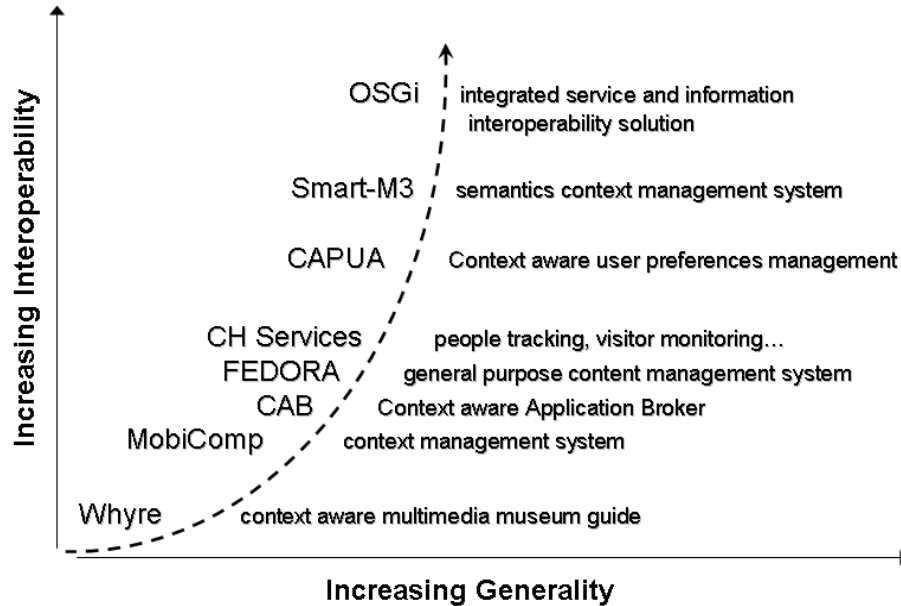
Figure 34: architecture components for extended interoperability

The proposed solution is based on the “separation of concerns” principle. Each module can base its behaviour on the others if they are available.

Often legacy architectures are limited by the devices involved and by the services offered which depend on the specific use case considered.

By increasing the level of abstraction of the architecture and the generality of actors and modules belonging to the architecture - the interoperability will improve too.

This should be considered when different scenarios are fused and evolve into a new, more general smart environment configuration. Based on the roadmap of my research, activity, fig. 33 shows how the increasing generality of actors and tools leads to increasing interoperability levels.



**Figure 33: From dedicated context-aware platforms to interoperable smart spaces for multi-domain context aware services**

The underlying vision is that when all information about the surrounding environment is easily available, the life quality will greatly benefit, as the variety of applications that that could get an added value by making their behaviour dependent on the environment “state” is only limited by fantasy [1].

The first requirement for an extended interoperability concept is an architecture component to handle communication and connectivity between heterogeneous multivendor equipment.

The strategy is to provide a framework able to transparently manage the connection/communication channel. This allows to forget issues like:

- communication protocols between multivendor devices
- best communication channel to use
- communication channel interoperability

A solution in this respect could be provided by a service architecture that can export communication facilities – e.g. NoTA [47].

Then the requirement arises to handle services and provide mechanisms like service discovery, service composability and service interoperability. This was shown in Section 5 where a service platform for cultural heritage applications was analysed.

Then the need for information interoperability comes into play. Data sharing, as currently implemented for example in the context management framework MobiComp (Par. 5.2.1) is just a step forward but it is not enough. A shared agreement on information semantics is also needed.

This work is about smart environments, therefore the term *information* has been mostly associated to “context”. Context information – in a SE for advanced services– could not just be stored or exchanged as is, because the possible differences in the associated semantics could be source of lack of mutual understanding.

A popular solution is to add information to information by adopting ontologies. Ontologies define the domain, the entities involved and their properties. By adopting ontologies it is possible to prevent a wrong interpretation of shared data and to help in case of need for data alignment in different domains.

Ontologies can be also adopted to describe not only the user and his environment but also services and security mechanism that are going to improve the quality of the entire platform. This information representation moves the scene from machine readable information to machine interpretable information. Cross domain application become more natural and easy to develop.

On top of this substrate of communication and information interoperability, new advanced services can be developed for multi vendor sensors and devices, such as the tool chain developed for the cultural heritage domain (Section 5).

By exploiting information interoperability, interoperable connectivity and communication facilities together with the digital representation of the environment, services can be composed to form new and more advanced services; in this way appropriate answers to the growing demand for new multi-domain and cross-domain services may be provided.

In this respect a service oriented mechanism based on the OSGi framework was added to an architecture natively devised for information level interoperability (see section 5.4). OSGi introduces standardized mechanisms to integrate easily and dynamically new platform services. As shown in section 5.4.2, many OSGi features can be exploited to support service discovery by an application. The same mechanism may be used to enable service discovery for the end user – e.g. to discover services like a museum guide service or a pedestrian navigation service. Furthermore the mechanism shown in section 5.4.3 could be implemented using OSGi and could be exploited to extend the service discovery feature to not-OSGi based services.

Zero-effort interfaces and context-aware smooth user interaction models should now be investigated to maximize user satisfaction in multidomain service-based applications offered by future interoperable smart environments.



# Publications

1. D. Manzaroli, L. Roffia, E. Ovaska, P. Azzoni, T. Salmon Cinotti, V. Nannini, S. Mattarozzi, *Smart M3 and OSGi: The Interoperability Platform* accepted for presentation at the International Workshop on Semantic Interoperability for Smart Spaces (SISS 2010) in association with the IEEE Symposium on Computers and Communications (ISCC 2010)
2. L.Roffia, S. Bartolini, D. Manzaroli, A. D’Elia, T. Salmon Cinotti, *Requirements on System Design to Increase Understanding and Visibility of Cultural Heritage*, accepted chapter for “*Handbook of Research on Technologies and Cultural Heritage*” Edited by Dr. Georgios Styliaras, Dr. Dimitrios Koukopoulos and Dr. Fotis Lazarinis of the University of Ioannina
3. L. Roffia, L. Lamorte, G. Zamagni, S. Bartolini, A. D’Elia, F. Spadini, D. Manzaroli, C. A. Licciardi, T. Salmon Cinotti, *Personalized Context Based Services for Dynamic User Groups* at 2nd International Workshop on Social Aspects of Ubiquitous Computing Environments (SAUCE2009, October 12, 2009), 5th IEEE International Conference on wireless and mobile computing, networking and communication (Marrakech, Morocco, October 12-14, 2009)
4. N. Ryan , P. Mohr, D. Manzaroli, G. Mantovani, S. Bartolini, A. D’Elia, M. Pettinari, L. Roffia, L. Sklenar, F. Garzotto, T. Salmon, “*Interoperable multimedia mobile services in cultural heritage sites*” EPOCH Conference on Open Digital Cultural Heritage Systems, Roma, (2008)
5. T. Salmon Cinotti, M. Pettinari, L. Roffia, D. Manzaroli, S. Bartolini, “*Technology meets Culture: from MUSE to EPOCH*”, International meeting Vesuviana, “Archeologie a confronto”, Bologna, 2008, Antequem edition, pp 845-857
6. D. Manzaroli, P. Lacchè, M. Pettinari, L. Roffia, A. D’Elia, T. Salmon Cinotti (2008) *Enhancing Social Life with Path Solvers, Rendez-vous without Constraints on MeetingPlace and Time* - SAUCE 2008 1st International Workshop on Social Aspect of Ubiquitous Computing Environment (October 12, 2008) held in conjunction with WiMob-2008, 4th IEEE International Conferences on wireless and mobile computing, network and communication (Avignon,France, October 12-14, 2008)



7. G. Raffa, P. H. Mohr, N. Ryan, D. Manzaroli, M. Pettinari, L. Roffia, S. Bartolini, L. Sklenar, F. Garzotto, P. Paolini, T. Salmon Cinotti, *CIMAD - A Framework for the Development of Context-aware and Multi-channel Cultural Heritage Services*, in ICHIM07 September 30, 2007 Toronto
8. M. Pettinari, D. Manzaroli, S. Bartolini, L. Roffia, G. Raffa, L. Di Stefano, T. Salmon Cinotti, *A Stereo Vision based system for advanced Museum services*, EPOCH Publication, The Integration of Location Based Services in Tourism and Cultural Heritage, edited by David Pletinckx, *Workshop Proceedings*, Page(s): 57 - 68, ISBN:978-963-8046-88-8 Ed. ARCHAEOLOGIA, 2007
9. N. Ryan, G. Raffa, P. H. Mohr, D. Manzaroli, L. Roffia, M. Pettinari, L. Sklenar, L. Di Stefano, T. Salmon Cinotti, (2007) *A Smart Museum installation in the Stadsmuseum in Stockholm - From Visitor Guides to Museum Managemen*, in The Integration of Location Based Services in Tourism and Cultural Heritage. (pp. 83-90). ISBN: 978-963-8046-88-8
10. N. Ryan, T. Salmon Cinotti, D. Manzaroli, M. Pettinari, G. Raffa, L. Roffia, S. Ghosh, P. Mohr, L. Sklenar, *Smart Museums, Sites and Landscapes - From Visitor Guides to Collection Monitoring* in Interactive Salon - New Technology for Visitors in Cultural Heritage. (pp. 10 - 13), HALINA GOTTLIEB, HANS OEJMYR, STOCKHOLMS STADSMUSEUM publication, STOCKHOLM 2006

# Bibliography

- [1] F. Spadini, S. Bartolini, R. Trevisan, F. Vergari, G. Zamagni, A. D'Elia, L. Roffia, T. Salmon Cinotti, *Approaching the design of interoperable smart environment applications*, in 2nd International Nota Conference, 2009.
- [2] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary, *A Compilation of IEEE Standard Computer Glossaries*. New York, NY, 1990.
- [3] A. Mian, R. Beraldi and R. Baldoni, A closer Investigation of Service Discovery Protocols in Multihop Mobile Ad-Hoc Networks, IEEE Pervasive Computing, 2009
- [4] Sofia work package 5.11 deliverable: *Interoperability Platform Principles*: <http://www.sofia-project.eu> .
- [5] Peristeras, B., Tarabanis, K. *The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C4IF) for Information Systems interoperability*, in IBIS - Interoperability in Business Information Systems, Issue 1, 2006.
- [6] Tolk, A., S. Y. Diallo, C. D. Turnitsa, and L. S. Winters. 2006. *Composable M&S Web services for netcentric applications*, in Journal for Defense Modeling and Simulation, 27-44.
- [7] Park, J., Ram, S. *Information systems interoperability: What lies beneath?*, in ACM transactions on information systems, vol. 22, no 4, Oct. 2004, pp. 595-632.
- [8] Guédria, W., Naudet, Y., Chen, D. *Interoperability Maturity Models - Survey and Comparison*, in On the move to meaningful Internet systems: OTM 2008 workshop, LNCS, Vol. 5333/2009, pp. 273-282.
- [9] Brailer, D. *From the Field. Interoperability: The key to the future health care system*, in Health affairs - Web Exclusive, W5-19-21.
- [10] Shabo, A., Rabinovici-Cohen, S., Vortman, P. *Revolutionary impact of XML on biomedical information interoperability*, in IBM Systems Journal, Vol. 45, no. 2, 2006, pp. 361-372.

- [11] Open Health Tools, <http://www.openhealthtools.org>
- [12] Garzotto, F., Manzaroli, D., Paolini, P., Salmon Cinotti, T., Sklenar, L., Raffa, G., Roffia, L., Bartolini, S., Pettinari, M., Ryan, N., Mohr, P., CIMAD/EPOCH, *A Framework for Developing Customized Multi-channel Cultural Heritage Services*, in International Cultural Heritage Informatics Meeting - ICHIM07: Proceedings 2007, Archives & Museum Informatics.
- [13] Peristeras, V., Tarabanis, K., Loutas, N. *Cross-Border Public Services: Analysis and Modelling*, in Proceedings of the 40th Hawaii International Conference on System Sciences, 2007, 7 p.
- [14] Tursi, A., Panetto, H., Morel, G., Dassisti, M. *Ontology-based products information interoperability in networked manufacturing enterprises*, in Proceedings of the IFAC conference on Cost Effective Automation in Networked Product Development and Manufacturing, IFAC-CEA'07. October 2-5, Monterrey, Mexico. 9 p.
- [15] O'Reilly, T., Battelle, J. *Web Squared: Web 2.0 Five Years On*, in Special Report. Web 2.0 Summit. 13 p.
- [16] A. K. Dey, G. Abowd, and D. Salber. *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*, in Human Computer Interaction (HCI) Journal, 16:97-166, 2001.
- [17] A. K. Dey. *Understanding and using context*, in Personal and Ubiquitous Computing, 5(1):4-7, 2001.
- [18] T. Strang and C. Linnhoff-Popien. *A context modeling survey*, in Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004.
- [19] H.Truong, S. Dustdar. *A Survey on Context-aware Web Service Systems*, in International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2 No.4, pp.263-77, 2007 .
- [20] S. Mello, R. Correa, *From Credit Cards to Gift Registries*, in IMB Web Seminar, Connecting everything with Smart SOA Connectivity & Integration (2009), Webcast.

- [21] Jong-yi Hong, Eui-ho Suh, Sung-Jin Kim, *Context-aware systems: A literature review and classification*, in Expert Systems with Applications Volume 36, Issue 4, May 2009, Pages 8509-8522.
- [22] W. N. Schilit. *A System Architecture for Context-Aware Mobile Computing*, in PhD thesis, Columbia University, 1995.
- [23] Jini Technology. <http://www.jini.org>.
- [24] Service Location Protocol. Internet Engineering Task Force Request For Comments 2608.
- [25] Overview of SGML Resources. <http://www.w3.org/MarkUp/SGML>.
- [26] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, 2006. <http://www.w3.org/TR/2006/REC-xml-20060816/>.
- [27] K. Henriksen, J. Indulska, and A. Rakotonirainy. *Modeling context information in pervasive computing systems*, in F. Mattern and M. Naghshineh, editors, Pervasive, volume 2414 of Lecture Notes in Computer Science, pages 167-180. Springer, 2002.
- [28] T. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, in Morgan Kaufmann Publisher, 2001.
- [29] A. Schmidt and K. van Laerhoven. *How to build smart appliances*, in IEEE Personal Communications.
- [30] P. D. Gray and D. Salber. *Modelling and using sensed context information in the design of interactive applications*, in M. R. Little and L. Nigay, editors, EHCI, volume 2254 of Lecture Notes in Computer Science, pages 317-336. Springer, 2001.
- [31] A. Ranganathan and R. H. Campbell. *A middleware for context-aware agents in ubiquitous computing environments*, in M. Endler and D. C. Schmidt, editors, Middleware, volume 2672 of Lecture Notes in Computer Science, pages 143-161. Springer, 2003.
- [32] M. Berchtold, C. Decker, T. Riedel, T. Zimmer, and M. Beigl. *Using a context quality measure for improving smart appliances*, in ICDCS Workshops, page 52. IEEE Computer Society, 2007.
- [33] T. R. Gruber. *A translation approach to portable ontology specifications*, in Knowledge Acquisition, v.5 n.2, p.199-220, June 1993

- [34] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung: *Ontology Based Context Modeling and Reasoning using OWL*, in Pervasive Computing and Communications Workshops, IEEE International Conference, 2004
- [35] H. Chen, F. Perich, T. W. Finin, and A. Joshi. *Soupa: Standard ontology for ubiquitous and pervasive applications*, in *MobiQuitous*, pages 258-267. IEEE Computer Society, 2004.
- [36] EPOCH (2004). *The European Network of Excellence on ICT Applications to Cultural Heritage*, in Contract no. IST-2002-507382. Epoch Repository. Retrieved December 22, 2009, <http://public-repository.epoch-net.org/presentations/EPOCH-Presentation.pdf>
- [37] EPOCH project: [http://www.epoch-net.org/index.php?option=com\\_content&task=view&id=231&Itemid=360](http://www.epoch-net.org/index.php?option=com_content&task=view&id=231&Itemid=360)
- [38] MobiComp, Ryan, N., Salmon Cinotti, T., & Raffa, G. (2005). *Smart Environments and their Applications to Cultural Heritage*, in Proceedings of a workshop held in conjunction with UbiComp'05 (Ed.), EPOCH Publication, Tokyo 2005 (pp. 7-11). Archaeolingua Ed
- [39] N. Ryan , P. Mohr, D. Manzaroli, G. Mantovani, S. Bartolini, A. D'Elia, M. Pettinari, L. Roffia, L. Sklenar, F. Garzotto, T. Salmon. "Interoperable multimedia mobile services in cultural heritage sites" EPOCH Conference on Open Digital Cultural Heritage Systems (2008)
- [40] Interactive Salon, *A touring exhibition about new technologies and concepts for communication with visitors in the context of cultural heritage*, <http://www.tii.se/projects/interactivesalon>
- [41] TiLab - TELECOM Italia research group: <http://telecomitalia.mobi/en/chisiamo/rs.html>
- [42] Marco Marengo, Nicoletta Salis, Massimo Valla., *Context Awareness: servizi mobili "su misura"*
- [43] L. Roffia, L. Lamorte, G. Zamagni, S. Bartolini, A. D'Elia, F. Spadini, D. Manzaroli, C. A. Licciardi, T. Salmon Cinotti, *Personalized Context Based Services for Dynamic User Groups*, in WiMob, pp.411-416, 2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2009

- [44] SOFIA project - Smart Objects For Intelligent Applications, <http://www.sofia-project.eu>
- [45] Artemisia: <http://www.artemisia-association.org/sofia>
- [46] ARTEMIS European Technology Platform - Advanced Research & Technology for EMbedded Intelligence and Systems, <http://www.artemis.eu>
- [47] NoTA World, Open Architecture Initiative, <http://NoTAWorld.org>
- [48] DUCATI Energia: <http://www.ducatienergia.it>
- [49] Salmon, C. T., Nagaraj, R., Mincolessi, G., Sforza, F., Raffa, G. Roffia, L. (2004). *WHYREe: A context-aware wearable computer for museums and archaeological sites*, in proceedings of the 8th IEEE International Symposium on Wearable Computers (ISWC2004), Arlington, Virginia, pp 174-175.
- [50] G. Raffa, L. Roffia, M. Pettinari, R. Nagaraj, F. Sforza, G. Mincolessi, T. Salmon Cinotti, *Context-aware computing for Cultural Tourism – Experiences from the MUSE project*, in: “The Integration of Location Based Services in Tourism and Cultural Heritage. (pp. 69 - 81), Daniel Pletinckx Editor. ISBN: 978-963-8046-88-8. Published by Archaeolingua, 2007, Budapest
- [51] Arces Laboratory, MML group: <http://www.arces.unibo.it/content/view/51/290/>
- [52] FieldMap Landscape Mapping : <http://www.fieldmap.cz>
- [53] Java Native Interface: <http://java.sun.com/j2se/1.4.2/docs/guide/jni>
- [54] FEDORA, Lagoze, C., Payette, S., Shin, E., & Wilper, C. (2006), *Fedora: an architecture for complex objects and their relationships*, in International Journal on Digital Libraries 6(2), 124-138.
- [55] Fedora content management: <http://www.fedora-commons.org/about>
- [56] Arces Laboratory, Alma Vision: <http://www.arces.unibo.it/content/view/55/290/>, <http://www.arces.unibo.it/content/view/43/290/>
- [57] Klein, M., Fensel, D., Van Harmelen, F., Horrocks, I., (2001) *The relation between ontologies and XML schemas*, Electronic Articles in Computer and Information Science
- [58] L. Lamorte, C.A. Licciardi, M. Marengo, A. Salmeri, P. Mohr, G.Raffa, L. Roffia, M. Pettinari, T.S. Cinotti, *A platform for enabling context aware*

*telecommunication services*, in Proc. Third Workshop on Context Awareness for Proactive Systems, Guildford,UK (2007)

[59] R. Reichle, M. Wagner, M.U. Khan, K. Geihs, M. Valla, C. Fra, N. Paspallis, G. A. Papadopoulos, *A Context Query Language for Pervasive Computing Environments*, in Proceedings of 5th IEEE Workshop on Context Modeling and Reasoning (CoMoRea '08) in conjunction with the 6th IEEE International Conference on Pervasive Computing and Communication (PerCom), 2008

[60] OSGi™ - The Dynamic Module System for Java™, <http://www.osgi.org>

[61] Smart-M3 Wiki, <http://en.wikipedia.org/wiki/Smart-M3>

[62] Smart-M3 Open Source Project, <http://sourceforge.net/projects/smart-m3>

[63] RDF Semantics, <http://www.w3.org/TR/rdf-nt>

[64] O. Lassila. *Enabling Semantic Web Programming by Integrating RDF and Common Lisp*, in Proceedings of the First Semantic Web Working Symposium. Stanford University, 2001.

[65] XPath query: <http://www.w3.org/tr/xpath20>.

[66] Åbo Akademi University, Ontology to Python tool, <http://sourceforge.net/projects/smart-m3>

[67] Equinox, <http://www.eclipse.org/equinox>.

[68] Knopflerfish Open Source OSGi Service Platform, <http://www.knopflerfish.org>.

[69] Apache Felix, <http://felix.apache.org/site/index.html>.

[70] Concierge OSGi, <http://concierge.sourceforge.net>.

[71] OSGi Technical Whitepaper,

<http://www.osgi.org/wiki/uploads/Links/OSGiTechnicalWhitePaper.pdf>

[72] OSGi Core Specification, [www.osgi.org/download/r4v41/r4.core.pdf](http://www.osgi.org/download/r4v41/r4.core.pdf).

[73] OSGi Initiative, OSGi Service Platform R4, 2007,

<http://www.osgi.org/Release4/HomePage>.

[74] H. Chang, T. Chun Jie, *A new strategy of language pack management for wireless applications*, in IBM Shanghai Globalization Laboratory (SGL), 2005, Webcast.

- [75] L. Lamorte, C. Venezia, *Smart Space a new dimension of context*, in PERSIST Workshop on Intelligent Pervasive Environments, at the Edinburgh Convention Centre at Heriot Watt University in Edinburgh, Scotland, AISB 2009 Convention, 2009
- [76] G. Raffa, *Context-Aware Computing In Smart Environments*, PhD Thesis, University of Bologna, Faculty of Engineering, DEIS Department 2005





# Tools

This section lists the tools used.

<b>Tool</b>	<b>Description</b>
Eclipse	Java, Python IDE
C++Builder 6	C/C++ IDE
Microsoft Visual Studio 2005	C/C++ IDE
NetBeans IDE 6.1 - C++	Java and C/C++ IDE
Apache ANT	Java build tool
Dreamweaver MX	HTML, PHP, JavaScript IDE
PAPYRUS	Open source tool for graphical UML2 modelling
Apache	HTTP Server
Apache Tomcat	Java Servlet Server
JVM (1.4, 1.6)	Java Virtual Machine
EasyPHP	PHP and MySQL administration tool
MySQL	Database management system
ArcView GIS 3.2	Maps geo-processing tool
GIMP 2	Image manipulation program
notepad++	Advanced text editor



# Acronyms

SW	Software
DLL	Dynamic Link Library
SOA	Service Oriented Architecture
XML	Extensible Markup Language
UML	Unified Modeling Language
GUI	Graphic User Interface
OSGi	Open Service Gateway initiative
IDE	Integrated Developing Environment
DDBMS	Distributed Database Management System
MVC	Model View Controller
HCI	Human-Computer Interaction
CH	Cultural Heritage
MID	MobiComp ID
POI	Point Of Interest
MMPI	MobiComp Multi Platform Interface
VTS	Video Tracking System
ITS	Inertial Tracking System
MEL	MobiComp Enabler Library
PTA	People Tracking Application
CAP	Context Aware Platform
CaPUA	Context-aware Preferences, Users, and Activities
SOFIA	Smart Object For Intelligent Application
CIMAD	Common Infrastructure/Context Influenced Mobile Acquisition and Delivery
ARTEMIS	Advanced Research & Technology for EMbedded Intelligence and Systems



# Acknowledgements

*EPOCH*, The European Network of Excellence on ICT Applications to Cultural Heritage was funded by the European Commission under the Community's Sixth Framework Programme, contract no. IST-2002-507382.

*SOFIA*, Smart Objects for Intelligent Applications, is funded through the European Artemis programme under the subprogramme SP3 Smart environments and scalable digital services, within the 7th European Framework Programme, contract no. 100017.

I'd like to thank my colleagues, my supervisor and all of those people who worked with me during this thesis, particularly ARCES and VTT (Oulu) staff who made this work possible.













