University Of Crete
Computer Science Department

BASIC BUILDING BLOCKS FOR AMBIENT INFORMATION SYSTEMS

By

KONSTANTINOS VANDIKAS

Master’s Thesis

Heraklion, July 2007
BASIC BUILDING BLOCKS FOR AMBIENT INFORMATION SYSTEMS

by

KONSTANTINOS VANDIKAS

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Author:

_____________________________________________
Konstantinos Vandikas, Computer Science Department

Board of enquiry:

Supervisor

_____________________________________________
Constantine Stephanidis, Professor

Co-Supervisor

_____________________________________________
Anthony Savidis, Associate Professor

Member

_____________________________________________
Apostolos Traganitis, Professor

Member

_____________________________________________
Antonis Argyros, Associate Professor

Approved by:

_____________________________________________
Panos Trahanias, Professor
Chairman of the Graduate Studies Committee

Heraklion, July 2007
Recent research on Ambient Information Systems has explored the provision of information systems across a wide range of deployment contexts with emphasis on user mobility and location-oriented information delivery.

Such work has resulted in a broad spectrum of concrete outcomes regarding design approaches for information presentation ‘on the move’ with primary emphasis on unobtrusive feedback, context-sensitive information mining and overall aesthetic quality.

Additionally, the consolidation of such work seems to lead to somehow standardized technical vocabulary regarding the primary requirements of mobile information systems, the dominant software design patterns, micro and macro architectures, structure of information and the technical approaches to location sensing.

However, there is no systematic work on addressing the implementation of these systems by proposing a universal set of basic building blocks and architectural elements. We argue that the general domain of ambient intelligence would significantly benefit from the identification of prominent building blocks for ambient information systems.

The collection of building blocks is comprised of a data access layer, a location-sensing client, a data synchronization service, a Map Viewer and a mobile location-administration system. For each building block, we provide a systematic account of the main challenges it addresses in the context of ambient information systems. The data access layer provides a flexible mapping between physical artifacts within the environment and objects within the relational database. The location-sensing client allows us to query the wireless infrastructure to retrieve user positions in real time. The data synchronization service allows us to ensure that information mirrored in distributed resources remains always up to date. The Map Viewer provides a flexible, yet aesthetically pleasing graphical representation of the user’s surroundings. Finally, the mobile location-administration system allows the administrator to interactively program ‘on site’ the physical position of information elements.

Supervisors: Constantine Stephanidis, Professor
Anthony Savidis, Associate Professor
ΠΡΟΣΦΑΤΕΣ ΜΕΛΕΤΕΣ ΓΙΑ ΤΑ ΔΙΆΧΥΤΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΩΝ έΧΟΥΝ ΑΣΧΟΛΗΘΕΙ ΜΕ ΤΗΝ ΕΞΕΡΕΥΝΗΣΗ ΤΩΝ ΜΕΘΟΔΟΛΟΓΙΩΝ ΔΙΑΘΕΣΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΣΕ ΜΙΑ ΕΥΡΕΙΑ ΚΛΙΜΑΚΑ ΣΕΝΑΡΙΩΝ ΜΕ ΚΥΡΙΟΤΕΡΗ έΜΦΑΣΗ ΣΕ ΑΥΤΟ ΌΠΟΥ Ο ΧΡΗΣΤΗΣ ΒΡΙΣΚΕΤΑΙ ΣΕ ΚΙΝΗΣΗ ΚΑΙ ΧΡΗΣΙΜΟΠΟΙΕΙ ΜΙΑ ΦΟΡΗΤΗ ΣΥΣΚΕΥΗ ΓΙΑ ΝΑ ΛΑΒΕΙ ΠΛΗΡΟΦΟΡΙΕΣ ΣΧΕΤΙΚΑ ΜΕ ΤΟ ΠΕΡΙΒΆΛΛΟΝ ΤΟΥ ΚΑΙ ΤΙΣ ΔΥΝΑΤΟΤΗΤΕΣ ΠΟΥ ΤΟΥ ΠΑΡΕΧΕΙ.

ΑΥΤΕΣ ΟΙ ΜΕΛΕΤΕΣ ΈΧΟΥΝ ΑΠΟΔΩΣΕΙ ΜΙΑ ΠΛΗΘΩΡΑ ΠΑΡΑΔΕΙΓΜΑΤΩΝ ΑΠΟ ΣΧΕΔΙΑΣΤΙΚΕΣ ΠΡΟΣΕΓΓΙΣΕΙΣ ΌΣΟΝ ΑΦΟΡΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΠΑΡΟΥΣΙΑΣΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΓΙΑ ΤΟΝ ΚΟΣΜΟ ΤΟΥ ΧΡΗΣΤΗ ΜΕ ΤΕΤΟΙΟ ΤΡΟΠΟ ΩΣΤΕ ΜΗΝ ΤΟΥ ΑΠΟΣΤΑΤΙЗΕΙ ΤΗΝ ΠΡΟΣΟΧΗ ΑΠΟ ΤΗΝ ΤΡΕΧΟΥΣΑ ΕΡΓΑΣΙΑ, ΑΛΛΑ ΝΑ ΠΑΡΟΥΣΙΑΞΕΤΕ ΜΕ ΑΙΣΘΗΤΙΚΑ ΕΥΧΑΡΙΣΤΟ ΚΑΙ ΚΑΤΑΝΟΗΤΟ ΤΡΟΠΟ ΣΕ ΠΟΙΚΙΛΕΣ ΒΑΘΜΙΔΕΣ ΒΑΘΜΙΔΕΣ.

ΕΠΙΠΡΟΣΘΕΤΑ, Η ΣΤΑΘΕΡΟΠΟΙΗΣΗ ΤΕΤΟΙΩΝ ΜΕΛΕΤΩΝ ΦΑΙΝΕΤΕ ΝΑ ΕΧΕΙ ΟΔΗΓΗΣΕΙ ΣΤΗ ΣΥΝΚΕΝΤΡΩΣΗ ΕΝΟΣ ΕΝΙΑΙΟΥ ΤΕΧΝΙΚΟΥ ΛΕΞΙΛΟΓΙΟΥ ΣΧΕΤΙΚΑ ΜΕ ΤΙΣ ΠΡΟΤΕΥΟΝΤΑΙ ΑΠΑΙΤΗΣΕΙΣ ΓΙΑ ΤΑ ΦΟΡΗΤΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΩΝ, ΤΑ ΚΥΡΙΑΡΧΙΚΑ ΠΡΟΤΥΠΑ ΣΧΕΤΙΚΑΣ, ΤΙΣ ΜΙΚΡΟ / ΜΑΚΡΟ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ, ΤΗ ΔΟΜΗ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΙΣ ΤΕΧΝΙΚΕΣ ΠΡΟΣΕΓΓΙΣΕΙΣ ΓΙΑ ΤΟΝ ΕΝΤΟΠΙΣΜΟ ΘΕΣΗΣ.

ΠΑΡΑ ΤΗΝ ΕΠΙΤΥΧΙΑ ΌΛΩΝ ΤΩΝ ΠΑΡΑΠΑΝΩ, ΣΤΗΝ ΠΑΡΟΥΣΙΑ ΒΙΒΛΙΟΓΡΑΦΙΑ ΦΑΙΝΕΤΕ ΝΑ ΑΠΟΣΥΝΑΞΕΙ ΜΙΑ ΣΥΣΤΗΜΑΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ ΣΧΕΤΙΚΗ ΜΕ ΤΗΝ ΥΛΟΠΟΙΗΣΗ ΤΕΤΟΙΩΝ ΣΥΣΤΗΜΑΤΩΝ Η ΟΠΟΙΑ ΘΑ ΠΡΟΤΕΙΝΕΙ ΕΝΑ ΕΝΙΑΙΟ ΣΥΝΟΛΟ ΑΠΟ ΒΑΣΙΚΑ ΚΑΤΑΣΚΕΥΑΣΤΙΚΑ ΤΜΗΜΑΤΑ ΚΑΙ ΣΤΟΙΧΕΙΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΓΙΑ ΤΕΤΟΙΑ ΣΥΣΤΗΜΑΤΑ. ΥΠΟΣΤΗΡΙΖΟΥΜΕ ΌΤΙ Ο ΕΥΡΥΤΕΡΟΣ ΧΩΡΟΣ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΔΙΆΧΥΤΗΣ ΕΥΧΑΡΙΣΤΟ ΘΑ ΜΠΟΡΟΥΣΕ ΝΑ ΕΠΙΦΕΛΗΘΕΙ ΣΗΜΑΝΤΙΚΑ ΑΠΟ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΤΩΝ ΠΡΩΤΑΡΧΙΚΩΝ ΚΑΤΑΣΚΕΥΑΣΤΙΚΩΝ ΤΜΗΜΑΤΩΝ ΓΙΑ ΔΙΆΧΥΤΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΩΝ.

Η ΕΡΓΑΣΙΑ ΑΥΤΗ ΠΡΟΤΕΙΝΕΙ ΜΙΑ ΣΥΛΛΟΓΗ ΚΑΤΑΣΚΕΥΑΣΤΙΚΩΝ ΤΜΗΜΑΤΩΝ Η ΟΠΟΙΑ ΑΠΟΤΕΛΕΙΤΑΙ ΑΠΟ ΕΝΑ ΜΗΧΑΝΙΣΜΟ ΠΡΟΣΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ, ΕΝΑ ΠΕΛΑΤΗ ΓΙΑ ΜΗΧΑΝΕΣ ΕΝΤΟΠΙΣΜΟΥ ΘΕΣΗΣ, ΜΙΑ ΥΠΗΡΕΣΙΑ ΣΥΝΧΡΟΝΙΣΜΟΥ, ΕΝΑ ΜΗΧΑΝΙΣΜΟ ΓΙΑ ΤΗΝ ΠΑΡΟΥΣΙΑΣΗ ΧΑΡΤΩΝ ΚΑΙ ΕΝΑ ΣΥΣΤΗΜΑ ΦΟΡΗΤΗΣ ΑΝΤΙΣΤΟΙΧΙΑΣ ΧΩΡΙΚΩΝ ΑΝΑΦΟΡΩΝ. ΚΑΘΕ ΤΜΗΜΑ ΑΝΑΛΑΜΒΑΝΕΙ ΝΑ ΕΠΙΤΥΧΕΙ ΜΙΑ ΣΥΣΤΗΜΑΤΙΚΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΩΝ ΠΡΟΚΛΗΣΕΩΝ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΖΕΙ. Ο ΜΗΧΑΝΙΣΜΟΣ ΠΡΟΣΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΠΑΡΕΧΕΙ ΜΙΑ ΕΥΛΟΓΗΤΗ ΔΙΑΣΥΝΔΕΣΗ ΑΝΑΜΕΣΑ ΣΤΑ ΦΥΣΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΚΑΙ ΣΤΗΝ ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΧΕΣΙΑΙΚΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ. Ο ΠΕΛΑΤΗΣ ΓΙΑ ΜΗΧΑΝΕΣ ΕΝΤΟΠΙΣΜΟΥ ΘΕΣΗΣ ΜΑΣ ΕΠΙΤΡΕΠΕΙ ΝΑ ΕΠΕΡΩΤΗΣΟΥΜΕ ΤΗΝ ΑΣΥΡΜΑΤΗ ΥΠΟΔΟΜΗ ΔΙΚΤΥΟΥ ΚΑΙ ΝΑ ΑΝΑΚΤΗΣΟΥΜΕ ΤΗ ΘΕΣΗ ΕΝΟΣ ΑΝΤΙΚΕΙΜΕΝΟΥ. Η ΥΠΗΡΕΣΙΑ ΣΥΝΧΡΟΝΙΣΜΟΥ ΜΑΣ ΕΠΙΤΡΕΠΕΙ ΝΑ ΔΙΑΣΦΑΛΙΣΟΥΜΕ ΤΟ ΓΕΓΟΝΟΣ ΌΤΙ ΚΆΘΕ ΠΛΗΡΟΦΟΡΙΑ ΠΟΥ
ανακλάται σε κατανεμημένες πηγές παραμένει πάντα επίκαιρη. Ο μηχανισμός
παρουσίασης χαρτών παρέχει μια ευέλικτη και αισθητικά ευχάριστη απεικόνιση του
περιβάλλοντος του χρήστη.

Τέλος, το σύστημα φορητής διαχείρισης της αντιστοίχισης χωρικών σημείων παρέχει τη
dυνατότητα στον διαχειριστή να αλληλεπιδράσει άμεσα με το περιβάλλον του και
καταγράψει τις θέσεις των αντικειμένων που βρίσκονται γύρω του στον χώρο.

Επόπτης: Κωνσταντίνος Στεφανίδης, Καθηγητής
Επιβλέπων: Αντώνιος Σαββίδης, Αναπληρωτής Καθηγητής
In memory of my father
ΕΥΧΑΡΙΣΤΙΕΣ (ACKNOWLEDGEMENTS)

Οφείλω να αναγνωρίσω τη συμβολή συγκεκριμένων ανθρώπων στην επιτυχή ολοκλήρωση της μεταπτυχιακής αυτής εργασίας. Ευχαριστώ τους επόπτες μου Καθηγητή Κωνσταντίνο Στεφανίδη και Αναπληρωτή Καθηγητή Αντώνιο Σαββίδη καθώς και τα μέλη της εξεταστικής επιτροπής Καθηγητή Απόστολο Τραγανίτη και Αναπληρωτή Καθηγητή Αντώνιο Αργυρό.

Η συνεχής στήριξη της οικογένειας μου, καθώς και οι θυσίες που κάνανε δίνοντας μου δύναμη να συνεχίσω την προσπάθεια αυτή, δεν μπορεί να περιγραφεί με λέξεις.

Τέλος, πάνω από όλα θα ήθελα να ευχαριστήσω τους φίλους μου που μοιράστηκαν μαζί μου τα σχόλια τους και τις ανησυχίες μου κατά την εξέλιξη αυτής της έρευνας. Τους εύχομαι ότι καλύτερο στη ζωή τους.
# Table of Contents

Ευχαριστίες (Acknowledgements) ....................................................................................... 11  
List Of Figures ....................................................................................................................... 4  
List Of Tables ......................................................................................................................... 5  
1 Introduction .................................................................................................................. 1  
   1.1 Challenges ............................................................................................................. 1  
   1.2 Contributions ......................................................................................................... 3  
   1.3 Thesis Statement ................................................................................................... 3  
   1.4 Thesis Outline ........................................................................................................ 4  
2 Background And Related Work ...................................................................................... 5  
   2.1 Location-Sensing Systems ...................................................................................... 5  
       2.1.1 Distance based ............................................................................................... 5  
       2.1.2 Direction based .............................................................................................. 6  
       2.1.3 Signal-strength based ..................................................................................... 6  
       2.1.4 GPS ................................................................................................................ 8  
       2.1.5 Ekahau Positioning Engine .............................................................................. 8  
       2.1.6 NearMe .......................................................................................................... 9  
       2.1.7 E911 ............................................................................................................ 9  
       2.1.8 Collaborative Location Sensing ..................................................................... 10  
       2.1.9 Robot Localization ........................................................................................ 10  
       2.1.10 Summary of Location Sensing options .......................................................... 10  
   2.2 Ambient Information Systems .............................................................................. 12  
       2.2.1 Position-Based Interaction for Indoor Ambient Intelligence Environments ... 12  
       2.2.2 Bus Mobile ................................................................................................... 13  
       2.2.3 Dangling String ............................................................................................. 14  
       2.2.4 Digital Family Portrait ................................................................................... 15  
       2.2.5 Infocanvas .................................................................................................... 16  
       2.2.6 Information Percolator ................................................................................. 17  
       2.2.7 Irwin ............................................................................................................ 18  
       2.2.8 Kadinsky – Aesthetic Information Collages ................................................... 19  
       2.2.9 Kimura ......................................................................................................... 20  
       2.2.10 Lumitouch .................................................................................................... 20
2.2.11 Notification Collage ................................................................. 22
2.2.12 Scope ..................................................................................... 22
3 Design .............................................................................................. 24
  3.1 Overview .................................................................................... 24
  3.2 Information Capacity ................................................................. 27
  3.3 Notification Level ................................................................. 28
  3.4 Representation Fidelity ......................................................... 28
  3.5 Aesthetic emphasis ............................................................. 29
  3.6 Modularity ............................................................................... 29
  3.7 Scalability ............................................................................... 30
  3.8 Extensibility ............................................................................ 31
4 Implementation .............................................................................. 32
  4.1 Overview .................................................................................... 32
    4.1.1 .NET Compact Framework .............................................. 34
    4.1.2 Overview of the J2ME ....................................................... 35
    4.1.3 Multiple Platforms .......................................................... 36
    4.1.4 Multiple Languages ........................................................ 37
    4.1.5 Web Services ................................................................. 37
    4.1.6 Development Tools ....................................................... 38
    4.1.7 Summary ......................................................................... 39
  4.2 Data infrastructure ................................................................. 41
4.3 AMIS Macroscopic architecture ............................................... 44
  4.3.1 Data-Access Layer ............................................................... 45
  4.3.2 Location-Sensing client ..................................................... 46
  4.3.3 Network-layer ................................................................. 48
  4.3.4 Map Viewer ....................................................................... 49
  4.3.5 Data Synchronization Service .......................................... 52
4.4 AMIS Tools ............................................................................... 54
  4.4.1 Mobile Location-Administration Tool ............................... 54
  4.4.2 Desktop Location-Administration Tool .............................. 56
  4.4.3 Content Management Tool ............................................. 58
5 Case Study: A Museum Guidance System ..................................... 60
  5.1 Area and item positioning .................................................... 60
  5.2 Content association .............................................................. 61
List of Figures

Figure 1: Bus Mobile
Figure 2: Dangling string
Figure 3: Digital Family Portrait
Figure 4: DFP Architecture
Figure 5: Infocanvas
Figure 6: Kadinsky architecture
Figure 7: Kimura Architecture
Figure 8: Lumitouch
Figure 9: Notification Collage
Figure 10: Scope
Figure 11: Database diagram
Figure 12: AMIS Macroscopic architecture
Figure 13: Data-Access layer class diagram
Figure 14: Location-sensing class diagram
Figure 15: Ekahau XML packet
Figure 16: Network-layer class diagram
Figure 17: Map Viewer class diagram
Figure 18: Map Viewer file caching class diagram
Figure 19: Map Viewer Engine class diagram
Figure 20: Synchronization server class diagram
Figure 21: Synchronization client class diagram
Figure 22: Mobile Location-Administration Tool area functionality 1
Figure 23: Mobile Location-Administration Tool area functionality 2
Figure 24: Desktop Location-Administration Tool
Figure 25: Content management tool
Figure 26: Mycenae floor-plan
Figure 27: Synchronization Console
Figure 28: Visitor emulation
Figure 29: Blender floor plan
LIST OF TABLES

Table 1: Comparison of location sensing options ............................................................... 11
Table 2: Smart client development platform features.......................................................... 33
Table 3: .NET CF vs. J2ME ................................................................................................... 40
Table 4: Ram requirements for single texture in megabytes, with and without mipmapping
(MM) .................................................................................................................................. 57
Table 5: Maximum texture size for popular GPUs .............................................................. 57
Table 6: Item content snippet.............................................................................................. 62
1 INTRODUCTION

The immense progress in information technology has opened up exciting research challenges in personal and pervasive computing, transcending the traditional computing paradigms. The aim of pervasive computing is to create systems that are embedded in the environment, almost invisible and constantly available enhancing the information access without distracting the users from their main tasks. Among the emerging technologies expected to prevail in pervasive computing environments are wearable computers, sensors in cars and appliances, smart homes and smart buildings that can make our lives easier and more productive.

The great technological development in wireless communications had a dramatic impact on the deployment of pervasive computing. In addition, the proliferation of high-speed wireless local area networks has enabled users to access the Internet while moving. The goal here is to enable users to interact effectively with their physical surroundings, such as print a document on the closest printer or locate one’s self. Such is the vision of Ambient Intelligence.

Animated and inanimate objects surround people, in their environments. Profoundly enough, animated objects are the people around us, our friends, family members, people we have not yet come to meet, animals and all other living creatures. On the other hand we have inanimate objects; items that comprise our surroundings and may present a point of interest in various contexts. For example, our car is an inanimate object and it is an interesting one since we need it for our daily transportations, we need to know where we parked it so we can use it early in the morning to go to work. Another example of an inanimate object is a museum artifact such as the "Prince of Lilies" which is a plaster relief at the end of the Corridor of Processions found in the Knossos Palace. Notably, the museum artifact is not as practically useful as our car, however, while on a museum tour it is our main concern since to know where it is, how we can see it, or what is it about. Furthermore, another point of interest may be a nuclear factory, which we are visiting as part of an exhibition, or even worse we are working in. What happens in case of a nuclear break down? What if we find ourselves near a contaminated area or what if we need to know our way out of the site in case of an emergency? In this scenario, each area in the nuclear factory becomes our point of interest and we need an intuitive and aesthetically convenient way to interact with it. The main goal of this thesis is to combine the proper methodologies and tools that allow one user to navigate into an indoor environment and interact with a metaverse of information regarding the inanimate objects and their context.

1.1 Challenges

A possible definition of an Ambient Information System (AMIS) is an information system that delivers information using distributed interactive resources for users that are “on the move”.

KONSTANTINOS VANDIKAS
This definition builds upon the 2WEAR Project definition (1) of “ambient user interfaces”. The main difference between the two distinct approaches is that in the case of the 2WEAR Project, the user interfaces are distributed while in the case of an AMIS the distributed property is attributed to the information sources. The outcome of an AMIS is the establishment of a virtual metaverse where users navigate within both the physical (ambient) and digital (information) space. In this metaverse, virtual representations are combined with tangible artifacts within the environment. Hence, the ultimate goal of an AMIS is the facility of What You See Is Where You Get it (WYSIWYG revisited), thus resolving the “where am I” issue for physical spaces.

When building large-scale ambient information systems there are certain challenges that need to be faced. First, for the realization of such a system, it is paramount that applications can exploit information about the context in which they operate preferably in a generic manner. For this purpose, various types of information should be assembled to form a representation of the context of the device, the users and the resources they have at their disposal. Second, the layer of contextual information has to be enriched with on-site spatial information of artifacts that may be points of interest but are deprived of virtual representation. The mobile level editor deals this problem by estimating the artifacts position from a location-sensing system. However, the location-sensing system provides an estimation of the user’s position with an error margin escalating from 2 to 3m. This creates a significant limitation for the level editor, since physical artifacts are not physically distributed in equally spaced 2 to 3m.

A certain course of action must be made in order to ensure a proper clustering, one that enables us to create a distinction between different artifacts that exist in the same physical environment and are less than 2m or 3m apart. Another aspect is the provision of a method for the synchronization of the on-site recording activities since it is more efficient to use more than one people when recording artifacts in large physical spaces. This method has to be made in such a manner that allows us to conserve bandwidth and make sure that no misunderstandings are made when more than two people are recording the same artifact.

Concerning the presentation of contextual information, Pousman et. al. (2) provides a set of guidelines:

- Display information that is important but not critical.
- Move from the periphery to the focus of attention and back again.
- Focus on the tangible; representations in the environment.
- Provide subtle changes to reflect updates in information (should not be distracting).
- Are aesthetically pleasing and environmentally appropriate.

The only known facts for dealing with such challenges are that:

- The emerging domain of ambient information systems is development-wise highly demanding. An AMIS is an ever-growing system, constantly updated with information.
- There are no standardized views of what architecturally comprises such a system. As described in further detail in Section 2 the majority of the literature on such systems
focuses solely on user interface and neglect architectural matters that may lead to an ease in the development-wise approach of a user interface.

• We entirely miss libraries of ready-made components for the implementation process

1.2 Contributions

In absence of systematic work on addressing the implementation of such systems, we attempt to identify reusable high-level architectural components applicable over the “widest” possible range of Ambient Information Systems. We argue that the general domain of ambient intelligence would significantly benefit from the identification of a collection prominent building blocks. Our contribution is the implementation, deployment, validation and documentation of such components as a set of basic building blocks for Ambient Information Systems. The proposed collection is comprised of a Data Access Layer, a Location Sensing Client, a Data Synchronization Service, a Map Viewer and a Mobile Administration Tool.

1.3 Thesis Statement

Ambient Intelligence environments consist of fixed, architecturally created, physical artifacts. Such artifacts model the space inhabited in everyday life, including for example, rooms, corridors, offices and more. These artifacts possess a significant amount of contextual information. A key aspect to that information is position. However, how can we obtain that information? Global positioning systems do not function in indoor spaces. Image processing provides more than enough information regarding a scene of physical artifacts at the cost of highly powerful yet expensive hardware. Thus, our research problem can be stated as:

• Sensing - Locating users within indoors, outdoors and hybrid environment
• Positioning – Record digitally the physical artifacts that surround the user without “crowding” the environment with lots of sensors
• Organizing – Location-oriented and location-aware organization of information items

This thesis proposes a simple approach using a PDA, the 802.11 infrastructure and a relational database system. The user roams the AmI environment holding a PDA that retrieves its position from a location sensing system. On the PDA’s screen, the user sees a visual representation of the floor plan and, using the pen, records the position of each significant artifact. In order to make this procedure more scalable, a synchronization mechanism among the PDAs has been implemented so that each member of the team that
is recording the artifacts can receive updates from teammates that are within physical or screen view range.

Our approach has already been tested using the WLAN infrastructure, while currently we are integrating GPS and sensory systems. The reason behind the WLAN location sensing choice is the widespread use of Access Point that can already be found in most facilities. Aside from location sensing, the other core of approach is Database storage with a relational database, since relational databases are currently the dominant paradigm and are already adopted by most organizations (e.g. museums, malls, libraries, universities, etc.).

The significance of this approach is that it provides a low cost solution using ingredients that exist almost in every environment whether that is dense urban precincts, shopping malls, airports, railway stations, exhibition halls, industrial compounds, conference/convention centers, large office buildings, cultural heritage sites, large museums, theme parks, etc.

### 1.4 Thesis Outline

The thesis is organized as follows. Chapter 2 presents the definition of Ambient Information systems and a classification of these systems. In addition, it provides information on location-sensing techniques. Chapter 3 describes the design overview of the building blocks, and elaborates on their modularity, scalability and extensibility. Chapter 4 provides detailed information on each building block. Chapter 5 presents the case study of the development of the museum e-guide application, and the related evaluation. Chapter 6 discusses the main conclusions of the thesis. Finally, chapter 7 describes our future work.
2 BACKGROUND AND RELATED WORK

This chapter presents the main concepts that are used in the work presented in this thesis, starting from basic location-sensing techniques and moving on to more specific issues on Ambient Information Systems.

2.1 Location-Sensing Systems

This section presents an overview of location-sensing systems, based on the basic technique they use to estimate one’s position, and describes some representative systems of each category.

2.1.1 Distance based

Early location-sensing systems, which targeted to in-building localization, required specialized hardware to determine a device's location. For instance, the Active Badge location system (3) and the more recent Active Bat system (4) are two of the first systems based on distance measurements in order to locate an object. Active Badge uses specialized tags emitting diffuse infrared pulses detected by ceiling-mounted sensors. The system has accuracy granularity of room size, but it needs one base station in every room. The Badge communicates with the base station every 10 sec. One of the main limitations is the sunlight and fluorescent interference with the infrared pulses. Active Bat uses an ultrasound time-of-flight technique to provide accurate physical positioning. Specifically, users and objects must carry Bat tags. These tags emit an ultrasonic pulse to a grid of ceiling-mounted receivers and a simultaneous 'reset' signal over a radio link. Each ceiling sensor measures the time interval from ‘reset’ to ultrasonic pulse arrival and computes its distance from the Bat tag. The Active Bat has location accuracy of nine centimeters in 95% of the measurements. It needs one base station in every 10 square meters. The Bat makes 25 computations in every sec for every room. However, it requires a large infrastructure (ceiling sensor grids) and has a considerable maintenance cost.

The Cricket Location Support System (5) uses both specialized ultrasound and radio frequency receivers to detect signals transmitted by fixed ultrasound emitters. The additional radio frequency signals are used for synchronization of the time measurement and to distinguish ultrasound signals that stem from multi-path effects and ignore them. The mobile object performs triangulation computations relative to the beacons. Cricket has simpler hardware and infrastructure than the previously mentioned systems, but worse accuracy. Specifically, it does not require a grid of ceiling sensors with fixed locations, but returns an estimation of the users’ position with a possible error of a four foot by four-foot
region. In the Spot-On system (6), special tags use radio signal attenuation to estimate distance between tags. In this system, there are multiple base stations, which provide signal strength measurements, and a central server, which aggregates this data. The aim is to localize wireless devices relative to each another, rather than to fixed base stations, allowing for ad-hoc localization. Accuracy depends on the cluster size, and every cluster must have at least two tags. However, radio signal attenuation is less accurate than the time-of-flight.

Finally, EasyLiving (7) uses motion-tracking cameras to determine the distance of an object in a home environment. It needs three cameras in every room and its precision varies. However, vision location systems need high processing power and must continuously struggle in order to maintain analysis accuracy as scene complexity increases.

2.1.2 Direction based

Other location-sensing systems are based on the capability of the nodes to sense the direction from which a signal is received in order to determine a device's location. A representative case of this approach is that of Niculescu and Badri Nath (8). They designed and evaluated a cooperative location-sensing system that operates in an ad-hoc network. They examined two algorithms, the 'DV-Bearing', which allows each node to get angle estimation to a landmark, and 'DV-Radial', which allows a node to get angle estimation to a landmark. Their system uses primarily angle estimations and assumes specialized hardware that allows a host to calculate the angle between two hosts. This can be done through antenna arrays or ultrasound receivers. Hosts gather data, compute their solutions, and propagate them throughout the network. Specifically, nodes immediately adjacent to a landmark get their angle estimations directly from the landmark. Assuming that a node has some neighbors with orientation for a landmark, it will be able to compute its own orientation with respect to that landmark, and forward it further into the network. The main disadvantage is that if a node uses imprecise information to solve its location, it is likely that the solution will have a higher position estimation error than if the information used had accurate readings directly from landmarks that know their position exactly.

2.1.3 Signal-strength based

More recent systems for location sensing-computing use off-the-shelf wireless networking hardware, measuring radio frequency (RF) signal intensity to determine a mobile object's location. The RADAR system (9) was one of the first tracking systems based on the IEEE 802.11 wireless networking technology. RADAR measures at the base station the signal strength and signal-to-noise ratio of signals that wireless devices send, and then uses this data to compute the position within a building. The signal strength measurements employ a signal strength map with approximately one scan every square meter. Specifically, RADAR maintains a database of \((x; y; z; SSi)\) values, where \(x; y,\) and \(z\) are the physical coordinates of...
the training sample, and $SS_i$ is the signal strength measurements from the $i$-th access point (AP) for $i = 1; \ldots; n$. Each measured signal strength vector is then compared against the database, and the coordinates of the best matches are averaged to give the solution. The RADAR approach offers two advantages: it requires only a few base stations and uses the same infrastructure provided by a building's general-purpose wireless networking. Bahl and Padmanabhan (10) report localization accuracy of about 3 meters of their actual position with about 50 percent probability. However, significant changes in the environment, such as moving metal file cabinets or large groups of people congregating in rooms or hallways, may necessitate reconstructing the predefined signal strength database or creating an entirely new database. They discuss the problems of localizing an object in multiple floors and changing environmental conditions, as well as tracking of moving users. The error distance for tracking a mobile user is at most 3.5 meters. Niculescu and Nath (8) introduced a location-sensing algorithm that works on simple geometric principles of Euclidian geometry concerning triangles and quadrilaterals. The information of the landmark locations is slowly propagated towards the nodes that are further away, while at the same time closer nodes enrich this information by determining their own location. Three variations of this algorithm are presented: 'DV-hop', 'DV-distance', and 'Euclidian'. In 'DV-hop', which is a completely ad-hoc approach, the nodes that somehow know their position, referred to as anchors, flood their location to all nodes in the network? Each unknown node records the position and the number of hops to at least three anchors. Whenever an anchor 1 infers the position of another anchor 2, it computes the distance between them, divides that by the number of hops and floods this average hop distance into the network. Each unknown node uses the average hops distance to convert hop counts to distances, and then performs a triangulation to three or more distant anchors to estimate its own position. 'DV-hop' works well in dense and regular topologies, but for sparse or irregular networks the accuracy degrades to the radio range. The 'DV-distance' method is similar to the DV-hop one, with the difference that distance between neighboring nodes is measured using radio signal strength, and is propagated in meters rather than in hops. Finally, the 'Euclidean' method works by propagating the true Euclidean distance to the landmark, so this method is the closest to the nature of GPS. This method uses simple properties of quadrilaterals. For instance, using 5% landmarks, Niculescu's algorithms have an error of approximately 25%, 45% and 55% of the transmission range, for the 'DV-hop', 'DV-distance' and 'Euclidian' algorithms, respectively. However, when the number of landmarks increases to 20%, the algorithms perform at approximately 12%, 25% and 12% of the transmission range. Saverese et al. (11) proposed another approach that is closest to the approach presented in this thesis. Their system employs a distributed algorithm that determines the position of nodes in an ad hoc network in two phases, namely the 'startup' and 'refinement' phase. In the 'startup' phase, landmarks broadcast their location among all nodes in the network, and hosts estimate their position by triangulation. These estimations are rough approximations and are improved in the second phase of the algorithm. The 'refinement' phase proceeds in iterations. At each iteration, each host broadcasts its position estimate, receives the positions and the corresponding range estimates from its neighbors, and computes a least square triangulation solution to determine its new position. After a number of iterations, when the position update becomes small, the refinement phase stops the final position is reported.
The authors report that their algorithm is able to achieve position errors of less than 33% of the radio range, with 5% range error, 5% anchor nodes and an average connectivity of 7.

2.1.4 GPS

The most widespread and reliable localization system is the Global Positioning System (GPS), which is constituted by a constellation of 24 satellites (12). It uses these satellites as reference points to calculate the position of an object with precision of meters. Substantially, GPS gives to each cell-meter of the planet a unique address. It is categorized as a satellite systems of auto-localization, since each user must have a special GPS appliance. Moreover, the optical contact with at least 3 satellites is essential. GPS’s precision is better than 10 meters for military use and 100 meters for commercial use. Recently, the differential GPS (Differential GPS) has been developed, that improves considerably precision to 2-5 meters. It uses a network of stationary GPS receivers to calculate the difference between their actual known position and the position as calculated by their received GPS signal. Its disadvantages are that it needs a land station and has low degree of renewal. One of the basic disadvantages of GPS with respect to other systems of localization is that the appliance that is needed is quite expensive, bulky and energy consuming. Consequently, it is difficult to incorporate in other appliances. Additionally, to obtain less than 10 meters accuracy, GPS needs to take precise measurements from at least 3 satellites, which is difficult to be achieved in big cities with abundance of buildings. Consequently, an important disadvantage is that it cannot work in internal spaces, such as buildings.

2.1.5 Ekahau Positioning Engine

The Ekahau Positioning Engine (EPE) (13) is a well-known real-time location system based on signal strength calibration. Ekahau is capable of pinpointing Ekahau T201 Wi-Fi Tags, laptops, PDAs and other Wi-Fi enabled devices, with floor-, room-, and door-level accuracy. It uses a proprietary protocol to communicate signal strength readings to the server. The specifications of the system report that it locates a wireless device with up to 1-meter average accuracy (indoors, three or more overlapping access point signals) and it works with any standard 802.11 infrastructure and allows selection of contributing access points. Its server is Java-based, and runs on Windows or any type of UNIX with a working Java virtual machine, which is network independent software architecture. It is compatible with all major 802.11b, 802.11g and 802.11a WLAN adapters. In order to assess its suitability to the purposes of the present work, the Ekahau Positioning Engine has been experimentally tested at the Human – Computer Interaction Laboratory of ICS-FORTH. The results of the experiments lead to the conclusion that the accuracy of Ekahau is much lower than reported in the system’s specifications. In particular, if there are less of four APs in the environment, a wireless device can be located with up to 4-meter average accuracy. The accuracy of the system is enhanced considerably when there are more than seven APs in the environment.
Moreover, Scott Gifford (14) has tried Ekahau at the University of Michigan and found that although the tags are available at a moderate cost, the battery lifetimes are very short (approximately 1 day with location updates every second). The network was sufficiently dense that no additional infrastructure was required. In these experiments, it was found that the laptop had a median error of 2 meters, and was located in the correct room 86% of the time, and the PDA had a median error of 3.2 meters and was placed in the correct room 72% of the time.

2.1.6 NearMe

NearMe (15) is an 802.11-based location system that performs self-mapping (i.e., a location system which can build the radio map as the system is used). NearMe is a service designed to determine when two devices are in proximity. It allows 802.11 Access Points (APs) to be associated with physical places (e.g., 2nd floor copy room) or resources (e.g., duplex printer). NearMe uses radio traces to build a neighborhood graph of which APs are near each another out to eight hops. Although NearMe does not estimate absolute locations, it does estimate ranges between APs based on the traces and the neighborhood graph.

2.1.7 E911

The US Federal Communications Commission's E911 telecommunication initiatives require that wireless phone providers develop a way to locate any phone that makes a 911 emergency call (16). E911 is not a specific location-sensing system, but has led many companies that are developing a variety of location systems to determine a cellular phone’s location. Location-sensing systems developed to comply with the E911 initiatives can also support new consumer services. For example, a wireless telephone can use this technology to find the nearest gas station, post office, Movie Theater, bus, or automated teller machine.

Data from many cellular users can be aggregated to identify areas of traffic congestion. To comply with E911, vendors are exploring several RF techniques, including antenna proximity, angulation using phased antenna arrays, lateration via signal attenuation and time-of-flight, as well as GPS-enabled handsets that transmit their computed location to the cellular system. To meet the FCC requirement, positioning must be accurate to within 50 meters for 95 percent of calls with receiver-based handset solutions such as GPS, or to within 300 meters with network-transmitter-based approaches.
2.1.8 Collaborative Location Sensing

Cooperative Location-sensing (CLS) is a location sensing system that employs the peer-to-peer paradigm and a probabilistic framework to estimate the position of wireless-enabled devices in an iterative manner, without the need for an extensive infrastructure or time strenuous training (see 9.1.1). CLS can incorporate signal-strength maps of the environment to improve the position estimates. Such maps were built using measurements that were acquired from APs and peers during a training phase. CLS uses a particle-filters-based framework to model theoretically CLS. It also employs two real-time criteria for signal strength signature comparison, namely, confidence interval based and percentiles-based criteria. Depending on the comparison criteria, distance estimation varies from 1.8m to 2.5m (17).

2.1.9 Robot Localization

Research efforts leading to the development of two robot localization systems are reported in (18). These systems use an infrastructure of IEEE 802.11 access points. Based on the signal strength data, gathered by wireless cards at various predefined points of an indoor environment, they build a signal strength map of the environment. They locate a wireless device by using this map and applying a probabilistic inference of the position. Their first system is based on a two-step process. In the first step, a host uses a probabilistic model to compute the conditional probability of its location in a number of different locations, based on the received signal strength from nine APs. The second step of the algorithm exploits the limited maximum speed of mobile users to refine the results of the first step and reject solutions with significant change in the location of the mobile host. Ladd et al. report that 77% of the time, their stationary hosts can be located with at most 1.5 meters of error using 9 APs, whereas, in case of mobility, 64% of the hosts can be located with at most 1 meter of error using the same number of APs. The second system, which is based on the first one, uses a topological map of the environment instead of a grid-based map, as well as probabilistic techniques to achieve more accurate location results. Ladd et al. report that in all cases, they determined the correct cell in at least 70% of trials.

2.1.10 Summary of Location Sensing options

In conclusion the available options to provide location specific information can be summarized as manual, infrared, RFID(short), RFID(long), GPS and WLAN.

*Manual* is the case where the user inputs manually a code to indicate his position. Obviously the advantages of this approach are that it is inexpensive, easy to implement, and allows you
to spare all CPU processing cycles into content creation and usability. The disadvantage is that the user has to find the code numbers and since most sites do not possess content for every object, user cannot discern between which items have additional content unless they get close enough to each item and read the content number label.

*Infrared* is the case where Infrared triggers are placed in each area/room of the site. The advantages of this approach is that it is based on a simple technology with low energy requirements, it is inexpensive and since it is based on line of site it ensures a high level of accuracy. However its “line of site”-strength is also the main weakness of this approach since in a crowded site sensing the triggers can be difficult.

RFID represents the proximity-based scenario where RFID passive tags are placed in each area/room of the site and the user gains an awareness of each physical item when he approaches it. The advantages of this solution are that it has a low energy requirement and that is cheap. The disadvantage is that the user needs to be within 1-4 inches away from the tag in order to receive the proximity signal.

The following table includes a summary of the aforementioned approaches in terms of cost (not taking account development cost), line of sight, accuracy, indoor functionality and energy.

<table>
<thead>
<tr>
<th></th>
<th>Manual</th>
<th>Infrared</th>
<th>RFID (short)</th>
<th>RFID (long)</th>
<th>GPS</th>
<th>WLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase cost</td>
<td>None</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>Very High</td>
</tr>
<tr>
<td>Line of Sight</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Accuracy</td>
<td>10m</td>
<td>5-25cm</td>
<td>3-5m</td>
<td>~2m</td>
<td>3+ m</td>
<td></td>
</tr>
<tr>
<td>Functions indoors</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Energy Required on PDA</td>
<td>None</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1: Comparison of location sensing options
2.2 Ambient Information Systems

2.2.1 Position-Based Interaction for Indoor Ambient Intelligence Environments

A project that greatly inspired this thesis was the work of Blache et.al (19) in Position-Based Interaction for Indoor Ambient Intelligence Environments. In their work, the authors created a system that provides ambient proximity services (such as a virtual kiosk) using a personal communicator application running on the user’s PDA. The system’s requirements as underlined by the authors are the following:

- The selection of services should by default occur as an implicit command triggered by proximity detection (it may also of course occur by explicit selection from the user).
- The provisioning and operation of services should not require any manual operation or a pre-configuration of the personal communicator.
- The personal communicator should have dual connectivity, with wide-area wireless network for distant service provisioning on the one hand, with other ambient environment devices via WLAN, WPAN or other more means such as inductive coupling using RFID on the other hand.
- The personal communicator should act as a gateway to a chosen service infrastructure, and ideally should make it possible to interoperate between several such infrastructures (e.g. JINI, UPnP, OSGI, SLP, etc.).
- The location infrastructure should be independent of both the location-sensing technologies and the applications, and should make it possible to cooperate with all of these.
- This location infrastructure should be scalable from small-scale location management (ideally a few centimeters for close-range proximity detection) up to large scale location management by interfacing to cellular-network based or third party location-determination systems using standard protocols (e.g. MLP).
- The location infrastructure should make it possible to maintain information about a plurality of fully mobile devices (personal communicators, wearables, etc.) and about other ambient devices that may move from time to time (e.g. printers, coffee machines, etc...).
- The location infrastructure should make it possible to incorporate a rich set of model information about the environment (fixed features of the environment, points of interest, movable objects, other mobile devices tracked by the system,
This space modeling information should be obtainable with the assistance of interactive tools from architecture CAD systems.

In order to complete, these requirements they have created a JAVA based application that uses an SVG representation of the environment, an RDF schema for representing the data metaverse and a web service based mechanism for interacting with digital artifacts such as computers that are mounted on the wall of the exhibition site. For the proximity layer they used the 802.11 wireless infrastructure and RFIDS for close range detection in order to further enhance the position estimation of the wireless location-sensing system.

Their work motivated us to move one-step further and bring to our metaverse information about non-digital artifacts. The context data for the non-digital artifacts are recorded using two applications. The mobile administration system records information on the artifacts position and the content management application records meta-data on the artifacts such as the title and the description. It also prompted us to use a vector-based representation for the environment.

### 2.2.2 Bus Mobile

One of the earliest Ambient Information Systems was Bus Mobile (20), shown in Figure 1. Bus Mobile is designed to give users a sense of how much time is left until popular buses reach their chosen bus stops (21). It includes six tokens, each representing a bus line. At 24 minutes before a bus arrives at its stop, its token lowers 24 inches below the mobile’s skirt. It rises one inch every minute until it disappears under the skirt when the bus has left the bus stop. A heuristic evaluation of the original Bus Mobile found several usability problems. At first, the Bus Mobile did not properly use notification. The major notification happened at 24 minutes when the bus token lowered from zero to 24 inches. This action interrupted users rather than making them aware of the approaching bus. The most important notification needed was when the user actually had to leave for the bus stop, when the token was about 5 inches from the top. This event was not distinguished in any way from other events, making it essentially change blind to users. This project inspired the communication mechanism for the location-sensing client. We implemented it as an
asynchronous event based service that does not interrupt the applications current flow but gently raises a callback whenever a position-changed event has occurred. From there on, it is the user interface’s responsibility to alert the user in a non-interruptive manner.

2.2.3 Dangling String

Natalie Jeremijenko created the "Dangling String" (22), an 8-foot piece of plastic spaghetti that hangs from a small electric motor mounted in the ceiling. The motor is electrically connected to a nearby Ethernet cable, so that each bit of information that goes past causes a tiny twitch of the motor. A very busy network causes a madly whirling string with a characteristic noise; a quiet network causes only a small twitch every few seconds. Placed in an unused corner of a hallway, the long string is visible and audible from many offices without being obtrusive. It meets a key challenge in technology design for the next decade: how to create calm technology. At first, it creates a new center of attention just by being unique. However, this center soon becomes peripheral as the gentle waving of the string moves easily to the background. That the string can be both seen and heard helps by increasing the clues for peripheral attunement. The dangling string increases our peripheral reach to the formerly inaccessible network traffic. While screen displays of traffic are common, their symbols require interpretation and attention, and do not peripheralize well. The string, in part, because it is actually in the physical world, has a better impedance match with our brain’s peripheral nerve centers. This project can be categorized as the most simplistic approach for the creation of an ambient information system. It aims at showing us that ambient information systems are all around us and they are not hidden from our sight. In order to implement such systems we do not need to resort to complex mechanisms, we simply need to imagine ourselves in the position of the user, as he exists in surroundings and determine what his needs are and how we can fulfill them in an unobtrusive manner. The goal of calm computing inspired the notion behind the requirements analysis in this project.
2.2.4 Digital Family Portrait

The concept of Digital Family Portrait (DFP) (23) is a qualitative visualization of daily life that is meant to take its place alongside other family pictures on the fireplace mantle. Leveraging a familiar household object, the family picture frame, DFP leaves the photograph held by the frame untouched, while the frame itself is populated with icons that are updated daily. These icons represent, not only the current conditions in the remote location, but they also carry a history of previous conditions. This history is represented so that possible developing trends might be more readily seen.

As seen in Figure 4, data about the resident are collected in their home. This data is used to create a display that is updated once a day. This display is presented to the intended audience, the extended family living in a remote home, as well as presented locally to the resident. This qualitative visualization in many ways parallels the natural activities undertaken to keep an “eye out” for your Mom if she lived next door. Rather than glancing out the window from time to time to see if “things seem O.K. at Mom’s house,” by noting that Mom has picked up the morning newspaper, the family member would glance at the digital family portrait to be reassured that “life is proceeding as usual at Mom’s.” In this way, the digital family portrait is meant to be intentionally engaging and provide some of the back-story that goes missing when people are geographically distant. This back-story lowers
the startup threshold of a conversation by providing a context in which new conversations can be started. This project inspired the desktop version of the Level Administration. Through that system, we are able to monitor the user’s activities inside our Ambient Information System and know where each user is and what the user is doing for example which point of interest he is looking at and for how long. Seen from a data-mining point of view, this aspect of information can help the site administrators to organize their physical artifacts more efficiently or create better content for those artifacts that are not very popular.

2.2.5 Infocanvas

Figure 5: Infocanvas

The Infocanvas (24) approach bears much resemblance to the InformativeArt (25) project that uses LCD panels as ambient displays to depict email traffic flow or web site usage. The InformativeArt displays, however, present only one, predetermined source of information and modify particular aspects of predesigned abstract art. Conversely, allowing end-users to express their own creative designs is a key part of our effort. There are two steps in painting information on the InfoCanvas: specifying the information to be represented and visually mapping the data to the canvas. The first step in painting information is to identify its source. Currently, InfoCanvas accepts data from email accounts and web pages. Certain “rules” can be applied to each source for extracting and interpreting its data. Rules applied to an email source can monitor the number of messages, arrival of new messages, new messages from a particular person or with a particular keyword in the subject, and when a reply arrives. Web pages can be monitored for any general update in their content, the presence of a keyword in the content, or a value can be extracted that is associated with a keyword using the KPS Mining Algorithm (26). Multiple rules can be assigned to a source, with each rule having its own unique representation on the canvas. In representing the stock market, we identify a page from Yahoo as our data source that presents detail on the Dow Index by copying the URL into a source definition dialog box. Then we set an “Extract value associated with keyword” rule and give the keyword “Change”. This will find the first
The Information Percolator (27) is consisted of 32 transparent tubes filled with water, covering an area of roughly 1.4m by 1.2m (placed on a 20cm base). Inside these tubes, small bursts of air bubbles are released and allowed to rise to the top of the tube. Release of air within each tube is regulated by a micro-controller as described below. By releasing air for a specific (and precise) duration, a specific size of bubble burst can be created. Each such burst can be viewed as a pixel that rises up the tube. With proper timing and coordination, a series of such airbursts across the tubes can form images, which scroll from the bottom of the tubes to the top (note that the display must scroll and cannot be randomly accessed). Overall, the display provides 32 by approximately 25 pixels of scrolling display area. However, the character of the display is significantly different from most pixel-based display devices. For example, the boundaries of the pixels involved tend to be fuzzy and in flux – reflecting the chaotic nature of the bubbles that create them. Further, as pixels rise through a display they disperse, spreading out along each tube, so that by the time they reach the top of the tubes they have typically “dissolved” into the pixels above and below them. This creates an aesthetically interesting effect in which the image being displayed dissolves as it rises away from its point of origin, emphasizing its ephemeral nature. Finally, a range of non-
linear effects tends to make the details of the display chaotic in nature. For example, bubble speed is dependent on size – large bubbles typically rise slower than small ones (because they have more surface area). This causes smaller bubbles to catch them from below, making them larger and slower, which causes them to collect more small bubbles, etc. Overall, all these effects, while reducing the clarity of the display in the conventional sense, make it interesting to look at. Since an important part of the aesthetics of the display is found in its dynamics, readers are encouraged to view a short video clip of the display that is available on the web at: http://www.cs.cmu.edu/~hudson/bubbles/. A conceptually related display – the Linear Rainfall fountain – is described in (28). This device also uses water to create a pixelated display. However, rather than air bubbles rising in water, it employs streams of falling water. Each of many small streams of vertically dropping water can be turned on or off under computer control to form a set of falling pixels. Because of the speed of the falling water, the Linear Rainfall display tends to be faster paced making it less suitable for ambient information display.

2.2.7 Irwin

Irwin (29) monitors Internet information resources and alerts the user of updates and modifications. Irwin consists of a set of hyperactive tools, small reusable programs that can run simultaneously and share information. The central tool in Irwin handles the visualization and user interactions, while the remaining tools process the information from each resource and update the visualization tool when important changes occur. The information resources monitored by Irwin can include email folders, Usenet newsgroups, Web pages, and weather data. The email and newsgroup hyper tools monitor the messages in a folder or group, alerting the user when new messages arrive. The Web tool summarizes headers, lists, and hypertext links on a Web page, allowing the user to monitor news wires and hotlists. The weather tool monitors the weather conditions and forecast for a given city. When users initially start Irwin, it tries to determine their informational interests by examining their email files and Web bookmarks. Since users are not required to answer a list of questions at startup, they are often more likely to use Irwin. The user can then configure both the information monitored and the way in which it is displayed after getting used to the power and flexibility of Irwin. To show both an overview and details simultaneously, Irwin uses a confluent zoom display. A confluent zoom provides multiple views of a resource simultaneously with each view having a different level of granularity. With a confluent zoom, the user can see overview and details with minimal effort, and the context of the detailed views is maintained at all times by the coarser views. In its confluent zoom, Irwin employs an auditory cue, an icons view, a navigation bar, and several textual views. In the icons view (the leftmost view), the state of each resource is given by an icon. When the resource changes, the appearance of the icon changes and an auditory cue is played. If user selects an icon, the other views are updated to show information about the corresponding resource. A navigation bar (second from the left) shows a syntactic encoding of the resource contents and can be used to select the messages displayed in the textual views (the two rightmost views). Users can configure the orientation and placement of the icons with respect to the
other views and can even choose to hide the other views until some event happens, e.g. an icon is clicked.

2.2.8 Kadinsky – Aesthetic Information Collages

An aesthetic information collage is designed to be a type of ambient information display in a decorative object (see also (27)). Normally, the central imperative for an information display is to effectively convey information - if it can also be aesthetically interesting, that might be an added bonus. The Kandinsky system is an experiment in turning this imperative upside-down. We envision this system being used in a home or office setting to produce the equivalent of a painting or poster hanging on the wall. (With current advances such as low cost Organic LED displays, this may be widely practical in only a few years.) Like other images we hang on the wall, we would typically choose this display primarily because of its aesthetic properties. To these aesthetic properties, we wish to add the ability to convey ambient information - information that we may wish to be aware of, but that should not necessarily demand our attention.

Kandinsky (30) works by composing images representing information items to be displayed into a collage in a way that tries to maintain certain aesthetic properties. If necessary, the system will also create images to represent information given to it in only textual form.
2.2.9 Kimura

![Kimura Architecture](image)

Kimura (31), separates the user’s "desktop" into two parts, the focal display on the desktop monitor, and the peripheral displays projected on the office walls. As shown From Kimura's point of view, a working context is the cluster of documents related to a general activity, such as managing a project, participating in a conference, or teaching a class, as well as the collection of on-going interactions with people and objects related to that activity. Any cluster can have numerous documents, including text files, web pages, and other application files, that have been used in the course of the activity, plus indications of ongoing activity such as email messages without replies and outstanding print jobs. Kimura automatically tracks the contents of a working context, tagging documents based on their relative importance. As in previous systems, such as Rooms (32), users demarcate the boundaries of working contexts manually, as this operation is lightweight from the user's perspective and error-prone if left to the system. One contribution of this work is creating and using logs of activity to support both awareness, and resumption, of background tasks.

2.2.10 Lumitouch

LumiTouch (33) focuses on communicating emotional content in addition to presence detection. LumiTouch is designed to be an asymmetric, bi-directional channel of communication. It is similar to the symmetric, haptic approaches embodied in inTouch (34), and handJive (35). Until the recent introduction of web enabled devices, this particular combination of asymmetric (touch-to-light) and bi-directional data direction has been rarely
The LumiTouch is designed to be used like the picture frame, and remain visible in everyday life. The use of light as both an ambient representation and active data transmission allows the user’s attention to transition between passive and active. Figure 8 shows that this device is also in the relatively unexplored gray region between ambient and direct communication. When a user is in front of her LumiTouch, the corresponding LumiTouch will emit an ambient glow to indicate her remote presence. This additional background information helps people figure out if there is a recipient on the other end, or also if it is a convenient time to increase interaction levels (e.g. start active communication mode or use an alternative, like a phone).

When a user picks up the picture frame and squeezes, the feedback display area illuminates to show that the picture frame has been squeezed. The display colors are transmitted over the Internet to the corresponding remote LumiTouch. The display varies depending on the squeeze attributes (where, how hard, and how long the user squeezed). While the recipient party can simply enjoy this display, she has the option to pick up the frame and squeeze back a response to the first user, and begin an interactive exchange.

**Figure 8: LumiTouch**
2.2.11 Notification Collage

The Notification Collage (36), illustrated in Figure 9 and the accompanying video figure, follows the metaphor of a bulletin board containing a collage of randomly positioned and possibly overlapping visual elements. Using various client programs, group members can post a variety of media elements to the NC. Upon receipt, NC reconstructs these as discrete visual entities and randomly places them onto the left side of the vertical bar that splits the board. Overlap is allowed, and new items are always positioned on top. The author’s chose this metaphor for several reasons. First, the overlap of items inherent in a large collage acknowledges that there may be a large number of information fragments; too many to fit neatly on the display. Second, collages are customarily used to present unstructured information comprising diverse media, conceding that awareness information comes in many forms.

2.2.12 Scope

The Scope (37) is an information visualization designed to unify notifications and minimize distractions. It allows users to remain aware of notifications from multiple sources of
information, including e-mail, instant messaging, information alerts, and appointments. The design employs a circular radar-like screen divided into sectors that group different kinds of notifications. The more urgent a notification is, the more centrally it is placed. Visual emphasis and annotation is used to reveal important properties of notifications. Several natural gestures allow users to zoom in on particular regions and to selectively drill down on items. We present key aspects of the Scope design, review the results of an initial user study, and describe the motivation and outcome of iteration on the visual design.

By default, the Scope positions itself in the lower right-hand corner of the display, fitting in a 180x180 pixel region as shown in Figure 10. This is the peripheral awareness or low-LOD mode: in this modality, items reveal only the most important properties, which are designed to be maximally distinct. These are newness, ToType, and overdue status. Recall that urgency of items is still easily deduced from the item’s location along the radius of the Scope. This allows users to glance at the Scope and determine immediately whether new (pulsing) items have arrived, particularly in the high-urgency zone. If the user moves the mouse pointer into the Scope window, it responds by changing into the active interaction mode: the Scope is popped to the top of the window stack, the window doubles in size, and the items are rendered in high-LOD mode, displaying all of the properties described above. Users can now gain additional information at a glance, or obtain more details from an on-hover tooltip. We believe that these two levels of detail provide a fluid way for the Scope to achieve both unobtrusiveness and useful richness of display.
3 DESIGN

3.1 Overview

According to Pousman et al (2), an Ambient Information System is one that possesses the following characteristics:

- Display information that is important but not critical.
- Can move from the periphery to the focus of attention and back again.
- Focus on the tangible; representations in the environment.
- Provide subtle changes to reflect updates in information
- Are aesthetically pleasing and environmentally appropriate

From the aforementioned properties, the same authors have concluded to the following design dimensions for Ambient Information Systems:

- **Information capacity** represents the number of discrete information sources that a system can represent. Some systems are capable of displaying a single piece of data such as the current price of a stock index. Others can display the value of 20 (or more) different information elements on one screen.

- **Notification level** is the degree to which system alerts are meant to interrupt a user. Matthews et al subdivides notification level into five categories: ignore, change blind, make aware, interrupt, and demand attention. For our analysis we adopt those categories but replace the lowest level of system alert function, ignore (a degenerate case) with user poll.

- **Representation fidelity** describes a system’s display components and how the data from the world is encoded into patterns, pictures, words, or sounds. Some systems reproduce the information being monitored in a very direct way, while others are much more abstract in their representation.

- **Aesthetic emphasis** concerns the relative importance of the aesthetics of the display. Some system designers seek to build displays and artifacts with sculptural or artistic conventions. For these systems, being visually pleasing is a primary objective. Others however place relatively little focus on aesthetics and typically focus more on information communication ability.

However, designing a collection of building blocks that represents distributed information cannot be limited only to the principles of AMIS, it must also follow principles of pervasive computing.
Pervasive computing is another step beyond mobile computing. Whereas mobile computing may have focused on only a few nodes that may be moving and interacting with each other, pervasive computing seeks to weave computing into everything and have it all work together. Satyanarayanan identifies four additional research areas in pervasive computing beyond those of mobile and distributed computing (38). Those areas are:

- Effective use of smart spaces
- Invisibility
- Localized scalability
- Masking of uneven conditioning

Smart spaces are physical locations that are augmented with computational resources to enhance the human experience. These resources may be sensors that automatically detect the presence of particular people and their location or perhaps measure a physical property of the room such as light or noise. Alternatively, the resources may be computers and displays that augment the user experience by providing more information to the users present in the room.

The Access Grid, developed by Argonne National Laboratory, is one example of a smart space. Cameras in the room film what is currently going on in the room and microphones record conversations in the room. This data is then sent to participants at other sites, called nodes, which are participating in the conference. Ceiling mounted projectors are able to display a large format display the video streams on a wall to create the impression of being in the same room with participants at other access grid nodes. There are some customized applications to allow for distributed presentations and data visualization. A set of computers manages the interfaces to the entire system and provides features such as highlighting the current speaker and echo cancellation.

Effective use of such spaces means that handheld, notebook, and other forms of computers automatically are able to interface with the environment. For instance, if a presentation were taking place, a user watching the presentation with a notebook computer would be able to automatically download the slides of the current presentation and view them without any interaction from the presenter.

Another goal is invisibility, or allowing the users to interact with the system without knowing it. Currently most human-computer interaction is invasive; when a person is using a computer they are very aware of it as we must interact with computers through methods other than those used to interact with people. In a pervasive and ubiquitous system, the user should not have to change their method of interaction to interface with the computer systems in the room. This is especially important because if everything in the room has a computer chip in it, changing the method of interaction for each device can be quite time consuming and inconvenient.
The system should also be able to easily scale. For instance, the system may work well when there are only three people present in a room, but how will the system work when it is expanded for use in a shopping mall, airport, or stadium?

Finally, there is no way to guarantee the homogeneity of resources across all spaces. At work, a user may have a 100Mbps uplink to the Internet while at home he may be limited to 128Kbps. The applications in the environment must know how to handle both of these situations and adapt to them. Uneven conditioning is the result of having differing numbers and quality of resources depending on location. In most current applications, a drastic difference in the quality or availability of resources will cause an application to cease functioning. For these applications to adapt some form of user intervention is required. An example of such a situation is viewing video over the Internet; when the user requests the video file they must also select what speed of Internet connection they have. If for some reason the throughput from the video server degrades, because of congestion or other reasons, the video may pause and become choppy. To adapt the user must return and manually select the lower quality stream, which has the drawback that when more bandwidth is available, it will not improve the quality of the video.

Designing a system to properly address these issues is a daunting task and is not our main goal. Instead, the implementation of the building blocks takes a philosophy similar to that of many of the command line utilities for UNIX, do only your job and try to do it very well. Then using these simple tools it is possible to build complex systems to handle complex interactions. This also greatly simplifies the design and debugging of the system.

In addition to the aforementioned characteristics of Ambient Information Systems, the blocks are designed to be a system for context awareness, similar those described in chapter 2, building on the strengths and overcoming the weaknesses of such systems. Eventually, it is hoped, that it will be just one component of a much larger, vibrant, pervasive and ubiquitous computation environment.

To summarize, we focus on the following smaller issues, which are highly related to the more general issues of pervasive computing research.

- Information Capacity
- Notification Level
- Representation Fidelity
- Aesthetic emphasis
- Modularity
- Scalability
- Extensibility

Each of these features will be described in brief in this chapter, with additional information available with the discussion of the implementation of the system.
3.2 Information Capacity

The issue of information capacity is not a trivial one when dealing with ambient information. Our natural environment is filled with an enormous amount of information sources, yet not all of them have the same importance as the user can only operate within his own context, he can only perceive the environment as his five senses tell him to do so and he can only understand knowledge based on his education and external stimulation. In this collection of building blocks, we tackle with the amount of information in two ways:

First, we want to be able to show to the user more spatial information than he can perceive with his own sight using a floor plan bird’s eye view of the space the user is occupying and the location of all significant information sources that exist around him,

Secondly when the user approaches an information source we use an interactive cascading slide show to present each aspect of information, the content manager application has associated with the specific source.

The first goal is achieved through the Map Viewer component, which is inspired by video games, both in terms of implementation and aesthetic emphasis, and allows the user to rotate, zoom in, zoom out and scroll visually on the floor plan. It also displays his own position as reference in order to allow him to navigate thought the various artifacts that exist around him. The interaction between the user and the Map Viewer is done by tapping the pen on the PDA’s screen.

Furthermore, the desktop version of the Map Viewer is used in the Content Manager application in order to allow the administrator to fine tune the position of each information source and to associate further information to each source simply be viewing spatially the position of each artifact and selecting it for editing purposes.

The second goal is achieved through the cascading slide show, which is explained in further detail in our Museum e-guide Case Study. Each information source is comprised by a collection of media such as video, audio, text and images. When the user approaches an information source, the application detects the change in the user’s position and presents him with the new data that exist in front of him.
3.3 Notification Level

Notification level is another crucial matter since it can be very frustrating for a user, being surrounded with all these information sources, to be obtrusively notified whenever some instance of information has changed or when as he approaches source information that source information demands his attention to show him it is content. In order to tackle with this characteristic we have taken a strategic approach. We acknowledge the existence of two very different, very opposite scenarios that exist when a user is wandering around an area.

The first scenario assumes that the user is in control, he roams around in his environment by viewing the Map Viewer, therefore knowing where each information source is located, and when he approaches the information source, he is notified of the source’s content and he is asked whether he would like to view it.

The second scenario deals with the novice user. Similarly, to the previous scenario, the novice user also uses the Map Viewer to perceive the position of each information source. However, he operates in a mode that we call guided tour mode. The Map Viewer does not only display the user’s position in the map but it also shows him the path he must follow in order to reach a certain information source. When he reaches that information source the Map Viewer function is disrupted and the application goes directly into the slide show mode.

To summarize, we claim that the existence of these two complementary scenarios are required for using and also abusing properly the characteristic of the notification level. Since our greatest goal is to make the user familiar with his surroundings in a manner that bests suits his own preferences without being able to detect them in real time but by allowing the user to state them in the beginning of the application. This aspect will be discussed in further detail in the Implementation Section and more specifically in the NavigationScheme field of the Database schema.

3.4 Representation Fidelity

Representation fidelity is another crucial matter since the highest the resemblance between what is shown in the PDA screen to what exists in the real world, the more intuitive the whole procedure becomes for the user. In order to resolve the representation fidelity issue we have implemented the Map Viewer application using DirectX, inspired by the video game scene. Each information source is represented using vertices that are stored within a vertex buffer and are consisted of x,y,z axis info. This allows us to scale our information source capabilities to pretty much real life level. For example, instead of representing a cafeteria information source using a circle and writing a text label on top of it saying “cafeteria”, we can simply put a vertex based coffee mug on it and exploit all the vertex based assets such as scaling, rotating, scrolling with minimum computational overhead since all of the above
transformation are simple multiplications carried on the transformation pipeline. Furthermore, they are hardware accelerated meaning that the GPU is responsible for dealing with them, leaving the CPU with more computational cycles at the disposal of other functionality.

The e-guide application tackles the representation fidelity problem using images, videos, text and audio associated with each information source. The content is assigned by the content manager and is distributed to each PDA using the synchronization service. It is the responsibility of the content manager to associate the appropriate content to each information source, hence augmenting the representation fidelity as perceived by the user.

3.5 Aesthetic emphasis

Aesthetic emphasis is dealt as a shortcoming of the video game scene inspiration. More specifically, the use of vertices to display objects in 3D virtual environments is the most dominant approach in creating life-like objects. The intuition behind this is the notion that says one can display anything using a collection of triangles. Vertices are used to create triangles and combinations of triangles are used to create polygons, polygons are used to create objects with higher complexity and so on. In order to deal with the aesthetic emphasis we allow the developer who is injecting objects into the Map Viewer component to use mesh files. Mesh files are collection of geometric information for an object. Mesh files can be created by any 3D Editor such as Blender and once injecting into the pipeline the 3D engine behind the Map Viewer is responsible for analyzing all vertex information, all material and all lighting information. This can result in the creation of an “as to close to reality approach as possible”. However, the aesthetic emphasis matter cannot be dealt with objective terms since each person has his own set of preferences and likings to what he thinks is beautiful and aesthetically pleasing. By any means at all, we do not claim that the game engine behind the Map Viewer component can result to high aesthetic emphasis, however we claim that it is an appropriate environment where further studies can be made on what pleases the user the most.

3.6 Modularity

An issue that comes up as a corollary to the system being scalable is that of modularity. Modularity is the ability of the system to run with only some of the parts of it enabled, or with different versions of common parts. Our micro architecture is designed from the ground up to be modular and flexible. The runtime is broken up into a variety of components, which are described in detail in chapter 4. The only component of the system that is required for every instance is the database. The database contains the appropriate data structures that hold information on the user’s surroundings.
An example of where modularity is beneficial to the system is on a handheld device. If the device does not provide context information to remote sources, then it will not have any context providers and thus the runtime will not load the module for providing context. This saves memory and CPU cycles for other tasks on the handheld device.

Another example of a benefit of modularity is the ease with which components may be swapped in and out for different implementations of the same component. It may be the case that a system provides a context nugget to other context consumers, but it may not have the need or ability to provide the full set of services. Thus, it is possible to switch the standard context provider module for a more simplified one. In addition, the modularity property component was proven worthy during the implementation of the building blocks since a new version of the Positioning Engine became available and by merely switching from one positioning engine to another we were able to encapsulate the new positioning engines features.

3.7 Scalability

There are two different aspects of scalability we focuses on; one is scalability in terms of computational power needed to run the system, the other is scalability in terms of number of systems participating in the environment. The driving force for cross-platform compatibility is also the driving force for developing a scalable system. In a truly pervasive environment, there is the need to integrate information from a huge variety of providers. Some providers may be running on a super computer, while others are on desktop computers, or handheld devices. Still other context providers may be running on embedded systems, such as an RCX brick.

Designing a system for the least common denominator would result in a profound lack of features. While designing only for high end systems would result in many devices being unable to take advantage of the infrastructure. Thus, our collection of building blocks takes a compromise between the two; it provides a moderate amount of functionality inherent in the system, but allows lower end devices to not implement all of the features of the architecture.

Communication bottlenecks and single points of failure heavily influence the other aspect of scalability, the number of devices that are using the infrastructure. It is for this reason that we seek to maximize peer-to-peer communication wherever possible. Context providers that provide notification operations can future reduce bandwidth usage as we have the flexibility to allow multicast dissemination of context. However, this functionality is not yet fully implemented.
3.8 Extensibility

It would be naive to think that the collection of building blocks could, in its current form, satisfy all of the possible requirements for a context aware system in a pervasive computing environment. It is for this reason that extensibility has been imbued at the very core of the system.

The micro architecture was designed to allow easy replacement of components with more advanced or refined components. The implementation of the chosen communication layer is bound to a specific platform, programming language, or even transports protocol. However, it uses the Socket paradigm, meaning that any platform, programming language or operating system can establish a connection to the specified port and exchange information. The communication layer is important since it is the artery for exchanging information. Using the communication layer the synchronization service is able to notify the mobile handheld devices that there have been changes in the content and that they should update it.

Aside from software engineering aspects, the building blocks can also be benefited from hardware upgrades. For instance, the Map Viewer is a hardware-accelerated application meaning that any change or improvement in desktop and mobile GPUs can significantly benefit the applications features.
4 IMPLEMENTATION

This section contains an analytical description of the path we followed in order to implement the collection of building blocks. It begins by providing an overview of the most dominant platforms that can be used to implement mobile applications, and then describes the underlying data infrastructure. Then, moves on to the description of each building block and finally describes the supporting tools that are created using the building blocks and compensate for the creation of an ambient information system.

4.1 Overview

.NET CF (39) and J2ME (40) are platforms for managed smart mobile clients. They are the fundamental technologies/platforms for the development of advanced mobile applications. Compared to their predecessors, micro-browser technologies such as WAP, smart clients

- Support rich user interfaces
- Leverage device extensions (e.g. GPS and bar-code scanners)
- Support more flexible integration and security schemes
- Reduce network traffic and improve transactional stability through on-device data storage.
- Java / .NET-Managed environments greatly improve developer productivity, application reliability, and mobile code security.

<table>
<thead>
<tr>
<th></th>
<th>.NET CF 2.0</th>
<th>J2ME CDC (Connected Device Configuration)</th>
<th>J2ME CLDC (Connected Limited Device Configuration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Requirement</td>
<td>Powerful, expensive</td>
<td>Cheap, pervasive</td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>high</td>
<td>medium</td>
<td></td>
</tr>
<tr>
<td>Market focus</td>
<td>enterprise</td>
<td>Consumer and enterprise</td>
<td></td>
</tr>
<tr>
<td>Language support</td>
<td>C# 2.0, VB.NET</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>Platforms</td>
<td>Pocket PC, Windows CE, Windows Mobile</td>
<td>Major mobile platforms except</td>
<td>All mobile platforms</td>
</tr>
<tr>
<td>Feature</td>
<td>5.0</td>
<td>Palm OS</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Byte code compatibility</td>
<td>Standard .NET CLR</td>
<td>Standard Java 2</td>
<td>Not compatible with J2SE or CDC</td>
</tr>
<tr>
<td>API compatibility</td>
<td>.NET Framework subset</td>
<td>J2SE subset plus standard/optional packages</td>
<td>Partial compatibility with CDC with additional standard optional packages</td>
</tr>
<tr>
<td>Native APIs</td>
<td>Platform Invocation; consistent across supported devices</td>
<td>JNI: device- and OS-specific</td>
<td>N/A</td>
</tr>
<tr>
<td>Development tools</td>
<td>VS. .NET 2005</td>
<td>Command line, vendor SDKs, CodeWarrior, WebSphere</td>
<td>Command line, vendor SDKs, all major JAVA IDEs</td>
</tr>
<tr>
<td>Specification process</td>
<td>Single company</td>
<td>Community</td>
<td></td>
</tr>
<tr>
<td>Service gateway</td>
<td>N/A</td>
<td>Run gateways as OSGi servlets; run gateway clients via vendor-specific SDKs</td>
<td>Run gateway clients via vendor-specific SDKs</td>
</tr>
<tr>
<td>Security model</td>
<td>.NET model</td>
<td>Full JAVA security manager</td>
<td>Formal OTA specification</td>
</tr>
<tr>
<td>Client installation</td>
<td>ActiveSync</td>
<td>Sync, download</td>
<td>Formal OTA specification</td>
</tr>
<tr>
<td>Life cycle management</td>
<td>N/A</td>
<td>OSGi for gateway apps, J2EE client provisioning specifications for generic clients</td>
<td>Included in OTA spec, works with J2EE Client Provisioning Specification</td>
</tr>
</tbody>
</table>

Table 2: Smart client development platform features
4.1.1 .NET Compact Framework

Designed for mobile computing, .NET CF is a lightweight version of Microsoft's .NET Framework. The .NET CF Common Language Runtime (CLR) runs standard .NET byte code applications. .NET CF contains a subset of standard .NET API libraries necessary for mobile application development. It runs only on Windows CE/Pocket PC/ Windows Mobile 5.0 - powered high-end PDAs.

Code written on the .NET Framework platform is called managed code. The term "managed code" refers to the fact that the Common Language Runtime (CLR) provides several assurances for such code:

- Bad pointers.
- Memory leaks.
- Strong type-safety.

Furthermore, in the .NET CF:

- 28 of the 35 desktop controls are supported in the .NET Compact Framework. These controls are enhanced for the size and performance requirements in the Compact Framework.
- Not all .NET properties, methods, and events are supported in the Compact Framework controls.
- Supports the same Win32 API Access mechanism as the .NET Framework but has a simplified data-marshalling layer.
- OLE DB Wrapper – Not supported in .NET Compact Framework.
- Datagrids – Not built in to the editing support as in the .NET Framework. The .NET Compact Framework can only bind Datagrids to a single table object. The .NET Compact Framework only displays rows from a single table, and these are not able to be expanded as in the .NET Framework.
- Infrared Data Association classes - .NET Compact Framework provides for making infrared connections and Web listening classes for HTTP services to devices. These are not available on .NET Framework.
- Infrared send/receive – The .NET Compact Framework includes new APIs that allow applications to send and receive information over the IR port. This feature is not present on the .NET Framework.
- Serialization - Due to performance considerations, .NET Compact Framework does not support binary serialization using BinaryFormatter, or SOAP serialization using SoapFormatter. There is, however, support for serialization to transmit object data using SOAP in XML Web Services.
- Web services - Although .NET Compact Framework applications can consume Web services, they cannot provide them due to the lack of a Web server.
- XML support - Scaled-down XML support is available. .NET Compact Framework applications will have full support for reading and writing XML using the reader and...
writer classes, but they will not have support for XSLT or XPATH as these are resource consumptive.

- Threads - .NET Compact Framework does not support priorities as in the .NET Framework.
- Web Services – Only client implementation is supported in the .NET Compact Framework.
- Win32 Calls - .NET Compact Framework does not support calls from Win32; you can make calls from Compact Framework into Win32. MessageWindows class does communication between Win32 and Compact Framework.

4.1.2 Overview of the J2ME

In the Java camp, the situation is more complex. J2ME contains standardized configurations and profiles designed to provide the best compromise between portability and performance for a range of mobile devices. Configurations support the Java core APIs. Profiles are built on top of configurations to support device-specific features such as networking and UIs. Each valid combination of configuration and profile targets a specific type of device:

- Profiles on top of the Connected Device Configuration (CDC) target high-end networked devices. Those devices have similar hardware capabilities as .NET CF devices. The CDC includes a standard Java 2 Virtual Machine so it can run standard Java byte code and utilize Java 2 Platform, Standard Edition (J2SE) libraries if desired. The CDC Personal Profile has roughly the same functionality as the older Personal Java environment.
- Profiles on top of the Connected Limited Device Configuration (CLDC) target low-end PDAs and small cell phones. The CLDC uses a small footprint VM that is not compatible with J2SE or the CDC.

Limitations of CLDC include:

- No floating-point support. Since many of the processors used in the target platforms for CLDC do not have floating point hardware, the virtual machine is not required to support floating-point operations. In terms of the virtual machine, this means that certain byte code operations are not implemented. This leads to the following coding restrictions:
  - Variables of type float and double and arrays of these types cannot be declared or used.
  - Constants of type float and double (i.e., 1.0, 2.0F) cannot be used.
  - Method arguments may not be of type float or double.
  - Methods may not return double or float values.
  - Objects of type Float and Double cannot be created (and, in fact, these classes do not exist in CLDC)
Note that Sun does not supply a different version of its Java compiler for use when developing CLDC applications, so it is possible, using a J2SE compiler, to create Java class files that use floating-point types and, therefore, violate these rules. However, these class files will be rejected when they are loaded into the CLDC virtual machine during class file verification.

Aside from the floating-point restrictions, a few other Java language features are not available to CLDC applications:

- **Reflection**: The java.lang.reflect package and all of the features of java.lang.Class that are connected with reflection are not available. This restriction is applied partly to save memory, but it also saves having to determine whether application code has the privilege to access these features.

- **Weak references**: Weak references and the java.lang.ref package are not provided because of the memory required to implement them.

- **Object finalization**: Object finalization causes great complexity in the VM for relatively little benefit. Therefore, finalization is not implemented, and the CLDC java.lang.Object class does not have a finalize() method.

- **Threading features**: CLDC provides threads, but it does not allow the creation of a daemon thread (a thread that is automatically terminated when all non-daemon threads in the VM terminate) or thread groups.

- **Errors and exceptions**: J2SE has a large number of classes that represent error and exception conditions. Since Java applications are not, in general, expected to recover from errors (meaning thrown exceptions derived from the class java.lang.Error), most of the classes J2ME in a Nutshell: Representing them are not included in the CLDC platform. When such an error occurs, the device is responsible for taking appropriate action instead of reporting it to application code.

- **Java Native Interface**: CLDC does not provide the J2SE JNI feature, which allows native code to be called from Java classes. JNI is omitted partly because it is memory-intensive to implement and partly in order to protect CLDC devices against security problems caused by malicious application code.

### 4.1.3 Multiple Platforms

.NET CF supports only one OS platform—Windows. One could argue that .NET CF is cross platform to some degree because of the CLR. Windows CE and Pocket PC operating systems run on more than 200 different devices. .NET CF applications are directly portable across those devices only.

For many mobile developers, having their applications run on multiple platforms with minimum effort proves essential. This is where Java really shines. Many of the mobile platforms now have built-in Java support. Such platforms include:
• On cell phone devices: Motorola iDEN, Nokia Symbian OS, and Qualcomm Brew, Nokia Series 40.
• On low-end PDAs: Palm OS.
• On embedded or telematics devices: QNX Software Systems and Wind River’s VxWorks.
• On high-end PDAs: Symbian OS and different Linux flavors

Third party J2ME runtimes from Insignia and IBM are available for all mobile platforms, including all Windows flavors. The Java approach allows developers to be productive across many mobile platforms.

Given the variety of mobile devices, the lowest common denominator approach does not work. Quite a few standard J2ME extensions and optional packages support various features unavailable on every device (e.g., SMS (Short Message Service) and multimedia playback), which could confuse new developers. Device vendors also tend to add value to their solutions by providing proprietary J2ME extension packages. Even for well-defined standard J2ME platforms, such as MIDP (Mobile Information Device Profile), different vendors have slightly different implementations. Thus, for J2ME to keep its cross-platform promise, we must put more effort into the standardization process.

4.1.4 Multiple Languages

A much-touted benefit of .NET is its ability to support multiple programming languages. That ensures that .NET CF can reach a variety of developers and reuse existing libraries. However, due to the object-oriented nature of the .NET CLR, developers must be familiar with object-oriented programming concepts before they can write effective .NET code. The .NET CF development tool, VS.NET, currently supports only two major .NET languages: C# and VB.NET.

4.1.5 Web Services

XML Web services are the key to future enterprise integration. SOAP (Simple Object Access Protocol) is becoming the ubiquitous protocol for accessing componentized enterprise back ends. Being an early adopter and promoter of SOAP Web services, Microsoft has a head start on Web services integration with mobile devices. With the support from VS.NET, consuming Web services in .NET CF applications is easy. In many cases, developers do not need to write any code and can just treat the remote service as a local object.

On J2ME, SOAP client support is currently not standardized. We must rely on third-party J2ME SOAP libraries, such as the open source kSOAP, to build mobile SOAP clients. Popular J2ME IDEs such as Sun ONE Studio, CodeWarrior Wireless Studio, and WebSphere Studio Device Developer have recently added SOAP client stub generators for kSOAP. Oracle
supports J2ME Web services clients through its upcoming 9i wireless application server. The server communicates with J2ME clients using a proprietary RPC (remote procedure call) protocol and relays SOAP messages. In the future, the J2ME Web Services Specification (JSR 172, available in third quarter 2003) will likely standardize J2ME Web services client API.

4.1.6 Development Tools

The list below summarizes available mobile application platforms and developer groups most likely to adopt them:

- .NET CF: .NET desktop application developers; VB developers.
- J2ME/CDC: J2SE client application developers; Java Web applet developers; J2EE server application developers for OSGi gateways.
- J2ME/CLDC: All J2SE developers. To become familiar with javax.microedition APIs, especially the UI features, some learning is required.

4.1.6.1 .NET CF

Microsoft’s flagship IDE Visual Studio .NET is an excellent product that provides similar design interfaces for desktop and mobile applications. For example, to migrate a desktop UI design to .NET CF, you merely copy and paste visual components to a new designer window. VS.NET also features strong support for Web services integration and relational database access. Most auto-generated code snippets are easy to read and modify by hand.

VS.NET is tightly integrated with Visio Enterprise Network Tools edition, which can generate C# or VB.NET code from your UML (Unified Modeling Language) diagrams. VS.NET supports debugging on both high-fidelity emulators and real devices. However, VS.NET is not cheap. As of today, no free command-line tool exists for .NET CF development. Nor does any third-party IDE product support .NET CF. More tools will be available in the future when .NET CF matures.

4.1.6.2 J2ME

On the J2ME front, command-line tools and vendor-specific toolkits are readily available. Sun’s J2ME Wireless Toolkit is a widely used MIDP development tool. Antenna is an open source project that extends Java’s de facto Ant build tool to J2ME. However, for most developers, IDEs are still essential. All major Java IDEs now have J2ME modules or plug-ins:

- Sun ONE Studio Community Edition with wireless modules is free and has excellent support for enterprise features.
• JBuilder with MobileSet has a great visual UI designer and good UML design support.
• CodeWarrior Wireless Studio is bundled with many useful third-party tools. It supports development on both CLDC and CDC/PersonalJava.
• IBM WebSphere Studio Device Developer is based on the popular Eclipse IDE platform. It supports both on-device and emulator debugging. If you choose one of IBM’s smart mobile middleware solutions, WebSphere Studio Device Developer is naturally the best tool.
• Oracle9i JDeveloper IDE helps integration between Oracle mobile servers and J2ME clients.
• Data Representation’s Simplicity IDE has visual designers not only for UI components, but also for end-to-end communication logic components, such as an XML-based transaction engine. Simplicity supports integration with legacy (mainframe) information servers through a visual screen reader.

4.1.7 Summary

<table>
<thead>
<tr>
<th></th>
<th>.NET Compact Framework</th>
<th>J2ME CDC</th>
<th>J2ME CLDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface</td>
<td>Rich subset of</td>
<td>Rich subset of AWT (Abstract Windowing Toolkit), vendor-specific UI libraries</td>
<td>MIDP liquid crystal display UI, PDA Profile subset of AWT, vendor-specific UI libraries</td>
</tr>
<tr>
<td></td>
<td>windows forms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database API</td>
<td>ADO.NET</td>
<td>Rich subset of JDBC</td>
<td>Vendor-specific JDBC-like APIs</td>
</tr>
<tr>
<td>Mobile database</td>
<td>SQL Server CE, SQL</td>
<td>Any JDBC compatible</td>
<td>Vendor-specific JDBC-like API bridge</td>
</tr>
<tr>
<td></td>
<td>Server Mobile, Sybase iAnywhere Solutions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database synchronization</td>
<td>SQL Server Mobile Synchronization</td>
<td></td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>XML API</td>
<td>ADO.NET built-in</td>
<td>3rd party tools</td>
<td>3rd party tools</td>
</tr>
<tr>
<td>Web services</td>
<td>Built in</td>
<td>3rd party tools</td>
<td>3rd party tools</td>
</tr>
<tr>
<td>Web services tools</td>
<td>Integrated in VS.NET</td>
<td>kSOAP plug-ins for leading IDEs</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>Direct RPC</td>
<td>Not recommended</td>
<td>Rich subset of RMI</td>
<td>N/A</td>
</tr>
<tr>
<td>Email and PIM</td>
<td>Platform Invocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outlook APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JavaPhone and 3rd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>party APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMS</td>
<td>Platform Invocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>device SMS stack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instant Messenger</td>
<td>P/Invoke MSN and</td>
<td>3rd party APIs for most IM clients including jabber and jxta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other IM client APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Messaging</td>
<td>P/Invoke MSMQ</td>
<td>Proprietary JMS APIs</td>
<td>JMS via 3rd party toolkits (WebSphere MQ, Everplace, iBus Mobile)</td>
</tr>
<tr>
<td>Cryptography</td>
<td>3rd party APIs</td>
<td>JCE (Java Cryptography Extension) and 3rd party libraries</td>
<td>3rd party libraries</td>
</tr>
<tr>
<td>Multimedia</td>
<td>P/Invoke Windows</td>
<td>JMF subset</td>
<td>Built into MDP plus J2ME multimedia APIs</td>
</tr>
<tr>
<td></td>
<td>Media Player APIs /</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3rd party APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>Included Windows</td>
<td>Direct Draw on Canvas</td>
<td>GameCanvas support in MDP</td>
</tr>
<tr>
<td></td>
<td>Forms UI / DirectX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on Windows Mobile 5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location API</td>
<td>APIs provided by</td>
<td></td>
<td>3rd party</td>
</tr>
<tr>
<td></td>
<td>carriers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: .NET CF vs. J2ME

For our implementation, we chose to go with .NET Framework and .NET Compact Framework in order to investigate in further detail the intricacies of developing an ambient
information system in the most widely acclaimed platform for rapid application development.

4.2 Data infrastructure

While the collection of building blocks is designed having in mind scalability and extensibility issues, a portion of it is very much depended on the underlying data infrastructure, that is the centralized database that contains the schema of information and is filled with data. Despite the limitations of centralized systems, we try to make the most out of it by employing visible and hidden features that exist in today’s relational database systems. The database system of choice is Microsoft’s SQL Server 2005. Upon that database we designed our relational data schema in a generic manner, aimed at storing information for the integral questions, a context-aware pervasive system is supposed to have an answer for; these questions are:

1. What are users doing? (Logging of user’s activities)
2. Where am I? (Current location and current map)
3. What can I find near me? (Spatial information for proximate artifacts)
4. What can I learn about proximate items? (Data content)

The following schema describes the relational entities that we use for providing answers to these questions.

Figure 11: Database diagram
The first question is answered by the UserActions entity, which is capable of storing, in a
generic manner, an ActionType and UserActionDescription. The ActionType describes any
notable action a user can make within our system, while the UserActionDescription are the
arguments for these actions, represented in a comma-separated manner.

The second question is answered by the Maps entity, which contains information on the
current levels that comprise our site. However, it also employs the location sensing to fully
answer the demanding task of locating one’s position.

The third question is answered by the Items entity and more specifically by the ItemPoint
attributes which positions the item in pixel coordinates within a Map.

The fourth and final question is answered by the Data Entity, which capable is storing
multiple DataType such as video, audio, images and even special data types such as RDF
schemas that provide a deeper description of each item.

In order to support the relational database choice we provide the following arguments:

- **Performance benefits and scalability**: relational database have been around for
  quite some time now and have been optimized in terms of data retrieval speed and
  for storing significantly large amounts of information. This means that our hardware
  accelerated graphical user interface would be caught up waiting for data from the
database, thus impairing the user interface experience, but will be able to respond
  on time unaffected by the fact that other users may be working on the database at
  the same time. Furthermore, they have stronger data management mechanisms
  using memory-paging functionality than opponent object-oriented or RDF based
  databases since they do not attempt to load the whole database file into memory
  but only the required portion of data and the related information according to the
  principle of locality. The same principle is applied to SQL Server 2005 Mobile edition,
  which is used by the PDA clients.

- **Synchronization mechanisms**: Most popular database systems posses’
synchronization functionality between various data sources. SQL Server 2005 uses
the subscription/publication mechanism in order to ensure that the mobile users
have the same snapshot of information as the central database.

- **Support mobile devices**: Support for mobile databases is crucial for building
  ambient information systems. A full-blown database system cannot be used from a
  PDA since the CPU has limited processing power and there is notably less memory to
  work with. This means that a mobile database system must be specifically
  engineered for a PDA and furthermore be deprived of certain capabilities that may
  lead in rendering the device unusable. SQL Server Mobile 2005 has been tested with
  this collection of building blocks and has been proven to operate efficiently.

- **Reusability**: Can easily be connected to pre-existing databases that may already be
  present in the premises. This property is rather unappreciated parameter since most
  approaches claim that the database schema should be able to be augmented and
  not to reuse schemas that exist in other data sources. We argue that re-using
preexisting relational databases that may already be installed in a server inside the building may greatly benefit the repository of data that is presented in the user’s information sources. For example, a museum may already have an existing database with information on each artifact. This means that the user who will be associating each piece of data with each data source does not have dwelve deep into the data in order to create the presentation slide show but merely create an associate among each artifact and it’s spatial location.

- **Well documented and acknowledged**: Relational databases and relational algebra are core curriculum courses in all computer science departments. Even though relational statement may not have the ability to be as expressively powerful as FLWR expressions, most people are comfortable using them. However, this period is a transitional one since there is a significant amount of research being done on data retrieval expressions, an outcome of which is the integration of LINQ expressions to the new C# 3.0 compiler.

- **Integrated security mechanisms**: Security is also another significant aspect since we do not want unauthorized access to our data and we do not want an unauthorized user to alter them.

- **Schema meta-data**: Finally yet importantly, the most significant aspect of data relational systems have is meta-data information for each table. More specifically using a SQL query one can discover the columns and the data types of each table. This is equivalent to using an XSD schema or an RDF schema for viewing the properties of an entity.
4.3 AMIS Macroscopic architecture

This section describes the building blocks macroscopic architecture. More specifically, Figure 12 depicts each tool that compensates for the creation of an ambient information system and how each building block is employed by it.
4.3.1 Data-Access Layer

The Data-Access layer is responsible for the creation of a transparent layer between the developer and the database whether the database is centralized or mobile. The layer is consisted of three classes, the DataStore class, the MasterDataStore class and MobileDataStore class. As depicted in Figure 13 the DataStore class is an abstract class that receives a connectionString in its constructor and determines from that string in which type of database provider it is going to connect. The database providers can be any provider that has a .NET driver such PostgreSQL, Oracle, Firefox and of course SQL Server. The transparency between the database entity and the real life artifact is achieved using the functions PopulateTable and CreateDataAdapter.

*PopulateTable* receives as input the TableName, which may be Area, Item, User, or anything else that we have represented in our schema. Then it queries the relational databases metadata to discover the columns that comprise that data table as well as their data types in order to ensure a type safe transaction between the developer and the data store. Then it populates Data Table object with all this information and leaves at the developer’s disposal.
The developer does not need the inner composition of this data table since it is insignificant because Ambient Information Systems do not have a stable schema, a certain manner of compiling their information.

`CreateDataAdapter` creates a data adapter that feeds on the information `PopulateTable` has brought from the meta-data of a table. Using that information it can create the `SelectStatement`, `UpdateStatement`, `DeleteStatement` and `InsertStatement`. It also retrieves primary and foreign key information to ensure data integrity among transactions. With these tools the developer can interact with each `DataTable` and manipulate its contents leaving the responsibility of content quality to the Content Manager application.

### 4.3.2 Location-Sensing client

The location-sensing client queries the Ekahau Positioning Engine, a centralized off-the-shelf location sensing system that grants us with the user’s position in pixel-based floor plan coordinates. The operation of this layer is defined by the manner the Ekahau engine works so order to introduce a level of abstraction and make this component support feature generations of location-sensing systems we have built the Client class on top of the `PositioningClient` class.

The `PositioningClient` class defines a Positioning Engine as entity that operates using a Socket Server and stands centralized on an IP Address, monitoring calls in a specific Port.

The Ekahau Positioning Engine 4.0 publishes the user’s location information using the HTTP protocol. The user sends a parameterized HTTP request to the engine’s URL and the engine comes back with an XML packet.
The XML packet (Figure 15) is parsed afterwards and a callback is raised notifying the client that the user’s position has changed. Furthermore, the Client class has been imbued with a flavor of robustness due to a phenomenon that was observed while roaming around a site using the client. Since the quality of service criteria, for choosing the appropriate wireless access point is signal strength, it is expected from a wireless PDA, while roaming inside a building to disconnect from one access point and connect to another, if it sees fitting that the new access point can serve him more effectively. Besides, the observation, that signal strength may not be the best quality of service criteria for choosing an access point, we also observed that in various PDA devices, depending on the size of the stack structure that stores the available access point, the amount of time it takes to associate from one access point and associate to another, may impair the client’s performance and may also result in the loss of the HTTP response, even though it is scheduled to retransmit packet’s that were supposed to be delivered to the user.

In order to create a counter measure to this problem, we created a small buffer, for storing the user’s HTTP requests and retransmit them, in case one of them has not received its response. This ensures that the user will constantly receive positioning information while roaming a site, covered with 802.11 wireless infrastructures.
As one may notice the size of each XML packet, containing the user’s position, as well as diagnostic data, is approximately 623 bytes. If we limit our query on positioning data (posgood tag) only, the XML packet size drops to 436 bytes, thus we gain a 30% decrease in bandwidth usage. Decreasing bandwidth usage is important because in a scenario where we are required to track 1000 users, a 30% increase may significantly improve the performance of our wireless infrastructure. However, one still notices that positioning data still contains more information than we require for tracking a user efficiently. In order to track a user efficiently we only need the posx, posy and mapid tag. If we could limit the XML packet to just these three tags, the packet size would drop to 75 bytes and give us an 87% decrease in payload size. Unfortunately, with EPE 4.0 such an optimization is not available.

4.3.3 Network-layer

![Network-layer class diagram](image-url)
While each of the building blocks has a value of its own, the full benefit is only available via a comprehensive integration and combined use. The network infrastructure provides the connectivity between the components, while the network-layer provides a wide range of cooperative sharing capabilities. The foundation of the network-layer is the SocketBase class, which is the manifestation of the wrapper, programmatic pattern, around .NET’s socket class. The extended functionality it provides revolves around the creation of callbacks that are triggered asynchronously whenever the respective event has occurred. The collections of events are show in Figure 16. On top of the SocketBase class, we have implemented the SocketServer and the SocketClient. The SocketServer, upon construction, ties itself to a specific port and waits for SocketClient connection calls. For each client that establishes a connection, the SocketServer stores it is call into an array and has the ability to iterate that array and send or broadcast calls to its clients on demand.

SocketClient is a TCP based socket client that constructs itself with the endpoints IP address and port. It is used to connect with a SocketServer or with another TCP based socket server and send and receive messages to and from it.

4.3.4 Map Viewer

The presentation of area maps is mandatory for a number of tasks:

- Administrators move around, viewing the current map and the currently entered artifacts in the DB, to “program” the position information of artifacts
- Administrators centrally manage the content viewing the spatial distribution of information items
- Users explore the physical space, navigated effectively through the interactive map
- System administrators centrally monitor the traffic in a museum viewing user traces over area maps
Such tasks need an aesthetically convenient representation capability since it will be used by humans on a daily basis and deal with burden of representing one’s surroundings. In order to deal with this problem we have create the Map Viewer component, shown in Figure 17, which is based on the VertexEnvironmentObject class. This class contains a collection of vertex buffers that are later on passed into the DirectX World transformation pipeline. This enhances performance since all such transformations are simple multiplications executed in the device’s GPU, which exists even in PDA devices. Inheriting from the VertexEnvironmentObject class, comes a collection of vertex-based objects for the representation of displayed artifacts (information items, users, maps, and areas). Furthermore, in order to increase representation fidelity each vertex buffer can be skinned using a Texture that may be an image of the artifact it represents. In order to ensure that textures, meshes, animations and even sounds within the environment are reused we have create a file caching mechanism, depicted in Figure 18.
Finally yet importantly, we find the Camera tool, shown in Figure 19. The camera tool is used for navigation in space, allowing the user to move around the virtual space, zoom in and zoom out.

Figure 19: Map Viewer Engine class diagram
4.3.5 Data Synchronization Service

The Data Synchronization Service serves the sole purpose of creating a low-bandwidth cost solution for the replication of data between a centralized database and distributed PDA clients. The reasons for employing such an elaborate scheme are:

- The distributed PDA clients are not always and they cannot always be online since that exist in the physical space and there is no guarantee that each point in space is covered by an access point.
- There are a large number of PDA clients in space (e.g., the Acropolis scenario has 1000 PDAs). This means alternative content provisioning solutions such as streaming are rendered unrealistic since they demand a large amount of bandwidth and create network congestion bottlenecks.
- Finally yet importantly, since off-the-shelf pocket databases pose 100MB limit in content size, we cannot store are multimedia content into blob database fields because this would definitely overflow the pocket database limit.

In order to deal with these challenges we have created a mechanism that transforms the database synchronization problem into a well-defined file synchronization problem. This is achieved by using the database build-in synchronization facilitates to take a snapshot of a central database into a pocket database file. The file is then stored within a folder that also contains the supporting multimedia content and this transaction, the addition, deletion or update of each file, is noted in an index xml file along with the date it occurred.
The multimedia content management and pocket database management is handled by the Content Management tool which is described in further detail in 4.4.3. The Content Management Tools sends a signal to the Synchronization Server, shown in Figure 20, telling him that there has been a change in the database content and that the clients need to be synchronized.

Each synchronization client, shown in Figure 21, that is currently connected with the server receives the synchronization message and starts to synchronize its content. The client synchronization process is implemented by comparing the dates between the central index xml file and the one in the PDA client. If a date mismatch exists, the file transfer process is initiated. If a client is not online, the UserNeedsSync field is denoted as true within the database, as a reminder that when the client connects to the synchronization server, he will be notified that he needs to be synchronized.

In the deployment scenario, we assume that the PDA client is docked and connected to a wired network using ActiveSync. When the PDA receives a message from the Server, the synchronization process commences. However, we noticed that when the client receives content for the first time, the amount of content might reach an enormous amount of data, 100MB, 500MB, even 1GB. Synchronizing 100MB over ActiveSync takes 15 minutes on a one Gigabit Ethernet line, 75 minutes for 500MB and 150 minutes for 1GB. The linear increase in this pattern is justified by the fact that the test was conducted on the weekend and the wired network infrastructure was not under any stress. However, since the wired infrastructure nowadays, is faster than the wireless we feel that by scheduling

---

**Figure 20: Synchronization server class diagram**

---
synchronization actions to occur in the evening while the PDAs are docked there would not be any hindrance in network performance.

Furthermore, special steps have been taken into account in order to ensure proper synchronization functionality even in the absence of the index file. More specifically, if the central index xml file is missing, the synchronization server will try to recreate it using the contents of the synchronization folder and the last accessed field of each file.

4.4 AMIS Tools

4.4.1 Mobile Location-Administration Tool

The Mobile Location-Administration Tool aims at resolving the varying, location-sensing system, accuracy characteristics. In other words, this means that recording the artifacts position using a single point does not suffice since, in the worst-case scenario, that recording may be in a 3-meter radius area. Furthermore, the location sensing system lacks the ability to discern between walls and corridors; it cannot detect any information regarding the room arrangement.

Using the Mobile Location-Administration Tool, we deal with these problems by walking with the PDA Client. The PDA Client employs the location-sensing client, the data access layer and the Map Viewer component to deal with this problem. More specifically, the user’s
position is retrieved using the location-sensing client and is depicted on the Map Viewer. The user using the PDA’s pen signifies polygonal areas that bound each point. Furthermore, he can also record each item’s position. We associate information items with polygons (areas), as they are given “on site” using the real location sensing system.

The administrator moves around physical items, recording all points as they are returned by the location sensing system, and then defining interactively a polygon enclosing all these points (i.e. the real location area of items).

Figure 22: Mobile Location-Administration Tool area functionality 1

Figure 23: Mobile Location-Administration Tool area functionality 2
4.4.2 Desktop Location-Administration Tool

The Desktop Location-Administration Tool has many similarities with its Mobile counterpart; however, it deals with recording, each physical artifact’s, spatial position in a centralized manner. It is used to create the first version of the positioning of each artifact, which will be used by the mobile administration tool as a point of reference for fine tuning purposes. It has the same functionality as the Mobile tool, such as recording an item’s position and the polygon of the bounding area. Furthermore, it can also invoke the Content Manager and associate multimedia content to each item, thus, enhance the user’s navigation experience.

The only aspect that ought to limit the performance of the Desktop Location Administration Tool is the texture size of the floor plan. The following table displays memory requirements of the graphics adapter for each texture size.
Table 4: Ram requirements for single texture in megabytes, with and without mipmapping (MM)

<table>
<thead>
<tr>
<th>Size of texture</th>
<th>16bit</th>
<th>24bit</th>
<th>32bit</th>
<th>16 bit w/MM</th>
<th>24 bit w/MM</th>
<th>32 w/MM</th>
<th>24 bit with 4x compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024*1024</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2.7</td>
<td>4</td>
<td>5.5</td>
<td>0.75</td>
</tr>
<tr>
<td>2048*2048</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>10.9</td>
<td>16</td>
<td>21.3</td>
<td>3</td>
</tr>
<tr>
<td>4096*4096</td>
<td>32</td>
<td>48</td>
<td>64</td>
<td>42.6</td>
<td>64</td>
<td>85.3</td>
<td>12</td>
</tr>
<tr>
<td>8192*8192</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>170.6</td>
<td>256</td>
<td>341.3</td>
<td>47</td>
</tr>
<tr>
<td>16384*16384</td>
<td>512</td>
<td>768</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>256</td>
<td></td>
</tr>
</tbody>
</table>

However, most modern graphics adapters dispense a large amount of memory for texture storage. The following table displays maximum texture size for the most popular graphics adapters.

Table 5: Maximum texture size for popular GPUs

<table>
<thead>
<tr>
<th>Chipsets</th>
<th>Max texture RAM (MB)</th>
<th>Max Texture Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nvidia GeForce Series</td>
<td>~110/240 (128/256 on card)</td>
<td>4096</td>
</tr>
<tr>
<td>ATI Radeon Series</td>
<td>~110/240 (128/256 on card)</td>
<td>2048</td>
</tr>
</tbody>
</table>
4.4.3 Content Management Tool

The Content Management Tool is the database administration application that is used for storing multimedia or textual content. It is comprised of two components, the Item Editor and the Navigation Scheme editor.

The Item Editor is a series of forms, which are created based on each item’s database identification number (an auto generated, auto incremental number). Given that number, it allows us to associate information to this item. Such information could be audio (e.g. a narrated description of the item), video (e.g. an animation describing how the item works), image and text. The Item Editor establishes it is communication with the central database using the Data Access Layer building block. More specifically, it uses the MasterDatastore and dynamically creates a representation of each Item entity, in columns and rows format, and facilitates data manipulation such as insertion, update, deletion and selection of data. Furthermore, in case the MasterDataStore detects that more than one rows are affected in the database, it automatically breaks each commit into transaction and in a sequential manner, tries to commit each change. If one change fails to complete, than the whole
transaction is rolled back and no change is committed. This feature further enhances data integrity. Another building block, the Item Editor uses is the Synchronization Server. Specifically the Item Editor is the component that has an awareness of whether or not the MasterDataStore contents have been changed. Based on that awareness, it can determine when to invoke the Synchronization procedure by sending a TCP message to the Synchronization Server and telling him to broadcast a message to its client, notifying them that updates must be made.

The Navigation Scheme Editor is another form that depicts all the navigation scenarios within a site. The scenarios appear in different tabs, along with lists that contain all the selected items for each navigation scheme and all the available items for the same navigation. Using this application, the user can create, insert, delete or alter existing scenarios. Furthermore, he can also define the order that each item, in each navigation scheme, presents its self to the user while he is navigating himself in the physical space. The Navigation Scheme Editor also uses the Data Layer building block to establish a connection with the database.
5 CASE STUDY: A MUSEUM GUIDANCE SYSTEM

The testing and evaluation of the collection of building blocks has been conducted through the creation of a virtual version of the Mycenae museum. The process involves the following steps:

1. Assign areas and items position in each museum floor (Area and item positioning)
2. Associate each physical artifact with multimedia information (Content association)
3. Initiate the synchronization server and synchronize all clients (Synchronization)
4. After synchronization is complete take an emulated PDA client for a walk in the museum (Visitor Emulation)
5. If something is out of place rectify it and go to step 3 (Error handling)
6. If all is complete take the PDA for a walk in the museum and enjoy your tour (Virtual tour)

The following chapters are going to provide our insights while creating the museum.

5.1 Area and item positioning

In order to create the polygonal areas that bound each collection of items and each item we open the Content Management application. When the application starts, it displays the floor plan for each floor plan that has been inserted into the system using the Map Administration tool. The first step is to create each polygonal area. By clicking the “New Area” button, found on the application’s task pane, the application’s cursor turns into a cross, prompting the user to select each point that describes the desired polygon. The polygon can be consisted of many points. When the user decides to stop adding points, all he needs to do is press the right mouse button and the area creation procedure will be completed. In case the floor plan is not clear enough the users, can scroll or zoom in space until he reaches the desired perspective. If the user is not satisfied with each area position, he is allowed to move the area to a new place or delete it and recreate it.

The item creation procedure bears great resemblance to the one mentioned prior. The user selects “New Item” and a new item is created in the center of the floor plan. The user can than move the item around and associate it an area. Upon association, the item’s color will change from Slate Gray to Orange. In case a physical item’s position changes, the user is free to move it in the floor plan and re associate it with another area. Furthermore, if an area changes places in the physical realm, the user can move the area within the Content Management application and while moving the area, the associated items, will also move along. When the creation procedure finishes you should get a floor plan just as the one depicted in the following figure.
5.2 Content association

Once each area and each item has been recorded, the next phase is to associate multimedia content. Using the Content Management application the user selects each item and clicks on “Edit Item”. While editing the item’s content, the user specifies the title, textual, imagery and multimedia content. An example of such an item included in the Mycenae museum is the entrance chamber (“Προθάλαμος-Αίθουσα Υποδοχής Κοινού”). The entrance chamber has a short description with three images, an audio description and finally an analytical description with seven images. Each aspect images also contains a title and a description and in addition, it may be multilingual. An instance of such information is depicted in Greek in the following table.
Dear visitors, welcome to the Archaeological Museum of Mycanea, a modern local museum that opened up its gates in 2003. It is hidden in the northern side of the acropolis. It was build within the archaeological site providing a shelter for the archaeological artifacts found within. We hope you enjoy your stay.

<table>
<thead>
<tr>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dear visitors, welcome to the Archaeological Museum of Mycanea, a</td>
</tr>
<tr>
<td>modern local museum that opened up its gates in 2003. It is hidden</td>
</tr>
<tr>
<td>in the northern side of the acropolis. It was build within the</td>
</tr>
<tr>
<td>archaeological site providing a shelter for the archaeological</td>
</tr>
<tr>
<td>artifacts found within. We hope you enjoy your stay.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Title</th>
<th>Mycenae Archaeological Museum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image File</td>
<td>12M01F01b.jpg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Sketch</th>
<th>Acropolis hill</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>12M01F02.jpg</td>
</tr>
</tbody>
</table>

Table 6: Item content snippet

The navigation within the Item Editor is done in a tabular manner. The user selects the Image tab to associate a file, a title and a description for each image. This process takes a lot of time and the only benefit is the creation of virtual representation of each physical artifact without the need to embed a sensor to it.

5.3 Synchronization

Once all content has been committed to the database, the user notifies the Synchronization server to broadcast the “Sync Command” to all the PDAs. When a PDA receives that command, it connects to an FTP server and downloads the index file, which maps all changes. Then it compares the newly downloaded file with the local file, downloads the new additions, and deletes orphaned files that are no longer associated. The amount of time this process consumes depends on the content’s size. The synchronization system also contains certain facilities for logging each synchronization procedure, as well as a callback to enforce synchronization.
5.4 Visitor Emulation

Visitor emulation is the process where the User Monitor interactively “moves” a user from one point to another and the PDA client emulates the data representation each move is going to produce. Both applications are set to emulation mode and client waits for positioning information, not from the location-sensing client, but from the User Monitor itself. When the PDA client receives a new position it acts as if the user has physically moved to the specified position and depicts the information available from the information sources within his proximity.
5.5 Error Handling / Spatial Recalibration

The visitor emulation process aims at producing errors found within the process of spatial association and content correlation. If any error is found the user can repeat the previous steps, reposition each area, each item and rectify any data association. After that, he is still obliged to follow the life cycle we are presenting and synchronize the system again.
5.6 Virtual Tour

The final step is the virtual tour itself. The application is ready to be deployed and the only necessity is for the user to be willing to explore the artifacts within a museum. We hope that this process can prove that the emergent of this process is greater than the summation of the building blocks.
6 Conclusions

The collection of building blocks was designed in accordance to the design dimension of Ambient Information Systems, which are information capacity, representation fidelity, aesthetic emphasis and notification level. In addition, they are designed to be a scalable, modular and extensible solution for context aware computing. This thesis began by first providing a brief overview of some of the current Ambient Information Systems and on location-sensing technologies available for context-aware computing. It then outlined the design dimensions for a Ambient Information Systems and highlighted what aspects of those dimensions the building blocks fulfill. The collection of building blocks has proven its worth through the deployment of a museum guidance system. Furthermore, it is suitable for other types of applications such as Location-based games.

Chapter 4 described the collection of building blocks in micro and macro architectural level. The collection of building blocks is comprised of a data access layer, a location-sensing client, a synchronization service and a mobile location-administration system. For each building block, we have given a systematic account of the main challenges it addresses in the context of ambient information systems. The data access layer provides a flexible mapping between physical artifacts within the environment and objects within the relational database. The location-sensing client allows us to query the wireless infrastructure to retrieve user positions in real time. The synchronization layer allows us to ensure that information mirrored in distributed resources remains always up to date. Finally, the mobile location-administration system allows the administrator to interactively program ‘on site’ the physical position of information elements.

Chapter 5 provided an evaluation of the building blocks in the implementation of a museum e-guide application.

One conclusion is that WLAN is not the best choice for location-sensing today since it is very expensive to acquire an off-the-shelf location-sensing package and even more expensive to install access point within a site. However, if access points are already installed it may be a fine solution to the “where am I” problem and furthermore it can be augmented by using supplementary proximity devices within the physical space.
7 Future Work

As stated at the beginning of this thesis, the collection of building blocks is not meant to be used in a vacuum. It is meant to be the first component of a larger system that for pervasive computing. There are still many components that need to be developed for this vision to become a reality. Some of these components are an efficient system for remote data access and a system to provide more flexibility inherent to the system.

Some new generic building blocks could be implemented for specific location sensing technologies such as GPS location sensing, RFID /IF tags for proximity sensing and vision-based modules, for collaborative purposes, (e.g. complement on the accuracy problems of the WIFI LAN).

There also is work that can be done within the collection of building blocks to improve the system. Some of it is simple implementation issues with cleaning up the code, while others present larger issues that are necessary to make the collection of building blocks a robust environment, such as fault tolerance and adaptability.

Finally there should be wider vision beyond the collection of building blocks, that towards a complete pervasive infrastructure. A good place to look next is toward uneven conditioning and allowing services to automatically adapt to changing conditions.

Implementation-specific refinements to better generalize the APIs, providing hooks for extensions over different technologies, while making deployment even easier.
8 References


21. *Heuristic evaluation of ambient displays*. Mankoff, J.


9 Appendix

9.1 Wireless Location Sensing

9.1.1 Training Process

The training process for an 802.11-based location sensing system is a rather awkward one. The process, also known as calibration, aims at creating a distinction among physical cells using signal strength signatures from wireless access points. In order to complete, the calibration one must complete the following steps:

1. Set up at least four wireless access points.
2. Install the calibration software (such as the Ekahau Site Survey) on one laptop and a few mobile devices (laptops or pdas)
3. Mark a square grid out on the floor of the area you want to establish location sensing. Each point should be at least 5 feet from the other points.
4. Turn on the calibration software on your primary laptop and initiate the procedure
   - stand on grid point 1
   - with primary laptop on your hands
   - do a smooth 360 degree turn while standing on grid point 1 in exactly 20 seconds
   - move to grid point 2 and repeat until you’ve danced on all the grid points

This procedure allows the location-sensing engine to record the signal-strength signature pattern of each cell. The intuition behind the 20-second calibration and 360-degree turn is simple. We want to retrieve a valid signal strength sample from the wireless access points, one that is not affected by temporal changes (41).

9.1.2 Signal Strength Retrieval

The process of retrieving signal strength signatures from the wireless adapter is highly coupled with the low-level hardware structure. By viewing the source code of popular open-source network packet analyzers such as tcpdump (42) or iwlist (43), one can observe that each wireless adapter packet type, namely prism, madwifi (44) or ieee802.11 (45) has its own distinct structure and unique way of viewing the information it encapsulates.
As demonstrated in the following source snippet the packet analysis procedure in Linux based systems is done in the following fashion:

1. Determine the packet type (atheros (46), ieee802.11, prism); this is done by measuring the length of the packet.

2. When the packet type is determined a switch case structure retrieves information from each bit found in the packet.

However, recent changes in the Linux kernel have shown that the development process is moving towards a generic 802.11 stack (47) with a standardized pattern for retrieving such information. More specifically in Linux Kernel 2.6.22 (48), the new wireless stack (a donation from Devicespace) contains a complete software MAC implementation, WEP, WPA, link-layer bridging module, hostapd, QoS support to prioritize things like VoIP, 802.11g support and full debugging capabilities. The presence of such features alleviates the developers from having to re-implement them and obliges them to conform to these standards.

Such is the case in the Windows operating system where packet analysis is achieved through the Network Device Specification (NDIS). Through NDIS, the developer has signal strength level access information without having to re-implement his system or programmatic approach for each adapter or narrow it to a certain category of wireless adapters. As seen from the size in lines of code (SLOCS) of the packet analysis snippet, one can understand that is inefficient to maintain different analysis sequences for various wireless adapters and that such incompatibility sets back the development of signal strength based applications.

```c
static int print_radiotap_field(struct cpack_state *s, u_int32_t bit, struct packet* pkg)
{
    union {
        int8_t   i8;
        u_int8_t u8;
        int16_t  i16;
        u_int16_t u16;
        u_int32_t u32;
        u_int64_t u64;
    } u, u2;
    int rc;

    switch (bit) {
    case IEEE80211_RADIOTAP_FLAGS:
    case IEEE80211_RADIOTAP_RATE:
    case IEEE80211_RADIOTAP_DB_ANTSIGNAL:
    case IEEE80211_RADIOTAP_DB_ANTNOISE:
    case IEEE80211_RADIOTAP_ANTENNA:
        rc = cpack_uint8(s, &u.u8);
        break;
    case IEEE80211_RADIOTAP_DBM_ANTSIGNAL:
    case IEEE80211_RADIOTAP_DBM_ANTNOISE:
        rc = cpack_int8(s, &u.i8);
        break;
    } // end switch
}
```
case IEEE80211_RADIOTAP_CHANNEL:
    rc = cpack_uint16(s, &u.u16);
    if (rc != 0)
        break;
    rc = cpack_uint16(s, &u2.u16);
    break;

case IEEE80211_RADIOTAP_FHSS:
case IEEE80211_RADIOTAP_LOCK_QUALITY:
case IEEE80211_RADIOTAP_TX_ATTENUATION:
    rc = cpack_uint16(s, &u.u16);
    break;

case IEEE80211_RADIOTAP_DB_TX_ATTENUATION:
    rc = cpack_uint8(s, &u.u8);
    break;

case IEEE80211_RADIOTAP_DBM_TX_POWER:
    rc = cpack_int8(s, &u.i8);
    break;

case IEEE80211_RADIOTAP_TSFT:
    rc = cpack_uint64(s, &u.u64);
    break;

default:
    /* this bit indicates a field whose 
     * size we do not know, so we cannot 
     * proceed. 
     */
    printf("[0x%08x] ", bit);
    return -1;
}

if (rc != 0)
    return rc;

switch (bit) {
    case IEEE80211_RADIOTAP_CHANNEL:
        pkg->channel = ieee80211_mhz2ieee(u.u16, u2.u16);
        break;
    case IEEE80211_RADIOTAP_FHSS:
        printf("fhset %d fhpat %d ", u.u16 & 0xff, (u.u16 >> 8) & 0xff);
        break;
    case IEEE80211_RADIOTAP_RATE:
        pkg->rate = u.u8;
        break;
    case IEEE80211_RADIOTAP_DBM_ANTSIGNAL:
        pkg->signal = u.i8;
        break;
    case IEEE80211_RADIOTAP_DBM_ANTNOISE:
        pkg->noise = u.i8;
        break;
    case IEEE80211_RADIOTAP_DB_ANTSIGNAL:
        pkg->signal = u.u8;
break;
case IEEE80211_RADIOTAP_DB_ANTNOISE:
    pkg->noise = u.u8;
    break;
case IEEE80211_RADIOTAP_LOCK_QUALITY:
    printf("%u sq ", u.u16);
    break;
case IEEE80211_RADIOTAP_TX_ATTENUATION:
    printf("%d tx power ", -(int)u.u16);
    break;
case IEEE80211_RADIOTAP_DB_TX_ATTENUATION:
    printf("%dB tx power ", -(int)u.u8);
    break;
case IEEE80211_RADIOTAP_DBM_TX_POWER:
    printf("%dBm tx power ", u.i8);
    break;
case IEEE80211_RADIOTAP_FLAGS:
    if (u.u8 & IEEE80211_RADIOTAP_F_CFP)
        printf("cfp ");
    if (u.u8 & IEEE80211_RADIOTAP_F_SHORTPRE)
        printf("short preamble ");
    if (u.u8 & IEEE80211_RADIOTAP_F_WEP)
        printf("wep ");
    if (u.u8 & IEEE80211_RADIOTAP_F_FRAG)
        printf("fragmented ");
    break;
case IEEE80211_RADIOTAP_ANTENNA:
    printf("antenna %d ", u.u8);
    break;
case IEEE80211_RADIOTAP_TSFT:
    printf("%" PRIu64 "us tsft ", u.u64);
    break;
}
return 0;

9.2 3D Environment Modeling

This section describes the process of creating a 3D environment model from a floor plan using the open-source 3D Editor, Blender (49). The process was applied in the floor plan of the ground floor of the Institute of Computer Science. The foundation for this process is the displacement mapping technique. Displacement mapping allows a texture input to manipulate the position of vertices on rendered geometry. Unlike Normal or Bump mapping, where the shading is distorted to give an illusion of a bump, Displacement Maps create real bumps, creases, ridges, etc in the actual mesh. Thus, the mesh deformations can cast shadows, occlude other objects, and do everything that changes in real geometry can do.
For years, displacement mapping was a peculiarity of high-end rendering systems like PhotoRealistic RenderMan, while real-time APIs like OpenGL and DirectX, lacked this possibility. One of the reasons for this absence is that the original implementation of displacement mapping required an adaptive tessellation of the surface in order to obtain micro polygons whose size matched the size of a pixel on the screen.

In order to implement this procedure one must complete the following steps:

1. Create a plane

2. Apply the texture of the floor plan on the plane
   ([http://www.ics.forth.gr/~vandikas/maps/ground%20floor.jpg](http://www.ics.forth.gr/~vandikas/maps/ground%20floor.jpg))

3. Apply the displacement modifier on the texture

This process will let blender know that wherever it finds a dark area on the texture it may use it to create a wall. The wall's rigidness depends on how dark the area is which means that the final result will depict pyramid-like walls. By merging the polygons of each wall, you will finally reach a result such as the one in Figure 29.

![Figure 29: Blender floor plan](http://www.ics.forth.gr/~vandikas/maps/ics_ground_floor.blend)