Spatial Interaction within Ambient Environments

Ioannis Prokopiou

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Informatics

at
University of Crete, Greece
and
Université Paris-Sud 11, France

Thesis Advisors: Yacine Bellik,
Frédéric Vernier

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir®

Master 2 Interaction Specialty
Université Paris-Sud 11

This is to certify that **Giannis PROKOPIOU** conducted an internship in the LIMSI lab under the supervision of **Yacine BELLIK and Frédéric Vernier**, in partial fulfillment for the degree of Master of Science.

The thesis is entitled:

Spatial Interaction within ambient environments

The thesis has been defended in front of and approved by the following jury on September 19, 2013:

Anastasia Bezerianos

Patrick Bourdot

Michel Beaudouin-Lafon

Caroline Appert

September 19, 2013

Abstract

In ambient environments, the goal is to assist users in their everyday tasks by using real-time location sensors in their physical space. For this new type of environments, the Human-Computer Interaction requires the definition of new interaction methods. By using location sensors, the physical space itself can become a means of interaction. In this internship, we performed studies on how the interactions between the user and the ambient environment can be carried out, using the physical space. We identify the ambient object characteristics that are related to the physical space. We distinguish those that may be relevant to be used as an alphabet in a language for spatial interaction within ambient environments and then we propose such a language. Finally, we develop an application to test the proposed language within the IRoom (Intelligent Room), which is equipped with the Ubisense localization system, in LIMSI lab.

Περίληψη

Στα περιβάλλοντα διάχυτης νοημοσύνης, σκοπός είναι να παραχθεί βοήθεια στο χρήστη για την διεκπεραίωση των καθημερινών εργασιών του. Αυτό είναι κάτι που μπορεί να επιτευχθεί χρησιμοποιώντας συστήματα εντοπισμού θέσης σε πραγματικό χρόνο. Για αυτά τα νέου τύπου περιβάλλοντα, απαιτείται ο ορισμός νέων μεθόδων επικοινωνίας. Με τη χρήση συστημάτων εντοπισμού θέσης, ο φυσικός χώρος μπορεί να γίνει ένα μέσο επικοινωνίας. Σε αυτή την εργασία, πραγματοποιήσαμε έρευνα για το πώς μπορεί να επιτευχθεί επικοινωνία ανθρώπου-υπολογιστή σε περιβάλλοντα διάχυτης νοημοσύνης, χρησιμοποιώντας το φυσικό χώρο. Αρχικά, αναγνωρίζουμε τα χαρακτηριστικά ενός αντικειμένου τα οποία σχετίζονται με το φυσικό χώρο. Έπειτα, διακρίνουμε εκείνα τα χαρακτηριστικά τα οποία μπορούν να χρησιμοποιηθούν ως αλφάβητο μιας γλώσσας για τη περιγραφή χωρικής αλληλεπίδρασης σε περιβάλλοντα διάχυτης νοημοσύνης. Στη συνέχεια, προτείνουμε μια τέτοια γλώσσα. Τέλος, αναπτύσσουμε μια εφαρμογή με τη βοήθεια της οποίας γίνεται χρήση και επαλήθευση της προτεινόμενης γλώσσας εντός του IRoom (Έξυπνο δωμάτιο), το οποίο είναι εξοπλισμένο με το σύστημα εντοπισμού της Ubisense και βρίσκεται στο εργαστήριο LIMSI.

Acknowledgements

First, I would like to thank my supervisors, Yacine Bellik and Frédéric Vernier, for their invaluable guidance, advice and support throughout this project. What is more, I would like to express my gratitude to my family and friends for their unfailing understanding and support.

Contents

List of Figures

List of Tables

1 Introduction

In modern life, the usage of digital multimedia devices has increased. Over the last decade, multimedia devices have changed our living space and life style. Devices like displays, audio systems, etc. are becoming "smarter" by embedding mini computers, making them more featured. Thus, these devices have become more manageable and interactive over time. This interactive environment is regularly described as ubiquitous or smart. Before 22 years, Mark Weiser [1], used the term ubiquitous computing as a description on this field of Information Technology. The idea behind this is that modern information technologies tend to be merged into our surroundings. As stated by Mark Weiser "*for ubiquitous computing, one of the ultimate goals is to design technology so pervasive that it disappears into the surrounding*". This point of view on ubiquitous computing brought new research approaches, like tangible user interfaces [2], which focus on using physical objects as input method on several application interfaces. More recently, researchers aim to design ambient environments featured with multiple sensor technologies, such as interactive media arts [3, 4], interactive workspaces [5, 6].

The way tracking technologies are progressing, location information are getting more interesting for scientists and the industry as well. Using that information, multimedia environments can be enriched with real time data and automatically become more entertaining and useful. Since tracking technologies are becoming a standard, interaction designers have to add location information to the mix when designing smart environments. This will lead to extended design possibilities and will bring changes on how users perceive the environment.

In the past, researches on location based services aiming at information sharing between users through a computer or a handheld device were conducted. Nowadays, using recent tracking technologies, it is possible to track a user's motion and the location of an object with great accuracy. This leads to a new type of ambient environments where spatial interaction between users and objects in the same place occur.

In this thesis we identify the characteristics of the objects that are usable for spatial interaction. Afterwards, using those characteristics as our alphabet, we propose a language one can use to describe such an interaction within ambient environments, while specification examples are also included. Moreover, real-time location systems and more specifically the Ubisense localization system (which is the one that the IRoom is equipped with), are discoursed. Finally, a chapter is dedicated to the compiler application we developed and its features.

2 State of the Art

2.1 Ambient Intelligence

Interaction between human beings and digital information technology is currently limited. The input method of common user interfaces nowadays make use of devices such as keyboards, mice and visual display units, whereas ambient space stays unutilized. By using computing devices, such as sensors, we capture information that exist and stay unutilized at the physical user's space. Such information could possibly be the location of objects, the shape, movement, speed, orientation etc. Once this information become available to the system, new types of interfaces can be used, which will result to a simplified and more accurate human-computer interaction. This is the objective of AmI.

2.1.1 AmI requiremens

In Ambient Intelligence, two kinds of adaptation can be distinguished [7]: adaptivity and adaptability. In the first one, the system's behavior will be automatically adapted while in the second, the system's behavior can be manually adapted by the user according to his/her preferences.

AmI specific requirements	
Decentralization of the interface	Core application and User Interface should be completely separated. UI should become available across the network
The user does not need to know how devices are interconnected	User could move freely from one computer to another believing that they are part of the same system
Availability of interaction devices cannot be forecast during design	The interface should be adapted to the available devices that exist in the current environment

2.1.2 Ubiquitous computing

The use of computers everywhere can be a definition of what ubiquitous computing is; computers that are embedded in the environment in a non-intrusive way and stay invisible to the user. Each computer is configured to perform specific tasks with little -or even without- human intervention.

According to [8], ubiquitous computing is characterized by two main attributes:

- Ubiquity: Computation access is everywhere. User is not limited interacting through a single workstation. For example, in one's house there would be many Skype-capable devices at several locations placed and wirelessly connected, which would give the user the freedom to move from one place to another without having to initiate a new video call from the room's local device. It would be automatically transferred to the nearest device.

- Transparency: This technology is invisible and non-intrusive. For example, having a sensor attached to a door to check if the door is opened or closed.

2.1.3 Context-awareness

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 applications themselves [9].
The concept of context has a very broad view, in which anything can be considered as a context. A person may have his physical context (his location, time etc.), his social context (social relations) and his activity context (tasks he performs in his everyday life, such as listening to music or browsing the internet, etc.). In context-aware computing, a user has to be able to accomplish a task potentially via cooperating and collaborate with other users using multiple devices and networks during his movement in the environment. Thus, many smart applications would automatically adapt to the user's intention. As an example, your mobile phone which is connected with your Facebook profile page, notices that your favorite TV program is about to be on (information taken by your profile, your likes etc.), but it also notices -by using its GPS- that you are not at home. It automatically sends a request to your PVR at home to record the program. A smart application like the aforementioned one needs to gather context information from various different sources in order to be adapted to the preferences of the user, without the explicit user's consent. Context-aware smart applications need the context information to be exposed along with internal data and device functionalities. It is a requirement that has to be met because of the variety of devices that are used and the need for interaction with each other within the context.

2.2 Interaction themes

2.2.1 Tangible interaction

Tangible interaction is about physical interaction with physical (tangible) objects. Most of the times these objects are represented on an interface output medium, while at the same time they constitute the main means of interaction. Input is performed by physical (tangible) manipulation, while the result is represented on virtual space. With tangible interaction, every movement results to a change in physical space, as well as in virtual space. The virtual structure is defined by the software, while the physical structure is represented in physical space.

The main concepts for this type of interaction are:

- Haptic direct manipulation. Whether the user is able to grab and feel the active elements of the object.
- Lightweight interaction. Whether the user receives instant feedback when making small moves during interaction.
- Isomorph effects. Whether the system provides accurate representation that clearly shows the relation between the user's actions and the effects that they cause.

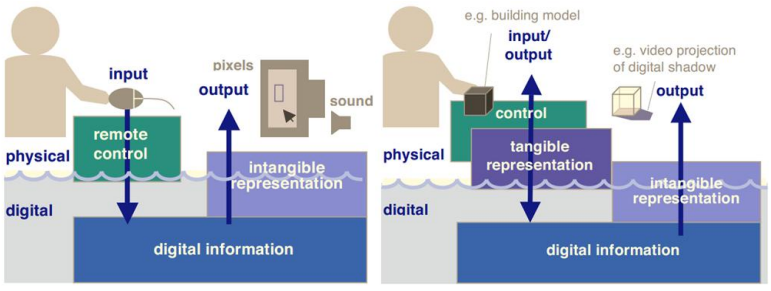

Figure 2-1: *Common GUI on the left, Tangible User Interface on the right*

2.2.2 Spatial interaction

Spatial interaction is the representation and simulation of flows of activity between locations in geographical space. Locations are usually represented as discrete points in space [13]. As we, humans, are spatial beings who live in space, our body is the central reference point for perception. We move in space and we encounter objects and people in it. We also move objects, configuring space regarding our needs. We perceive spatial qualities relatively to our own body. Spatial interaction also allows the movement of the entire body, making it possible to implicitly observe the space we are moving in.

Real space properties	
Inhabited space	By inhabiting space, we appropriate it, interpret it and give it meaning
Configurable materials	It is possible to configure our space by moving anything appropriately
Non-fragmented visibility	One is able to see someone pointing while being able to follow the pointing as the two points of interests are seamlessly connected
Full body interaction	Movement of the entire body is possible
Performative action	Able to communicate through body movement while doing something else

Spatial interaction is related to tangible interaction, as the latter is embedded in real space, meaning that interaction occurs by movement in space. In contrast to tangible interaction, spatial interaction within interactive spaces is not limited by just manipulating objects, but is also likely to be dependent on moving one's body.

2.3 Related work

At this part, previous research that is relevant for this work is presented. Each of them uses one or more properties from the notion of spatial interaction.

2.3.1 Location based

The following implementations use the idea of location tracking.

Proximo[17] is a location-aware mobile recommendation system. It provides guidance to users through tour buildings using Bluetooth technology as a real-time location system. The user's mobile device builds a context by keeping track of the user's location, updating the system with useful information. To pinpoint a user's position, low-cost beacons are sent over the area which the mobile device is able to sniff. Proximo has room-level accuracy which means that it cannot be used to precisely determine the exact user position in a given room.

During execution, the application continually monitors the user location and shows the area that the user is currently in. Experiments were conducted under a gallery context, where the system guided the users through the gallery by giving them painting recommendations.

Hello.Wall[18] is a large wall-sized ambient display which has built-in sensors and LEDs and is designed to blend in with the environment. The goal was to improve people's awareness of their environment by modifying the LED light patterns depending on who was walking by or standing. Based on the light pattern type, different type of information is acquired. Patterns could be public or private, readable only by specific people.

Hello.Wall was designed to form an aesthetic impact that would help in creating a mood in the space which is installed and also influence social meetings. It supports three different zones of interaction, which is user distance-dependent; (i) interaction zone, which is at the smallest distance, (ii) notification zone, which is when the user is not close enough and (iii) ambient zone, which is even further away. Different interaction patterns are created for each zone. At first, people are situated at the ambient zone. At the time they move closer to the system, they step into the notification zone and they are notified through their cell phones. When they finally approach the system, they enter the interaction zone where, they are able to interact directly with the patterns on the display.

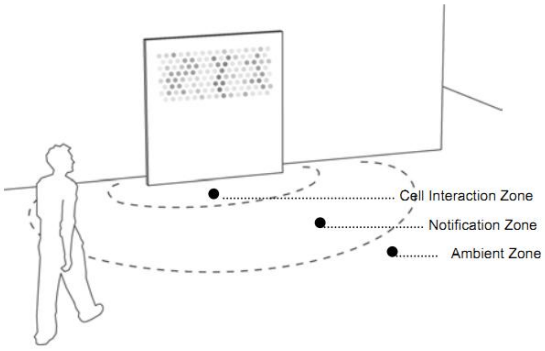

Figure 2-2: *Three zones of interaction of Hello.Wall*

Follow-Me [19] is a system that allows a user to move around a place where the interface of the application "follows" him to the nearest appropriate input or output device. The system uses cameras which are installed into a room. Then the user selects the cameras that are conveniently going to track his position. Afterwards, a video stream is created, which always shows the view of the nearest camera keeping track of the user's position, as he moves between different desks, displays etc.

Different applications give a different meaning to the word "nearest". Using the follow-me system entails each desktop having a space designated around it which, for example, the attached keyboard is in range. To support phone calls the system has to take into account the obstacles in space posed by other objects in the room. For camera selection, a good view of the user is required.

For this project, measuring only the distance is not enough to cover the requirements. For instance, the fact that a telephone device appears to be placed very close to a user does not grant that the user is able to access the device, as there could be a wall between them. Thus, to handle nearness only based by proximity data is not a good idea in this case.

The **CURB**[20] app started as an app for retrieving building information but it switched to a check-in application for mobile phones where users could check-in to rooms they visit frequently inside a building. Every room has an energy score, which is based on averages depending on the frequency that the user checks into. This score is calculated on a daily basis and its value is related to the amount of energy used, compared to an efficiency goal extracted from the previous year's energy consumption. The score is presented as a percentage grade. In case of better room efficiency, the green color is used as an indication for the good score of the room whilst the red is used to indicate a bad score. For the occupants who are contributing to the system, CURB provides badges for check-in purposes.

Figure 2-3: *The CURB app*

Active Badge[21] is one of the first location-aware applications. It is a tracking electronic device that users have to wear. Infrared signals are transmitted to sensors which are installed around the building. Thus, location-based services can be offered, since the active badge provides location information to the system. The first scenario that was used to experiment with the system was to have phone calls automatically routed to the nearest telephone device which is closest to the user.

2.3.2 Proximity based

The following implementations are proximity dependent.

MirrorSpace [22] is a multi-user interactive communicating system which is proximity dependent. While most of the existing systems create a shared space which is single, depending on the distance between the participants, MirrorSpace creates a continuum of space which allows interpersonal relationships to be interpreted.

An optical sensor, a micro controller and a PC are needed to determine the presence of users and blur the environment on the live feed. The blur effect is based on the distance of the participants. The moment a user joins the space, a camera which is installed in the center of the screen captures the space and projects it to the screen where the user is able to see himself and other participants who are also using the system. Depending on the distance between the user and the screen, the resulting specific user image is accordingly distorted in the live feed. This has an immediate effect in the way participants communicate with each other.

Figure 2-4: *MirrorSpace installation at Mains d'Oeuvres (Paris, May 2003)*

Pêle-Mêle [23] is a multiparty video communication system which is designed for group interaction. The components of this system consist of a screen which is equipped with a video camera connected to a small computer. The screen shows other places which are also equipped with the system, with a detailed view of other users who are currently using the computer. This way the system informs the user about the presence of other users.

The key point of the system is the three-level engagement that it introduces and the way that transitions between engagement levels take place. It constantly monitors the activity and classifies it accordingly. These three levels are: (i) away, showing video clips with activity that happened in the past and a view of the last image transmitted, (ii) available, showing video clips with activity that happened in the past and a delayed live stream, (iii) engaged, showing video clips with activity that happened in the past and a live stream which is also recorded for later use. Pêle-Mêle uses OpenCV's face detector to locate people and determine their distance.

Figure 2-5: *Pêle-Mêle, two examples of time composed pictures*

Pirates! [24] Is a multiplayer computer game which runs on handheld computers that are networked through wireless LAN. The virtual game environment is maintained and controlled by a local server which also keeps track of the players' progress over time. The handheld consoles are equipped with proximity sensors which are needed to retrieve the location of the player in the physical space where the game takes place.

In this game, game events, such as engaging a combat with other players or exploring an island, are depending on the player's location in the physical space. Short range RF sensors are embedded in several locations in the physical game area which correspond to islands in the virtual game area. The RF transceivers that the handhelds are equipped with are used to determine the players' position in the physical world or relative to each other.

Proximal Interactions [25] is another example that is triggered under specific spatial conditions. In this paper, the authors suggested a new way to connect two devices wirelessly using the standard wireless LAN. Currently one has to configure the system with appropriate IP addresses in order to setup a connection. To maintain network settings for later use is not the most convenient thing. Also, inputting that kind of information in handheld devices can be slow. In this new way, a near field channel is used which allows the exchange of information that is needed to configure the connection. For this to happen, the user has to point to a target device for infrared-equipped devices or to bring the two devices close enough, in case of RFID technology is used. After exchanging configuration info, the wireless LAN connection is automatically established and from then on, data is transferred through the standard WLAN channel.

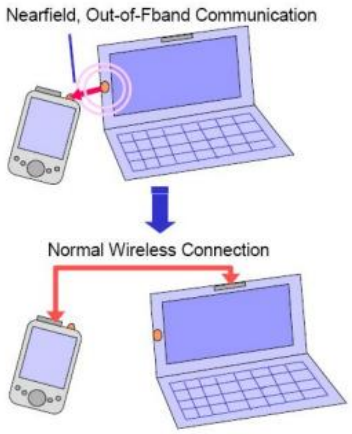

Figure 2-6: *Proximal Interaction Model*

ConnecTable [26] displays are displays that are able to act as one large display when they come really close to each other. That way, individual workspaces can be shared among users. A typical interaction that is supported using these displays is the drag-n-drop feature where a user can drag a digital object from his display and his workspace onto another connecTable. Moreover, by using a simple gesture you can get simultaneous views of a specific object which can then be shuffled to other displays as well. When the devices are disconnected, each user works again on their own single workspace.

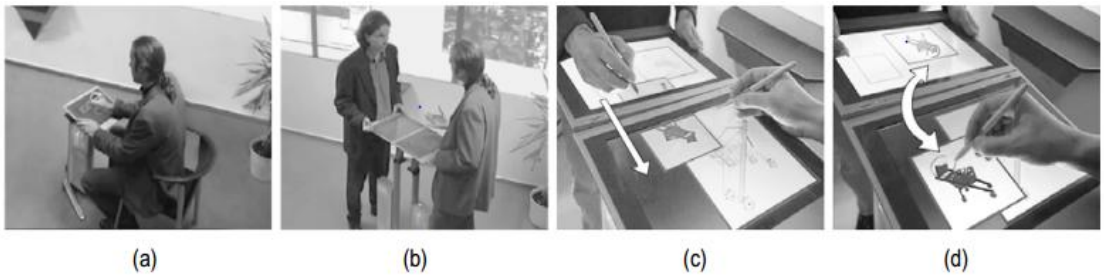

Figure 2-7: *ConnecTables. (a) Single workspace, (b) ConnecTables are placed next to each other, (c) Homogeneous display area enabling to exchange information objects, (d) shared views of the same object*

2.3.3 Multiple-methods based

The following implementations are based on more than one method. The first one depends on location tracking and orientation whilst the second one is reliant on location tracking, orientation factors and proximity.

This paper talks about a system named **Virtual Shelves** [27] which is an interface that allows users to execute commands by spatially accessing predefined locations in the air. Virtual shelves exist at the theta and phi planes in front of the user and different regions cause different kind of interaction (e.g. a button or a shortcut, etc.). Two studies were conducted. In the first, general kinesthetic accuracy was measured by getting the user to try to locate a region on the virtual shelves. In the other study, participants were given cell phones and were asked to operate them using orientation and reaching certain spatial locations. The results were encouraging and showed that users are able to perform such spatial operations accurately enough and are also able to execute tasks even faster than using the standard way of interaction for cell phones.

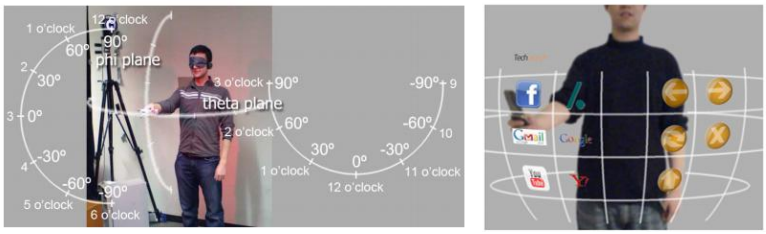

Figure 2-8: *Virtual Shelves. Left: A user orienting a phone to trigger shortcuts. Right: Web browser specific shelves available to the user when browsing the internet.*

iCam [28] is a handheld device capable of determining its own position, orientation and the location of objects in its environment, using given position sensors. iCam is equipped with a camera, a touch screen, a location tag and orientation sensors. Additionally, it uses a laser pointer which is used against a physical object in order to determine its relative position. One has two ways on how to use the system. At first, you can define the physical space and put content in it. When you trace parts like walls and doors, you allow the system to produce a geometric model of the space. Subsequently, you can add digital content. Then, one can use the camera to capture the space and see the augmented digital content on top of the video stream. Applications of this type, that mix computer generated information with the real world, are called Augmented Reality.

A simple usage scenario for the iCam is a mobile tour guide. POIs (Points of Interest) are captured and then digital content is added. Tourists then can use the iCam to navigate through space and retrieve information for any interesting point.


Figure 2-9: *iCam. The user scans the physical environment*

2.3.4 Other-methods based

The following implementations are based on methods that have not been mentioned above. The first one depends on sensing footsteps whilst the second one is reliant on gestures.

The Sensoric Garden's **Clavier** [14] is a walkway equipped with light sensors that are triggered when someone walks by. The system was installed in a park during a festival in Bremen (Germany). When users were walking along this path, they caused a colored spotlight reaction in the place they had their feet. Also, sounds from drums and beats were played which encouraged people to jump from light to light and to dance to their own music. Some also tried walking back and forth and used umbrellas and other objects to trigger many sensors simultaneously.

The system offered users ways to implicitly and explicitly interact. As users were passing by, interaction was taking place with the system in an implicit manner. Moreover, as a single user could only produce simple sounds, several users needed to cooperate in order a richer audio mix to be produced. Thus, group creativity was encouraged. Apart from the fact that Clavier highlights the spatial interaction theme, these effects make it a good paradigm for the embodied constraints given by the sheer size of an interaction area. Furthermore, the need for full-body interaction makes this interaction publicly available where actions are visible.


Figure 2-10: *Visitors dancing on the Clavier*

In a paper about "**imaginary interfaces**" [29], researchers are trying to create interfaces which will not require the need for any visual feedback. By using screen-less devices and empty hands, users are able to achieve spatial interaction. The aim of this experiment is to discover whether spatial interaction is possible when using imaginary interfaces, examining the possibility to create a system that relies on a user's recent memory without providing feedback that the user can acknowledge. Users form an L-shaped cross using their non-dominant hand and keep this cross as a reference for their imaginary space, while with the other hand they point and draw whatever they want. The device which they use, constantly monitors hand movements and discovers what command the user wants to execute. It also uses the L-shaped cross as a point of reference. The article states that it was difficult for users to remember the different gestures in order to accomplish things, because of the missing visual cues to help their memory.

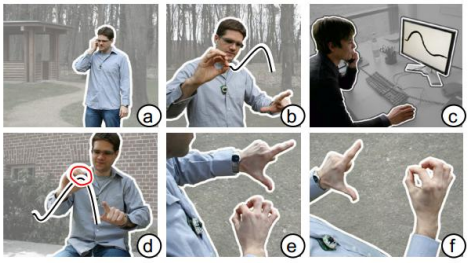

Figure 2-11: *Using Imaginary Interfaces*

Research work	Location	Proximity	Orientation	Other
Proximo	X			
Hello.Wall	X			
Follow-Me	X			
The CURB app	X			
Active Badge	X			
MirrorSpace		X		
Pêle-Mêle		X		
Pirates!		X		
Proximal Interactions		X		
ConnecTable		X		
Virtual Shelves	X		X	
iCam	X	X	X	
Sensoric garden Clavier				X
Imaginary Interfaces				X

Table 2-1: *Related work*

2.3.5 Conclusion

We reviewed several research implementations. Each of them uses one or more properties from the notion of spatial interaction. Five out of fourteen use only location-based methods, while another five use only proximity-based methods. One of them uses a combination of the aforementioned methods along with orientation-based methods, while another combines methods relying on location and orientation. Finally, the remaining ones use other methods, such as gestures or sensing footsteps.

What we noticed is that a system is created in every research project that uses some kind of spatial interaction. However, no existent language can fully describe the full range of interaction that takes place in those systems. Hence, there is currently no framework a developer can use in order to readily alter the functionality of such a system.

3 Spatial language definition

In this chapter we make an attempt to define our suggested spatial language through the two main subsections below. In the first one we refer to the spatial language properties, while in the second one we analyze the specifications of this language.

3.1 Spatial language properties

In this subsection, we assess the properties of the spatial interaction language that we suggest. In such a language, physical objects and their attributes play a crucial role for its structure. In other words, the fact that the state of an object can change from being still to be moving can be either meaningful or completely indifferent to us. This language should give us the possibility to describe any potential spatial interaction action in a given physical space. Therefore, apart from acknowledging the given space, it should be in a position to: (i) identify the ambient object characteristics that are related to it, (ii) make it possible for the developer to define various constraints, (iii) make it possible for the developer to register various events that will be raised under specific circumstances.

3.1.1 Object characteristics

Any given object comes with a number of specific characteristics. These ambient object characteristics are related to its sheer nature. From this group of attributes we merely make use of those that are relevant to us in order to efficiently define an object as an entity.

Characteristics	Explication	Value
Position	The coordinates of an object in a physical space	(x, y, z) coordinates
Size	The relative extend of an object regarding its length, width and height	(length, width, height) cm
Orientation	How the object's axes are placed	(x, y, z) degrees
Speed	The rapidity of movement of an object	cm/s
Acceleration	The rate of change of speed per unit of time	cm/s^2
Weight	(not actually spatial) An object's relative mass	g

Table 3-1: *Object characteristic*

3.1.2 Constraints

Generally, a constraint is a statement of restriction, modifying a requirement or set of requirements by limiting the range of acceptable solutions. A constraint can be used as an event attribute or as an object attribute that an event entails.

Constraints	Explication	Range	Object attributes
Distance	The amount of space between the given object and another object	$\min \leqq \text{distance\_value} \leqq \max$	X
Threshold	The time window in which an event should occur	$\min \leqq \text{latency\_value} \leqq \max$	X
Absolute orientation	The min/max permitted degrees of an object's absolute orientation	$\min(x, y, z) \leqq \text{current}(x, y, z) \leqq \max(x, y, z)$	✓
Relative orientation	The min/max permitted degrees of an object's orientation relative to another object	$\min(x, y, z) \leqq \text{current}(x, y, z) \leqq \max(x, y, z)$	X
Relative location	The area which the object is permitted to be coming from	$\min(x, y, z) \leqq \text{current}(x, y, z) \leqq \max(x, y, z)$	X
Speed	The min/max permitted velocity of an object	$\min \leqq \text{speed\_value} \leqq \max$	✓
Acceleration	The min/max permitted change of speed of an object	$\min \leqq \text{acceleration\_value} \leqq \max$	✓

Table 3-2: *Constraints*

3.1.3 Events

The idea behind the event-driven approach is that specific sections of code are executed based on what spatial interactions occur in the physical space. An event is a set of requirements that once satisfied can cause a specific function to be called. This function is associated with the event, meaning that, when the event is raised, the function is called to handle it.

Our suggest language supports two kinds of events: (i) Base (elementary) events, which are caused by alterations in certain object attributes, (ii) Composed events, which are caused by the concurrent occurrence of two or more elementary events.

Base (elementary) events	Explication	Event attributes
Enter	A registration of an Enter event will cause this event to be raised when the source defined object comes close to the destination defined object	Distance, relative orientation, relative location
Leave	A registration of a Leave event will cause this event to be raised when the source defined object leaves the destination defined object	Distance, relative orientation, relative location
Meet	A registration of a Meet event will cause this event to be raised when one object comes close to another	Distance, relative orientation, relative location
Separate	A registration of a Separate event will cause this event to be raised when one object leaves another	Distance, relative orientation, relative location
Orientate	A registration of an Orientate event will cause this event to be raised when one object changes its orientation	X
Null	A registration of a Null event will cause this event to be raised when one object's constraints are fulfilled even though the event has no constraints itself	X

Table 3-3: *Elementary events*

Composed Events	Event type	Explication	Event attributes
Temporal events	Sequence	A registration of a “sequence type” Temporal event will cause this event to be raised when the entailed events are raised in a specific order	Threshold
	Simultaneous	A registration of a “simultaneous type” Temporal event will cause this event to be raised when the events entailed are raised in an almost concurrent way	Threshold
Logical events	And	A registration of an “and type” Logical event will cause this event to be raised when the entailed events are both raised over time, without a specific order and without both happening at the same time to be mandatory	X
	Or	A registration of an “or type” Logical event will cause this event to be raised when either one or both of the two events occur	X

Table 3-4: *Composed events*

3.2 Specification language definition

In this chapter, we analyze the specifications of the suggested event-driver language. We present basic examples, explaining its functionality and capabilities.

We have chosen the standard XML format to be the format that our source code will be written in. XML stands for e**X**tensive **M**arkup **L**anguage and is an international data standard. It aims to separate presentation, structure and meaning from the actual content. Nowadays, XML is used to represent any kind of data structure. By using the XML format we take advantage of its great simplicity in its syntax and structure. XML files are easy to read, understand and translate into other environments.

XML advantages	
Simplicity	Easy to understand, using tags which you define
Organization	Easy to organize files based on their content
Accessibility	Accessing and updating the content is easy and time-saving
Standardization	XML is an international standard. Everyone is likely to be able to view or alter the content

Table 3-5: *XML advantages*

On top of these, from a developer's perspective, parsing an XML file today is a relatively easy task to perform, due to the fact that there are plenty of XML parsers available to use.

3.2.1 XML syntax

XML syntax rules are very simple and logical. Basic XML rules that we use in our source files consist of:

- Every element must have a closing tag.

  <tag> element </tag>
- XML tags are case sensitive

  <Tag> … </Tag>
- XML Elements must be properly nested

  <Student><Telephone>0683115113</Telephone></Student>
- XML documents must have a root element

  <root>
  <child> … </child>

  <root>
- XML attribute values must be quoted

  <Enter id=”enter1”> … </Enter>
- Comments in XML are declared like this:

  <!-- This is a comment -->

3.2.2 XML structure

In our spatial interaction language, before executing a scenario, one has to define some system-related elements, some environment-related elements and the scenario itself. These definitions are part of the source code that the developer should provide to our tool. To make things simpler, we decided to split the source code to three separate XML files, depending on their content:

Default filename (can change)	Information contained	
system.xml	System related	Location tags definition
environment.xml	Environment related	Space and objects definition
scenario.xml	Scenario related	Scenario definition (actions, constraints and events)

Table 4-1: *System, environment and scenario content*

The structure that these three files should follow is defined in their corresponding XML Schemas. An XML schema [30] describes the type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type.

3.2.2.1 System file

The system file contains system related information. More specifically, in this file the developer has to define the number of tags which are present on the system. He also needs to specify unique IDs.

```
<!-- Available tags in the system
     Attributes: id -->
<Tags>
     <Tag id="tag1"/>
     <Tag id="tag2"/>
     <Tag id="tag3"/>
</Tags>
```

In this example, we have declared the presence of three tags. We also define their unique tag IDs.

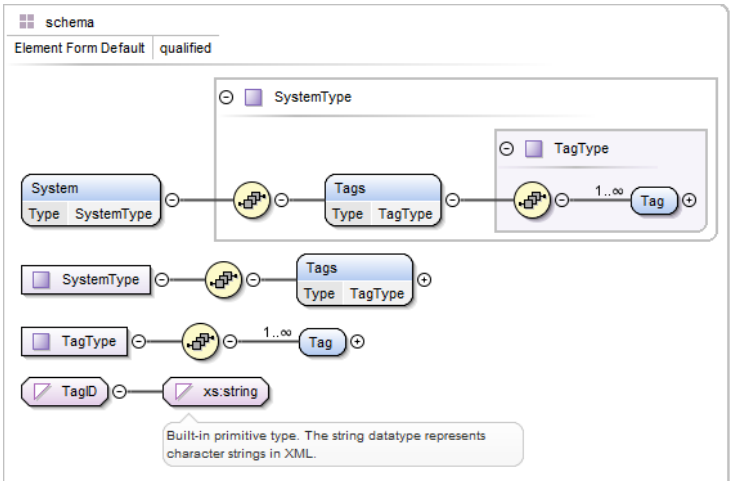

Figure 4-1: *System file XML Schema graphical representation*

3.2.2.2 Environment file

The environment file contains environment related information. More specifically, in this file the developer has to define a space with a size, a name and as always a unique space ID. Also he needs to define the objects which are present and have a meaning to the scenario.

```
<!-- Definition of Space
     Attributes: id -->
  <Space id="space1">
    <!-- Definition of Room -->
    <Name>MySpace</Name>
    <!-- Size of room
       Attributes: values for x,y,z (in cm) -->
    <Size>
      <x value="600"/>
      <y value="600"/>
      <z value="350"/>
    </Size>
  </Space>
```

In the above example, we have defined a space with an ID, a name, and a size.

```
<!-- Definition of Objects -->
 <Objects>
    <!-- Definition of Object
       Attributes: id, tag -->
    <Object id="object1" tag1="tag1">
      <Name>Track 1</Name>
      <!-- Object location in room
      Attributes: values for x,y,z (in cm)-->
      <Location>
        <x value="100"/>
        <y value="100"/>
        <z value="150"/>
      </Location>
      <!-- Size of object
      Attributes: values for x,y,z (in cm)-->
      <Size>
        <x value="30"/>
        <y value="15"/>
        <z value="5"/>
      </Size>
      <!-- Weight of object
      Attributes: weight value (in g)-->
      <Weight value="500"/>
      <!—Default absolute orientation of the object
      Attributes: values for x,y,z (in degrees) -->
      <AbsoluteOrientation>
        <x value="0"/>
        <y value="0"/>
        <z value="0"/>
      </AbsoluteOrientation>
    </Object>
</Objects>
```

Here we define an object. We set its name, location, size, weight, default orientation and the tag which is attached to, if there is any. The weight property is not actually a property that we can sense using a location system. Thus, its use is limited for developer guidance purposes only.

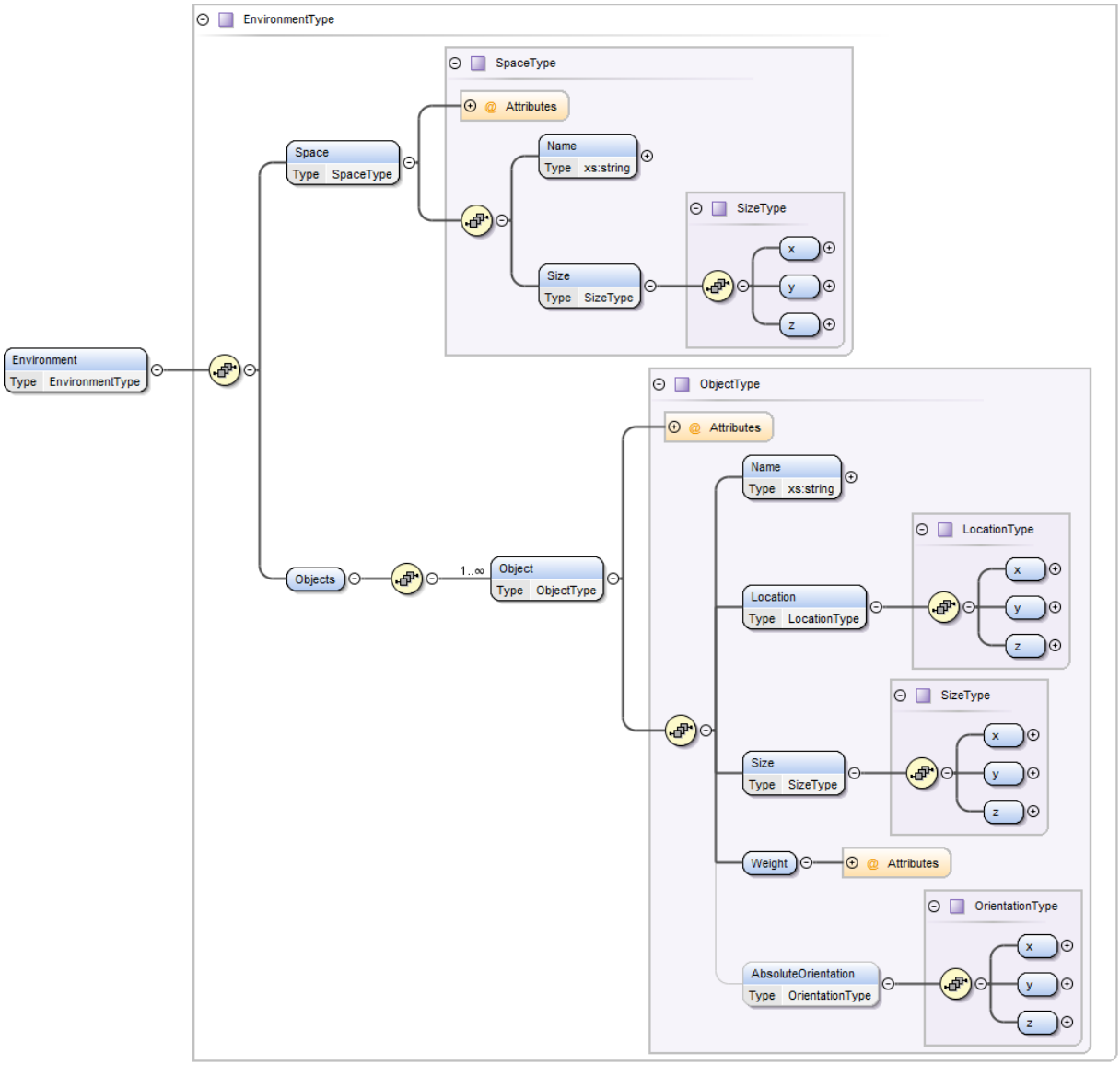

Figure 4-2: *Environment file XML Schema graphical representation*

3.2.2.3 Scenario file

The scenario file contains scenario related information. This means that the developer has to define the actions, constraints, any virtual objects and lastly, the events.

```
<!-- Definition of Actions -->
<Actions>
   <!-- Attributes: id, function to call -->
   <Action id="action1" function="function1"/>
   <Action id="action2" function="function2"/>
   <Action id="action3" function="function3"/>
</Actions>
```

In the above example, we define a set of actions which will trigger the call of their corresponding functions in case an event which is related to that action is raised (Currently, the functions are typical C# methods embedded in the implemented tool). After defining the actions, constraints are the next to be defined in the scenario file. In our case, the developer is free to define as many constraints as he wants for later use with events, adjusting their final requirements. The constraints consist of distances, thresholds, orientations, locations, speeds and accelerations.

Below, we define two different distance constraints (one as the default) with their minimum and maximum distance limits. Distances are used in events, such as the Enter event, to set how much should an objectA come close to an objectB in order to raise the event.

```
<!-- Definition of Distances -->
    <Distances>
      <!-- Attributes: id, min, max (in cm), default will be used if not set -->
      <Distance id="distance1" min="0" max="10" default="true"/>
      <Distance id="distance2" min="15" max="20"/>
    </Distances>
```

The purpose of setting thresholds is to allow the developer to write events which will only be raised within specific time frames. Here, we define a threshold of 2000ms and we set it as default.

```
<!-- Definition of Thresholds -->
    <Thresholds>
      <!-- Attributes: id, interval (in ms), default will be used if not set -->
      <Threshold id="threshold1" interval="2000" default="true"/>
    </Thresholds>
```

Absolute orientation refers to the orientation of the object, while relative orientation refers to the orientation of the object relative to the absolute orientation of another object. Using orientation constraints, one can create several events which will only be raised if the absolute orientation or relative orientation requirements are met. In this example, we define two different orientations with their min/max values which satisfy the rule. If orientation in an axis is not set, it could be any value.

```
<!-- Definition of Orientations (Absolute or Relative) -->
    <Orientations>
      <!-- Attributes: id, min, max for x, y, z axes (in degrees) -->
      <AbsoluteOrientation id="absoluteOrientation1">
        <x min="0" max="180"/>
        <y min="0" max="90"/>
        <z min="0" max="90"/>
      </AbsoluteOrientation>
      <!-- Attributes: id, min, max for x ,y, z axes (in degrees) -->
      <RelativeOrientation id="relativeOrientation1">
        <x min="0" max="60"/>
      </RelativeOrientation>
    </Orientations>
```

Relative location means to locate a place relative to the location of other places. In our case, relative locations are used to support having different events raised, based on different directions, where one object can approach another object (for example, in an Enter event case). In the example below, we define a relative location containing the margins, which we could later use to set as a requirement to an event.

```
<!-- Definition of Locations -->
    <Locations>
      <!-- Attributes: id, min, max for a(x), b(y), c(z), d -->
      <RelativeLocation id="relativeLocation1">
        <x min="0" max="2"/>
        <y min="0" max="3"/>
        <z min="0" max="1"/>
      </RelativeLocation>
    </Locations>
```

By defining speed constraints, we set the minimum and the maximum speed a tagged object can have to satisfy the requirement. At least one of the min/max values is required.

```
<!-- Definition of Speeds -->
    <Speeds>
      <!-- Attributes: id, min, max (in cm/s) -->
      <Speed id="speed1" min="10"/>
      <Speed id="speed2" min="4" max="8"/>
    </Speeds>
```

Respectively, by defining acceleration constraints, we set the minimum and the maximum acceleration a tagged object can have to satisfy the requirement. At least one of the min/max values is required.

```
<!-- Definition of Accelerations -->
    <Accelerations>
      <!-- Attributes: id, min, max (in cm/s²) -->
      <Acceleration id="acceleration1" min="2" max="5"/>
      <Acceleration id="acceleration2" max="8"/>
    </Accelerations>
```

Virtual objects are not real objects. A virtual object is a set of real objects that we want to put in proximity to each other and treat like one object. However, a virtual object cannot have standard size, orientation, etc. Below, we define a virtual object as a set of three objects. In order for this virtual object to exist, the objects contained should not exceed the maximum distance of the distance attribute used.

```
<!-- Definition of VirtualObjects -->
    <VirtualObjects>
      <!-- Attributes: id, distance, default will be used if not set -->
      <VirtualObject id="virtualObject1" distance="distance1">
        <Name>VirtualObject1</Name>
        <Object>Object1</Object>
        <Object>Object2</Object>
        <Object>Object3</Object>
      </VirtualObject>
    </VirtualObjects>
```

After defining actions, constraints and virtual objects, we have to register our events. Generally, in an event-driven language the flow of the program is determined by events (e.g. sensor outputs or user actions). In the scenario file we also associate an action with an event, meaning that when the event occurs, the function that is associated with that action is called to handle it. Starting with elementary events, we define an Enter event which will be raised when the source object (Object1) comes close to the destination object (Object2). A distance constraint is not entailed, thus, the default defined distance will be used.

```
<Enter id="enter1" action="action1">
        <Src>Object1</Src>
        <Dst>Object2</Dst>
</Enter>
```

At this point we define a Leave event which will be raised when Object1 leaves Object2. A distance constraint is not entailed, thus, the default defined distance will be used.

```
<Leave id="leave1" action="action3">
        <Src>Object1</Src>
        <Dst>Object2</Dst>
</Leave>
```

Below we define a Meet event which will be raised when Object1 and Object2 come close to each other. A distance constraint is not entailed, thus, the default defined distance will be used.

```
<Meet id="meet1" action="action2">
        <Object>Object1</Object>
        <Object>Object2</Object>
</Meet>
```

Here we define a Separate event which will be raised when Object1 and Object2 are separated. A distance constraint is not entailed, thus, the default defined distance will be used.

```
<Separate id="separate1" action="action4">
        <Object>Object1</Object>
        <Object>Object2</Object>
</Separate>
```

Below we define an Orientate event which will be raised when the orientation of Object1 changes from the orientation defined in the SrcOrientation element to the one defined at the DstOrientation element.

```
<Orientate id="orientate1" action="action5">
        <Object>Object1</Object>
        <SrcOrientation absoluteOrientation="absoluteOrientation1">Object1</SrcOrientation>
        <DstOrientation absoluteOrientation="absoluteOrientation2">Object1</DstOrientation>
</Orientate>
```

At this point we define a Null event which will be raised when the speed of Object1 matches the speed related constraint.

```
<Null id="null1" action="action6">
         <Object id="speed1">Object1</Object>
</Null>
```

Now we define a “sequence type” Temporal event which will be raised when the events entailed are raised in a specific order within a potential given threshold.

```
<TemporalEvent id="temporalEvent2" type="sequence" threshold="threshold1" action="action8">
        <Event id="enter1"/>
        <Event id="enter2"/>
</TemporalEvent>
```

Now we define a “simultaneous type” Temporal event which will be raised when the events entailed are raised almost concurrently within a given threshold.

```
<TemporalEvent id="temporalEvent3" type="simultaneous" threshold="threshold1" action="action9">
        <Event id="enter1"/>
        <Event id="enter2"/>
</TemporalEvent>
```

Here we define an “and type” Logical event which will be raised when the entailed events are both raised over time, without a specific order and without both happening at the same time to be mandatory.

```
<LogicalEvent id="logicalEvent1" type="and" action="action10">
        <Event id="enter1"/>
        <Event id="enter2"/>
</LogicalEvent>
```

Below we define an “or type” Logical event which will be raised when either one or both of the two events occur.

```
<LogicalEvent id="logicalEvent2" type="or" action="action11">
        <Event id="enter1"/>
        <Event id="enter2"/>
</LogicalEvent>
```

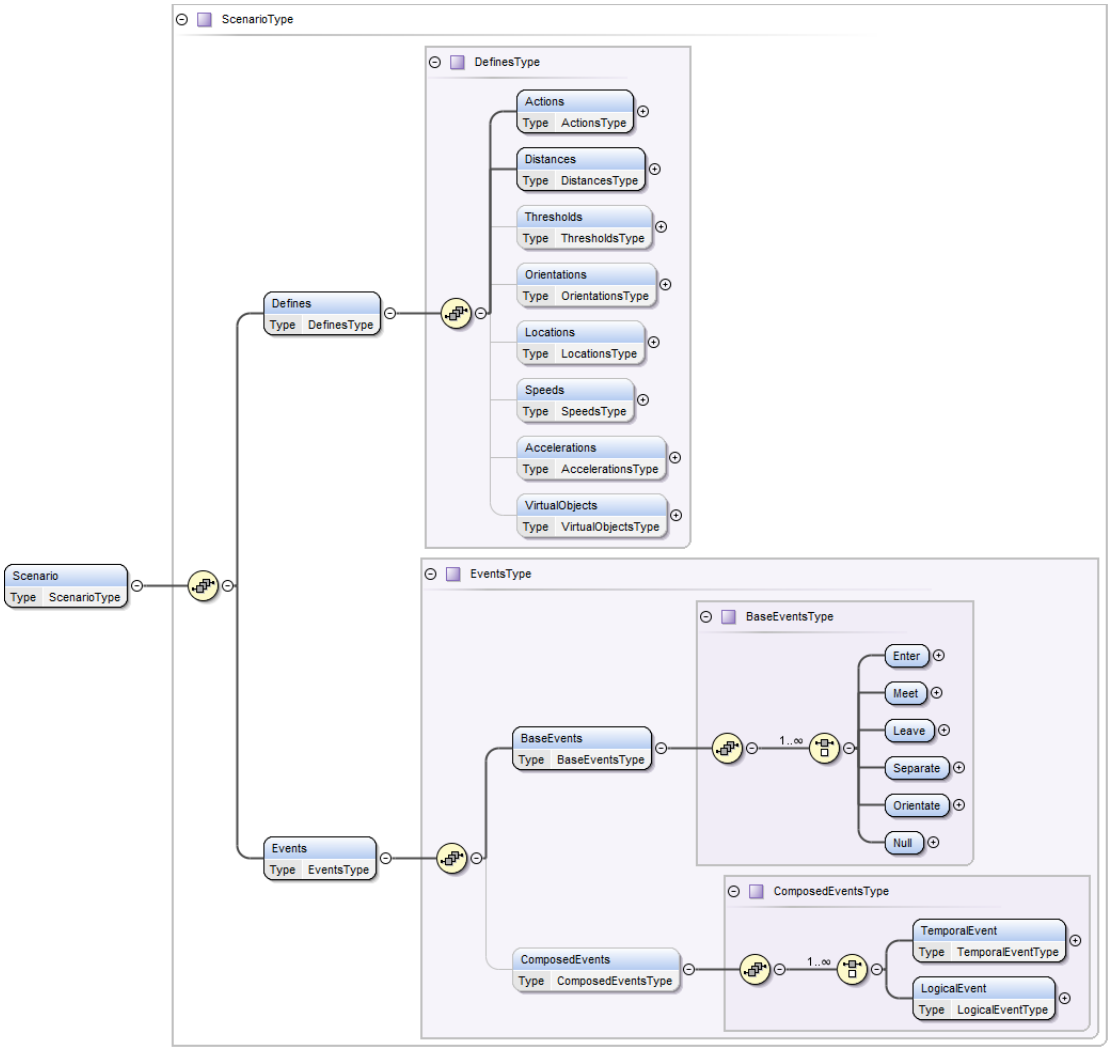

Figure 4-3: *Scenario overview XML Schema graphical representation*

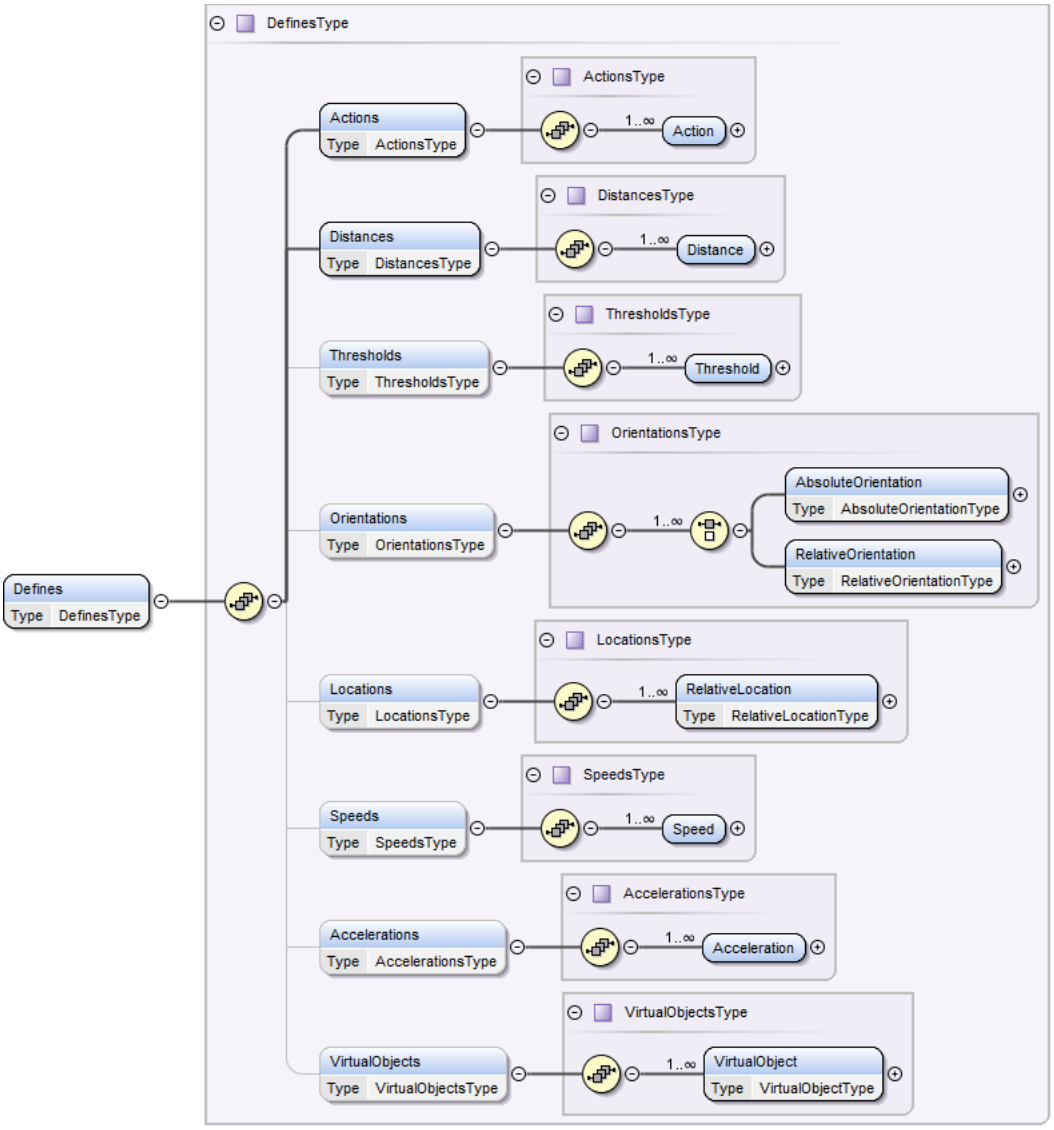

Figure 4-4: *Scenario definition, XML Schema graphical representation*

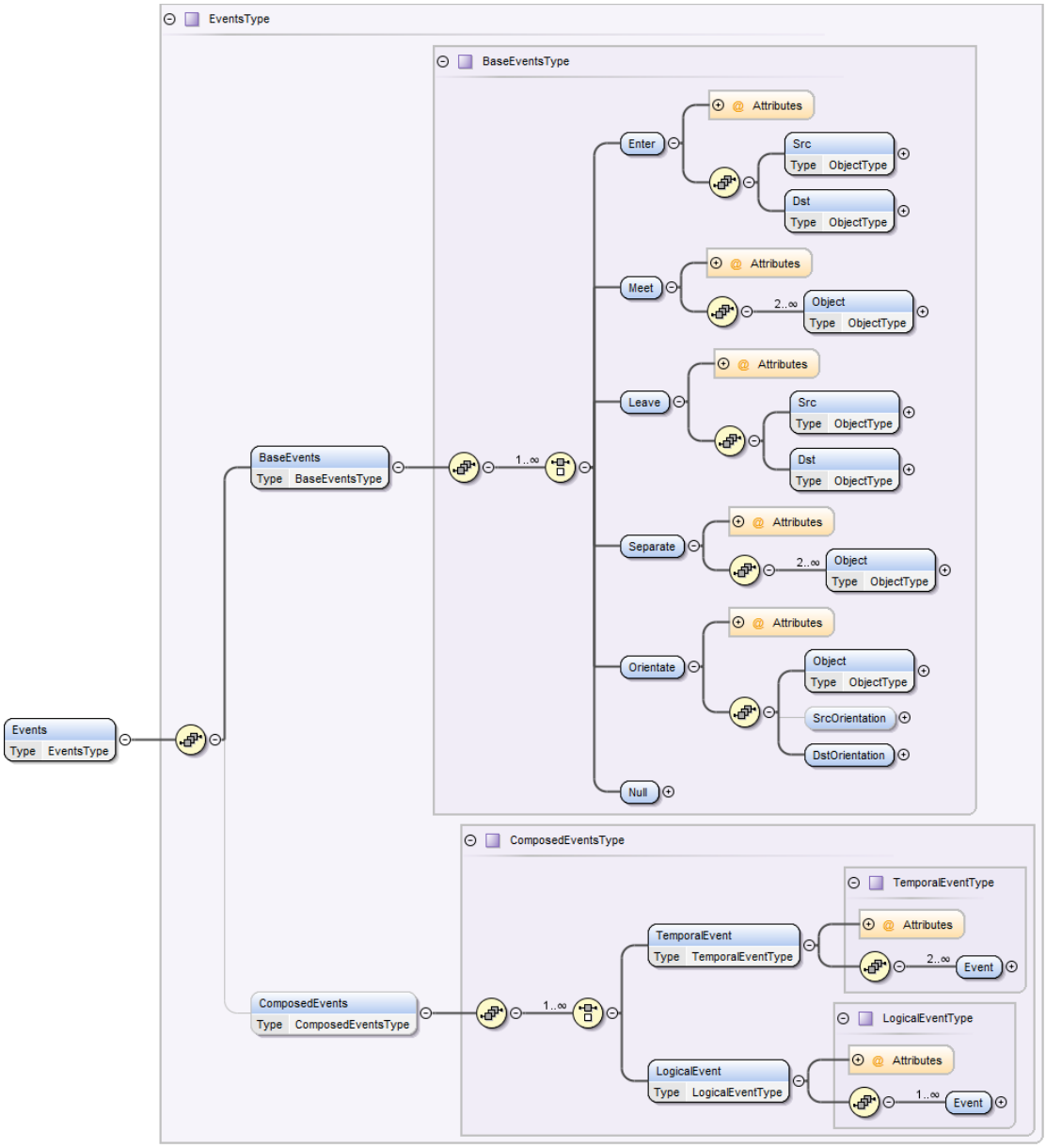

Figure 4-5: *Scenario events XML Schema graphical representation*

4 Real-time location systems

"Real-time locating systems are wireless systems with the ability to locate the position of an item anywhere in a defined space (local/campus, wide area/regional, global) at a point in time that is, or is close to, real time" [31]. Objects can be tracked and identified in real time by the usage of a RTLS.

Using a RTLS, one is able to detect the user's or object's location at a specific time and refresh rate. Technologies like camera vision, GPS, IR (infrared) RFID, WLAN, UWB can be used in a RTLS.[32] In two or three-dimensional space, location systems are able to approximate the location of an object using references related to known space coordinates. Usually, in order the system to calculate these values, parameters like distance and observable angles are used, which can be retrieved through various sources, like the difference between arrival times and field strength. Location systems aim to detect the exact location of users or objects, encountering as minimum errors as possible. Once we get the relative location of an object, which is related to some reference points, we can find its absolute location [33]. Noise, errors in measurement, as well as reference points which are not accurate enough can cause errors in the detection process.

Lately, location technologies are focused on improving the system's detection accuracy and also the process of integration of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation. To take advantage of the existing technologies, one has to estimate several aspects in order to choose the proper technology. Each location system serves a purpose which is associated with its advantages and limitations. Thus, location system requirements contribute in reaching the goal of the RLTS. "Good applications are those that achieve an adequate equilibrium between system requirements, technological advantages and associated costs" [34].

RTLS system requirements	
Accuracy	Accuracy of 30cm or less and a refresh rate of minimum 1Hz is generally considered acceptable by many applications for navigation, location tracking etc.
Cost of installation	Depending on the application and usage, the system should not be very expensive to acquire and install
Cost of operation and ease of use	The System should be simple to maintain, with an acceptable cost
Size and weight	Depending on the objects which will be tagged and the room that the sensors will be placed in, they should have suitable size and weight
Standards and regulations	Regulations to follow based on the country
Interoperability	The system must have an interface with wireless communication technologies and also tolerate other RF (Radio Frequency) signals as well
Range	To be able to detect the longest possible static or mobile object
Interferences	The system should have acceptable performance under multipath signals caused by obstacles

Table 5-1: *RTLS system requirements*

4.1 UWB Systems

UWB systems are short-ranged and typically used indoors. By using UWB, signals can be carried through thin obstacles. UWB is better than common RF in cases where reflections deform the direct path signal, making it difficult to distinguish. If conditions allow it, the UWB system can achieve accuracy of up to 15cm. Accuracy of a UWB system can be affected by the lack of direct line of sight tag signals and the existence of metallic objects in the local area.

UWB RLTS components	
Tags	Small sized, most of the times are attached to objects we want to track
Location sensors	Location sensors can determine tags location by using angles and signal timing. They are also capable of two-way communication
Location engine	The software that carries out the computation process to find the exact location, using techniques like TOA (Time of Arrival) and TDOA (Time Difference of Arrival)
Middleware	The software that connects the disparate applications, allowing them to communicate with each other and exchange data. It is the software that resides among the pure RTLS technology components (tags, sensors, and the location engine) and the business applications [32]
Application	The software that interacts with the RTLS middleware and does the work users are directly interested in [32]

Table 5-2: UWB RTLS components

4.2 Ubisense UWB System

The Ubisense location system uses UWB radio and it consists of the following components:

- Ubisensors, which are in fixed locations spread all over the area to be covered and network via Ethernet. Ubisensors are equipped with a common RF transceiver and UWB receivers.
- Ubitags, which are tags that can be attached to objects or carried by people, Ubitags are equipped with a common RF transceiver and a UWB transmitter.

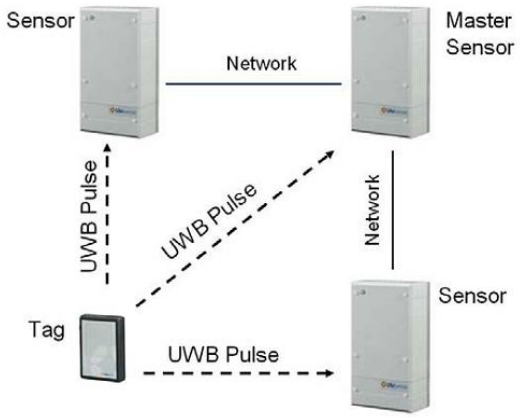

Figure 5-1: *Ubisense hardware components*

The Ubisensors are usually four to seven, organized in such way where every Ubisensor is responsible for a given area. One Ubisensor is set as the master sensor and it organizes a TDMA network using common RF. Every Ubitag is active based on schedule of slots that is allocated. This schedule can change dynamically from the software, in case we want to capture fast moving objects (faster refresh rate) or very slow moves (lower refresh rate). Two way communication is supported by the common RF channel between Ubitags and Ubisensors. Each Ubitag when is active, transmits its identity through common RF along with a UWB pulse sequence. The Ubisensors use this pulse to detect the location of the Ubitag. It is possible to get a Ubitag location by using only two Ubisensors. Ubisense location engine uses techniques like AOA (Angle of Arrival) and TDOA (Time Difference of Arrival) to find the location of the transmitting Ubitag.

The Ubisense platform is based on a client-server model, which allows communication from the server to clients and vice versa [Fig. 5-2].

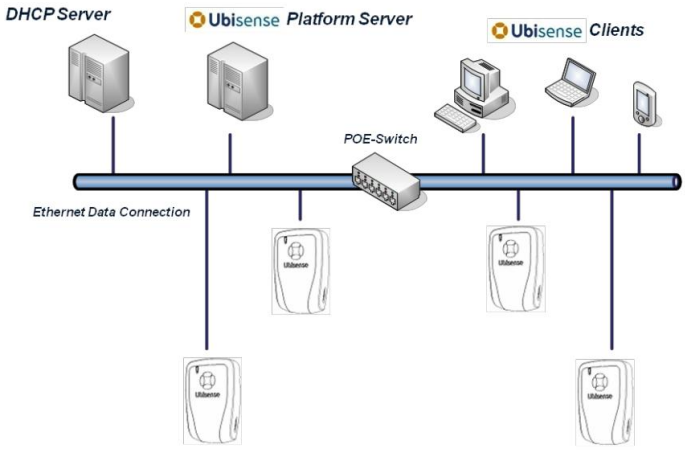

Figure 5-2: *Ubisense platform*

In the Ubisense software suite, location engine software monitors tag locations and is also capable of managing and controlling the Ubisensors, cells, filtering etc. In the following figure [Fig. 5-3], the Ubitags are represented by green dots.

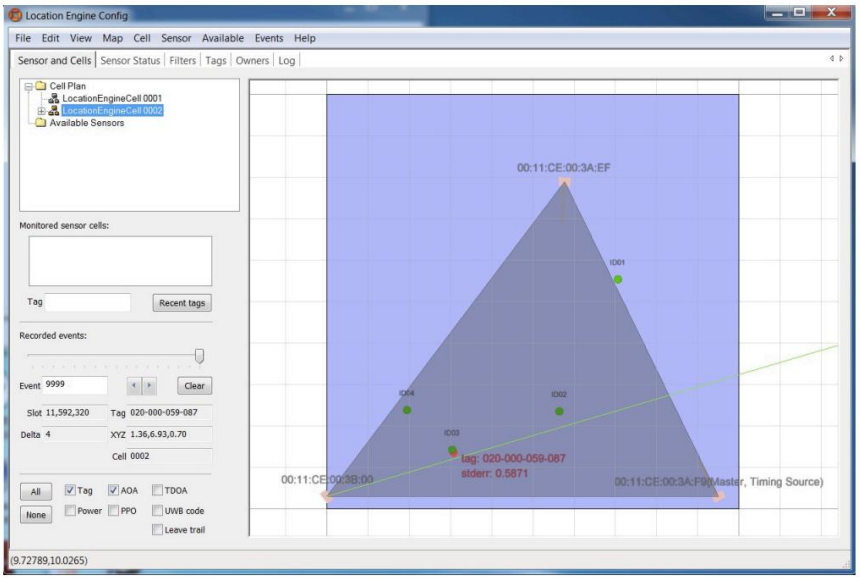

Figure 5-3: *Ubisense location engine*

Site manager is also part of the Ubisense software suite. Using this application you can edit object names, types, their representation etc. [Fig. 5-4]

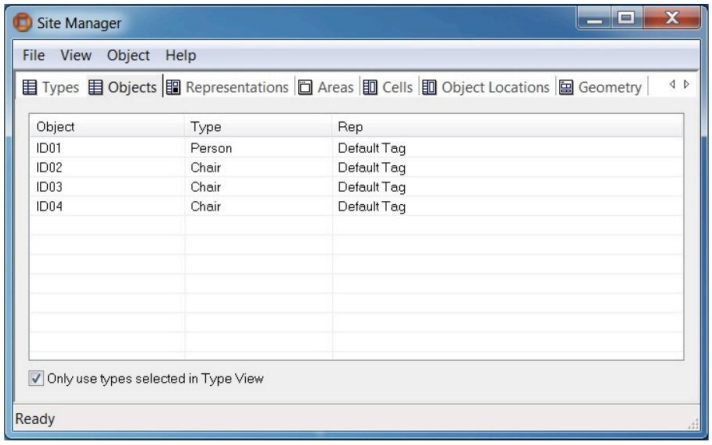

Figure 5-4: *Ubisense site manager*

The Ubisense simulator application [Fig 5-5] can simulate the movements of the tags without the need to install any sensors. It gives the possibility to create scripts based on real life scenarios and execute them without having the need to move any tag in the real environment. This tool was used to thoroughly inspect the operation of the compiler application which was developed in this project.

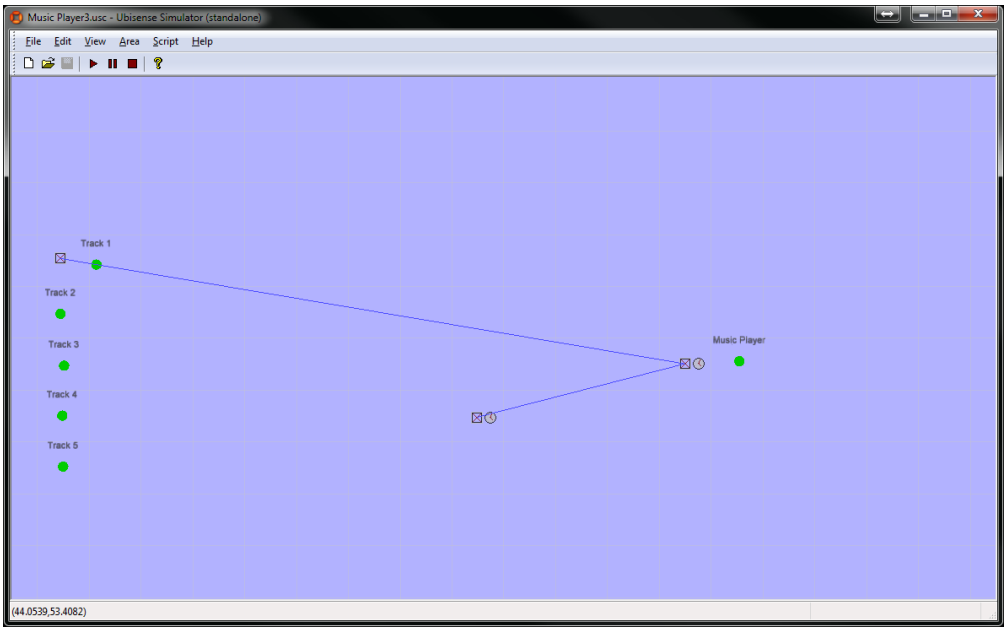

Figure 5-5: *Ubisense simulator*

Ubisense software suite is fully integrated within Microsoft .NET Framework. A .NET API is provided by Ubisense which allows system functions to be controlled. The API includes four DLL files which can be loaded (referenced) in MS Visual Studio .NET projects using .NET programming languages, like C# or VB.NET.

5 Compiler implementation

5.1 Architecture

This compiler application is written in C# language (.NET Framework), using the Visual Studio 2012 IDE and referencing the Ubisense Dynamic Link Library (DLL) files. It consists of some basic primary components, along with some components that have a secondary role for its operation. Components are described in detail in this chapter.

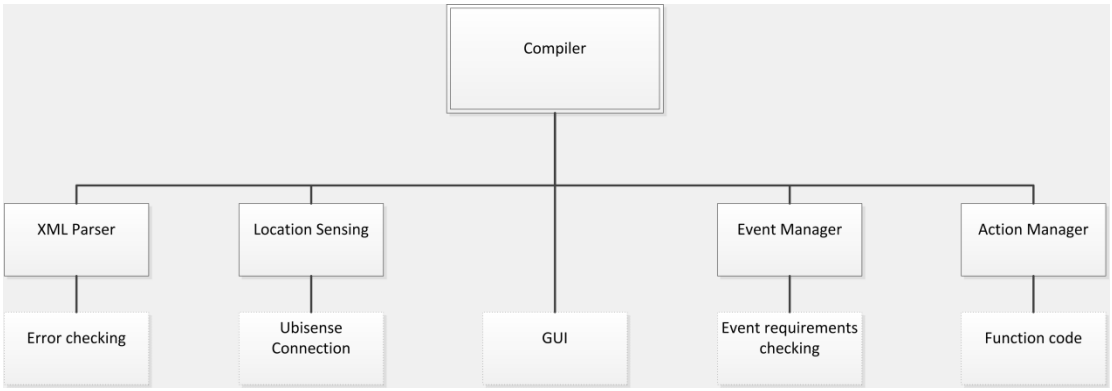

Figure 6-1: *Compiler components overview*

Basic components include the XML parsing process, the location sensing, the event manager and the action manager. Despite the fact that a connection to Ubisense system is mandatory for this implementation, generally it is something that can change depending on the installed localization system. In our case, we used the Ubisense system that IRoom is equipped with. The GUI of this compiler is designed in a user friendly way. Its tabbed design allows the user to easily browse the parsed source code and have a quick overview of the code, before executing the loaded scenario. The embedded log viewer will keep track and report everything on time. It is able to report several types of warnings and errors when parsing the source code, while at run-time it is able to provide relevant, event-driven feedback for the given scenario.

There are specific tasks that the compiler has to execute under a specific order. Starting from the successful parsing of the XML source code files, a connection to the localization system has to be established. Once we retrieve new location data from the system, the event manager processes these data. In case the requirements of a user defined event are met, the action manager is becomes responsible for executing the corresponding function code of that event. Below is the compiler's task flow diagram. Each task will be explained in detail in the following sub sections of this chapter.

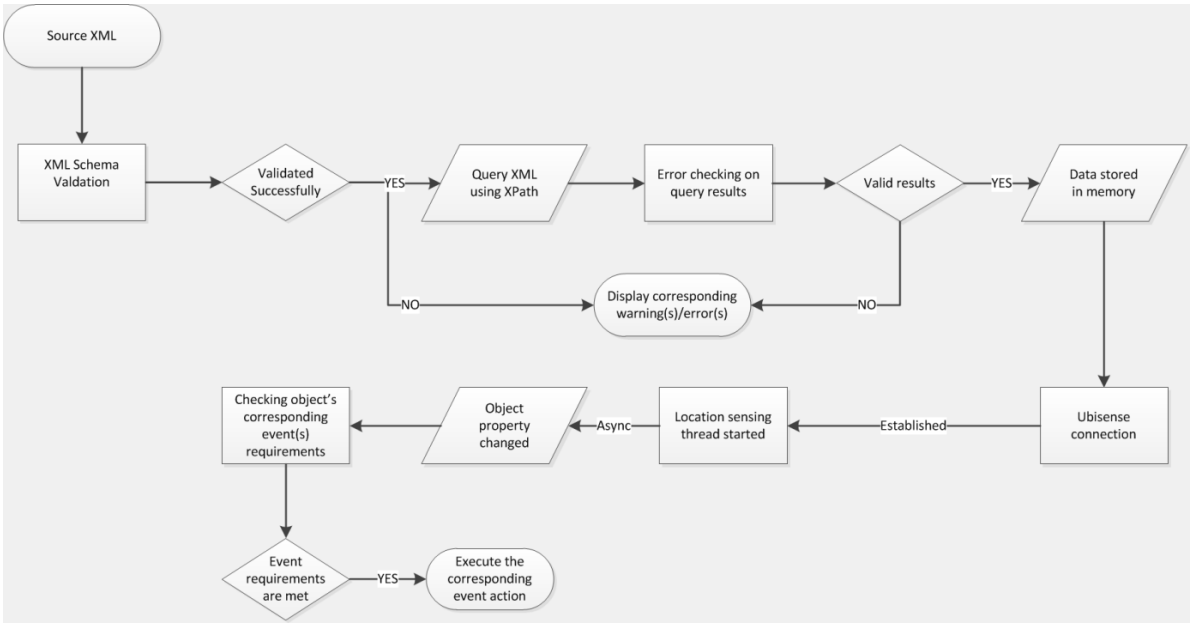

Figure 6-2: *Compiler's flow diagram*

5.2 XML Parsing

The first task which a compiler has to perform, is the parsing of the given source code. Parsing techniques can differ from source code to source code, depending on the language that the source code is written in. In our case, we need to parse source code files which are written in XML.

Given the .NET framework library, parsing an XML file can be done in several ways by using premade classes from the library. For our purpose, we choose to use the XPathDocument class which is able to parse the XML file using the XPath (XML Path Language) query language. XPath is a language for finding information in an XML file. It's, kind of, SQL for XML files. It is used to navigate through elements and attributes in an XML document. In addition, XPath can also be used to compute values (strings, numbers or Boolean values) from the content of an XML document.

Several XPath queries have to be executed to get the whole data contained on the source files. For example, to be able to get the id of the first tag that is registered on our system, we have to execute the following query on the system file:

```
"/System/Tags/Tag[1]/@id"     (Microsoft's parser starts the enumeration from number 1)
```

Getting data from the environment file -for example the name of the first declared object- can be done by executing the following query:

```
"/Environment/Objects/Object[1]/Name"
```

Likewise, getting the action id of the first Enter event, as it is declared on the scenario file, is done by executing the following query:

```
"/Scenario/Events/BaseEvents/Enter[1]/@action"
```

5.2.1 XML Error Checking

To ensure that the user's source code is valid, we have to use some kind of error checking. In our case, the compiler performs a two-phase error checking process.

At the first phase, the compiler tries to validate each one of the three XML source files against the XML Schemas that the compiler has built-in. This operation is performed using the XmlValidatingReader class of the .NET framework library. In case of a validation error, the compiler will provide feedback to the user via the log viewer.

In this example, an enter event is declared in the scenario.xml file without having an event id:

```
<Enter id="enter1" action="action1" distance="distance1">
        <Src>Track 1</Src>
        <Dst>Music Player</Dst>
</Enter>
```

We get the following error message:

```
[XML Schema error][scenario.xml] Line: 126, Pos: 14. The required attribute 'id' is missing.
```

At the second phase, we have already validated successfully the given source code, but Schema validation does not cover all error cases. One such case is when one can declare an Enter event that, for instance, states "when ObjectA comes close to ObjectB, then do actionA" in the scenario file, but the objects themselves are declared in the environment file. So, we cannot be sure if both objects exist before registering the event to the compiler. For cases like this, the compiler has to retrieve the object list from its memory and search for the specific objects in the scenario file. Below it's a flow diagram showing exactly how the secondary phase error checking works to identify such error cases.

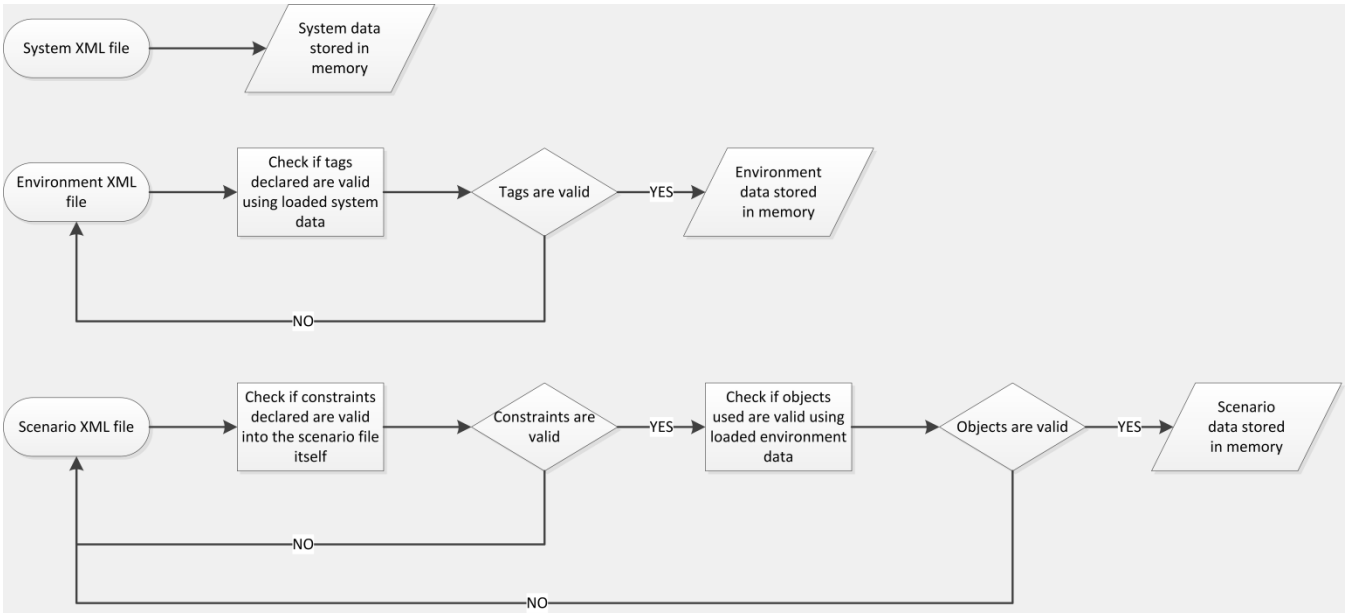

Figure 6-3: *Secondary phase error checking*

The error reported for the secondary phase is similar to the first phase. For example, in case we try to use a non-existent "tag9" in the environment file, which is not declared in the system file, we get the following error message:

```
[XML error][environment.xml] Tag "tag9" does not exist on current System
```

Compiler is also able to provide some warning messages as well. For example, when composing a scenario file, it is strongly recommended to define some constraints as defaults. Otherwise, the user will have to explicitly declare every constraint that is needed for every event. In that case, the compiler will throw a warning message, informing the user that some constraints are not declared as defaults.

```
<Distances>
        <Distance id="distance1" min="0" max="2" default="true"/>
</Distances>
```

```
[XML warning][scenario.xml] Default Distance is not declared
```

5.3 Location Sensing

Location sensing plays a crucial role when it comes to a developing software such as this compiler. Depending on the installed localization system that we want to get data from, connecting and retrieving such data asynchronously and at real-time may be different for each system. In our case, we used the Ubisense localization system which comes with a .NET API and provides great functionality and comfort.

By using the Ubisense system, a developer has the possibility to register spatial relations between objects in ambient space to the system. By doing this, the system is able to automatically provide feedback about the current state of a specific spatial relation in any application which is connected using the API. Such spatial relations can also be registered through code to the Ubisense system.

We examined the possibility of creating the compiler using this handy functionality that the API provides, but instead, the decision was not to take advantage of this. Ubisense is not the only localization system available. Thus, we want our implementation to be written in such a way, so as to be compatible with other real-time location systems as well. In our implementation, we made the least possible usage of the Ubisense API by only getting simple location events of the tags in our 3D space. So, every time the system senses a change in the location of a tag, it asynchronously sends us the new location of it, eliminating the need to poll the system in order to get any new locations. Once the compiler receives a new location, it updates the location of the corresponding object. Then this location is added to the object's location history.

5.4 The event manager

In our language, in order a specific action to be triggered, it is mandatory for a corresponding event to be raised. For an event to be raised, there are several requirements that have to be met. These requirements often relate to the speed of a given object, the acceleration, the spatial relation between several objects etc. The event manager is responsible to tell whether an event has all its requirements fulfilled or not.

At the time a position change occurs, the event manager gets the object whose position has just changed. Then, a search task is performed to find all the registered events which relate to that specific object. For each one of these events, depending on the event type, the event manager checks the requirements one by one, starting from the base requirement (these requirements differ from event type to event type), to then proceed to the several constraint requirements. Let's take the following enter event as an example:

```
<!-- Src object comes close to Dst object
        Attributes: id, action, distance (default distance will be used if not set) -->
      <Enter id="enter1" action="action1" distance="distance1">
        <Src>Mobile phone</Src>
        <Dst>Desk</Dst>
      </Enter>
```

In this example, let's suppose that the location of the mobile phone object was just changed. The event manager will be noticed accordingly about this incident. After finding the event which is related to this object, in our case the example event stated above, the event has to start the requirement checking procedure. For an enter event, the distance between those two objects have to be smaller or equal to the defined distance with id "distance1". This applies to enter event types.

For example if we register the same event as a leave event:

```
<!-- Src object leaves Dst object
        Attributes: id, action, distance (default distance will be used if not set) -->
      <Leave id="leave1" action="action2" distance="distance1">
        <Src>Mobile phone</Src>
        <Dst>Desk</Dst>
      </Leave>
```

Then, the same procedure will follow but having declared the event as a leave event, the distance will have to be greater than the defined in order to be raised.

5.5 The action manager

The final task that the compiler has to perform is to trigger the action which is related to a specific event every time this event is raised by the event manager. That does not mean that every event is related to a specific action. There can also be events which do not have any corresponding actions at all. In any case, the action manager will loop through the action list in order to search for a corresponding action for the raised event.

Once an action has been found, then, the action manager will perform this action. Generally, actions are nothing more than simple function calls which contain source code based on the given scenario. For example, for every action in our Music Player scenario, we had to write the part of the code which the compiler communicates with the Music Player application. The part below is the source code which tells the Music Player to put the first track at the second position in the playlist. At the same time, the compiler provides feedback via the log viewer about the new track1's position.

```
private void Function2()
{
        Program.mainForm.appendLog("[Event] Track 1 : position 2");

        Program.playerForm.deleteFromTrackList("Track 1");

        Program.playerForm.addToTrackList(2, "Track 1");
}
```

Figure 6-4: Experiment conducted in IRoom, LIMSI-CNRS

5.6 Graphical user interface

The GUI of this compiler is written using Windows Forms and is designed in a user friendly way. Its tabbed design allows the user to easily browse the parsed source code and have a quick overview of it. At each tab, one can find the loaded data that each XML file contains and review them before executing the scenario. Loading XML files is as easy as picking the folder which contains the source code, after the user has clicked the Load Folder option from the File Menu. There are also options to load each XML file separately. Finally, the log viewer which is located at the bottom of the main form is able to provide detailed feedback for any kind of situation. Messages containing parsing errors, warnings, messages of success, as well as run-time event related messages appear on the log viewer ordered by date/time. Below is a snapshot of the compiler at run-time, displaying the scenario tab and capturing some events which are displayed on the log viewer.

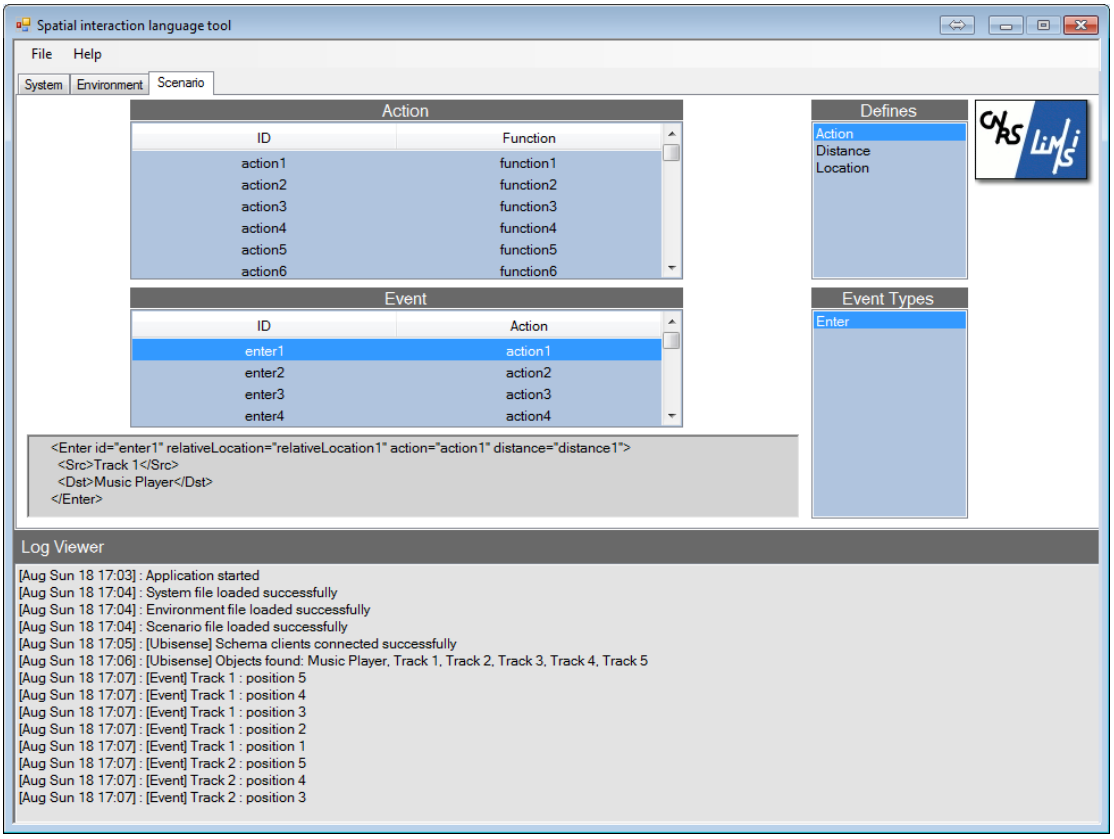

Figure 6-5: *Compiler snapshot*

6 Conclusion and Perspectives

Summing up, in this thesis we reviewed several research implementations which make use of one or more properties from the notion of spatial interaction. We observed that a lot of them used location-based methods or proximity-based methods. Some others used orientation-based methods in combination with one or more of the aforementioned ones.

From the above assessment we reached to the conclusion that no existent language can fully describe the full range of interaction that takes place in those systems. Thus, in this thesis, we presented an event-driven language that one can use to describe such an interaction within ambient environments. We identified the characteristics of the objects that are usable for spatial interaction. We also extracted the potential constraints and defined elementary events caused by the direct alteration on certain object attributes, as well as events which occur when a combination of elementary events takes place.

Finally, we effectively validated our suggested language by conducting an experiment in the IRoom, which entailed a real-life scenario, and was conducted with success.

Even though the above experiment was successful, there is still room for improvement. Further research could include, for example, trying to move away from the restrictions of inputting physical coordinates. Such an idea would imply designing a system that would be in a position to identify physical directions as an input of physical coordinates. Taking it a step further, developing a suitable IDE for the needs of our suggested spatial interaction language, would result to a more effortless, efficient and straightforward programming experience.

Bibliography

[1] Mark Weiser. (1991)
The computer for the 21st century,
Scientific American, vol 265, no 3, pp 94-104, 1991

[2] Hiroshi Ishii and Brygg Ullmer. (1997)
Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms.
In Proceedings of CHI 1997, Atlanta, GA, ACM, pp. 234-241.

[3] Aylward, R., and Paradiso, J. A. (2006)
Sensemble: A Wireless, Compact, Multi-User Sensor System for Interactive Dance.
In Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France, pp. 134-139.

[4] Tanaka, A., and Gemeinboeck, P. (2006)
A framework for spatial interaction in locative media.
In Proceedings of New Interfaces for Musical Expression (NIME). France, Paris: IRCAM, pp. 26-30.

[5] Bongers, B., and Mery, A. (2007)
Interactivated Reading Table.
University of Technology Sydney, Cairns.

[6] Johanson, B., Fox A., and Winograd, T. (2002)
The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms.
IEEE Pervasive Computing Magazine, 1(2).

[7] G. Pruvost, Y. Bellik. (2009)
Ambient Multimodal Human Computer Interaction
in proceedings of the poster session at The European Future Technologies Conference FET09

[8] Buxton, W. (1995). Ubiquitous Media and the Active Office. (1995)
Ubiquitous Video, Nikkei Electronics, 3.27 (no. 632),187-195.
Interaction Design and Information Visualization for Wall-Size Displays with User Tracking

[9] Anind K. Dey and Gregory D. Abowd. (2000)
Towards a Better Understanding of Context and Context-Awareness
In the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000),

[10] Eva Hornecker. (2004)
A Framework for the Design of Tangible Interaction for Collaborative Use
Paper at the Danish HCI Symposion, University of Aalborg, Nov. 16. HCI Lab Technical pp.57-61

[11] Eva Hornecker, Jacob Buur. (2006)
Getting a Grip on Tangible Interaction: A Framework on Physical Space and Social Interaction
Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ‘06

[12] Hiroshi Ishii. (2008)
Tangible Bits: Beyond Pixels
Proceedings of the Second International Conference on Tangible and Embedded Interaction (TEI'08)

[13] Michael Batty
Spatial Interaction: Encyclopedia of Geographic Information Science
http://www.spatialcomplexity.info/files/2012/01/BATTY-Spatial-Interaction-Encyclopedia.pdf

[14] Eva Hornecker. (2005)
Space and Place – Setting the Stage for Social Interaction
Position paper for ECSCW05 workshop 'Settings for Collaboration: the role of place'

[15] Eva Hornecker. (2006)
An Encompassing View on Tangible Interaction: A Framework
Position Paper for CHI 2006 Workshop: What is the Next Generation of Human-Computer Interaction?

[16] Eva Hornecker. (2005)
Tangible Interaction Design, Space, and Place
Position paper for 'Space, Place and Experience in HCI' Workshop at Interact 2005

[17] Eoghan Parle , Aaron Quigley. (2006)
Proximo, Location-Aware collaborative Recommender
School of Computer Science and Informatics, University College Dublin Ireland

[18] Thorsten Prante, Carsten Röcker, Norbert Streitz, Richard Stenzel, Carsten Magerkurth, Daniel van Alphen, Daniela Plewe. (2003)
Hello.Wall - Beyond Ambient Displays
In Proceedings of Ubicomp'03, pp. 277-278

[19] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper (2001)
Implementing a Sentient Computing System
IEEE Computer, 34(8):50-56

[20] Ryan David Bruner (2011)
The Curb App - Exploring Spatial Interactions
http://www.utexas.edu/finearts/aah/sites/files/aah/files/bruner.pdf

[21] Roy Want, Andy Hopper, Veronica Falcão, Jonathan Gibbons. (1992)
The Active Badge Location System
ACM Transactions on Information Systems, Volume 10 Issue 1, Pages 91-102

[22] Nicolas Roussel, HelenEvans, Heiko Hansen. (2004)
MirrorSpace: using proximity as an interface to video-mediated communication
Proc. of Pervasive 2004 Conference, volume 3001 of Lecture Notes in Computer Science

[23] Sofiane Gueddana, Nicolas Roussel (2006)
Pêle-Mêle, a video communication system supporting a variable degree of engagement
CSCW '06 Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work
Pages 423-426

[24] Jennica Falk , Peter Ljungstr , Staffan Björk , Rebecca Hansson (2001)
Pirates: Proximity-Triggered Interaction in a Multi-Player Game
In Proceedings of Human Factors in Computing Systems: CHI 2001

[25] Jun Rekimoto , Yuji Ayatsuka , Michimune Kohno , Hauro Oba (2003)
Proximal Interactions: A Direct Manipulation Technique for Wireless Networking
In Proceedings of INTERACT2003, Sep.-Oct

[26] Peter Tandler , Thorsten Prante , Christian Müller-Tomfelde , Norbert Streitz , Ralf Steinmetz (2001)
ConnecTables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces
UIST '01 Proceedings of the 14th annual ACM symposium on User interface software and technology
Pages 11-20

[27] Frank Chun Yat Li, David Dearman & Khai N. Truong (2009)
Virtual Shelves: Interactions with Orientation Aware Devices
UIST '09 Proceedings of the 22nd annual ACM symposium on User interface software and technology
Pages 125-128

[28] Shwetak N. Patel , Jun Rekimoto , Gregory D. Abowd (2006)
iCam: Precise at-a-Distance Interaction in the Physical Environment
In the proc. of Pervasive 2006

[29] Sean Gustafson, Daniel Bierwirth and Patrick Baudisch (2009)

Imaginary Interfaces: Spatial Interaction with Empty Hands and without Visual Feedback
UIST '10 Proceedings of the 23nd annual ACM symposium on User interface software and technology
Pages 3-12
[30] Liam R. E. Quin Liam R. E. Quin
XML Schema http://www.w3.org/standards/xml/schema

[31] International Organization for Standardization and International Electrotechnical Commission, 2006

[32] Malik, A. (2009).
RTLS for Dummies. Wiley, Hoboken, N.J

[33] Ward, A. M. R., and Webster, P. M. (2009).
Location Device and System using UWB and Non-UWB Signals.

[34] Muñoz, D., Bouchereau, F., Vargas, C., and Enriquez-Caldera, R. (2009).
Position Location Techniques and Applications.

[35] Pete Steggles, Stephan Gschwind.
The Ubisense Smart Space Platform