Development of a Web Service for Visualizing the Data of Ultra Wideband Real-Time Location System

JunYu JIAN

A Thesis

in

Concordia Institute for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montreal, Quebec, Canada

February 2011

© JunYu JIAN

Concordia University

Department of Engineering and Computer Science

Concordia Institute for Information systems Engineering

This is to certify that the thesis prepared

By: **JunYu JIAN**

Entitled: Development of a Web Service for Visualizing the Data of Ultra Wideband Real-Time Location System

and submitted in partial fulfillment of the requirement for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. A. Awasthi ______________________	Chair
Dr. S. Li ______________________	Examiner
Dr. C. Alexsandru ______________________	Examiner
Dr. A. Hammad ______________________	Supervisor
Dr. C. Wang ______________________	Supervisor

Approved by ______________________
Chair of Department or Graduate Program Director

____________ 20_____ ______________________
Dean of Faculty

ABSTRACT

Development of a Web Service for Visualizing the Data of Ultra Wideband Real-Time Location System

JunYu JIAN

The Ultra Wideband (UWB) Real-Time Location System (RTLS) is an Indoor Positioning Systems (IPS) designed to track and locate tagged objects and record detailed information including tag IDs and X, Y, and Z coordinates at every time step. UWB RTLS is valuable in diverse realms, such as industries, manufacturing, logistics, transportation, military, hazardous environments, immersive media and retail applications. However, in order to investigate the feasibility of extending the scope of UWB RTLS applications further into industrial and daily life areas, two initial problems are addressed in this research: (1) the definition of common requirements of UWB RTLS applications, and (2) the development of a Web service for UWB RTLS. The definition of common requirements of UWB RTLS application aims to determine the requirements of the Web service developed in this research. The development of the Web service for UWB RTLS aims to enhance the capabilities of UWB RTLS and satisfy the common requirements by implementing technology integration and collaboration, remote interaction, awareness visualization, and event detection. In order to demonstrate the usefulness of the proposed Web service, two applications have been developed using its services and those applications have been tested in several scenarios.

ACKNOWLEDGEMENT

First of all, my greatest appreciation goes to my two supervisors ─ Dr. Amin HAMMAD and Dr. Chun WANG for their intellectual support, personal instruction, selfless patience and spiritual encouragement. Their advices and criticisms were my most valuable asset during my studies and research. Overall, I feel very fortunate to have had the opportunity to know them and work with them.

In addition, I would like to give my appreciation to my research colleagues ─ Cheng ZHANG, Ali MOTAMEDI, Sonia RODRIGUEZ, and Yu SATOW for their kind support in this research. I deeply thank for their contributions.

DEDICATION

To my father WeiGuang JIAN, mother LiShan LU, and my friend Ying LI for their endless encouragement and support which made all of this possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Abbreviation	Description
ADN	Autodesk Developer Network
ADO	Active Data Objects
AEC	Architecture, Engineering, Construction
Ajax	Asynchronous JavaScript and XML
AOA	Angle of Arrival
API	Application Programming Interface
ASP	Active Server Pages
BAP	Battery Assisted Passive
BIM	Building Information Model
CLR	Common Language Runtime
ebXML	Electronic Business using eXtensible Markup Language
EJB	Enterprise JavaBeans
GDOP	Geometric Dilution of Precision
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IFC	Industry Foundation Classes
IIS	Internet Information Services
IL	Intermediate Language
IMES	Indoor Messaging System

IPS	Indoor Positioning Systems
Java EE	Java Platform, Enterprise Edition
JAX	Java APIs for XML
JCA	Java Connector Architecture
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
MEP	Mechanical, Electrical, Plumbing
OGC	Open Geospatial Consortium
OTW	On the Wire
PHP	Hypertext Preprocessor
RAC	Autodesk Revit Architecture Construction
RF	Radio Frequency
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
TDOA	Time Difference of Arrival
TI	Transaction Integrator
TOA	Time of arrival
UDDI	Universal Description, Discover, and Integration
UWB	Ultra Wideband

VB	Visual Basic
WMS	Web Map Service
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

CHAPTER 1 INTRODUCTION

1.1 General

Modern industries and daily life involve many various indoor activities which increasingly rely on indoor positioning technologies. Ultra Wideband (UWB) Real-Time Location System (RTLS), a reliable and precise system, is a promising technology which can be used to locate and track indoor objects. UWB RTLS can be particularly valuable in complex construction projects, such as airports, or in extreme conditions, such as burning buildings. UWB RTLS can generate real-time location data, which is a digital representation of the movements of tagged objects. In this research, real-time location data refer to the following attributes: tag ID, date (year, month, and day), time (hour, minute, second, and millisecond), X, Y, and Z coordinates in monitored sites (e.g. latitude, longitude, and altitude on earth), accuracy validity, accuracy Geometric Dilution of Precision (GDOP), accuracy error, and recording count.

In practical applications, UWB RTLS has some significant limitations. For example, (1) users are not able to develop their own applications by invoking Web Application Programming Interface (API) of UWB RTLS; (2) users cannot connect to UWB RTLS from Web browsers, observe detailed information of real-time location data in awareness visualization, or receive useful event messages calculated from real-time location data. Therefore, Web service is a crucial component in this research to enhance the capabilities of UWB RTLS in the areas of technology integration and collaboration, remote interaction, awareness visualization, and event detection.

1.2 Research Objectives

The primary objectives of this research are: (1) to define the common requirements of UWB RTLS applications which determine the requirements of the Web service developed in this research; (2) to develop a Web service which enhances the capabilities of UWB RTLS and satisfies the common requirements by implementing technology integration and collaboration, remote interaction, awareness visualization, and event detection; and (3) to develop two applications using the services provided by the Web service (a Web application and a software plug-in application) which demonstrate the usefulness of the Web service.

1.3 Thesis Organization

This thesis is presented as follows:

CHAPTER 2 LITERATURE REVIEW:

The main purpose of Chapter 2 is to review the general philosophies of the different types of positioning system technologies, environment modeling technologies, and Web Services. Firstly, different types of positioning systems are compared and particular emphasis is placed on UWB RTLS, the type of indoor positioning systems (IPS) used in this research. Then, three different types of environment modeling technologies are introduced focusing on Web Map Service (WMS) application, a prevailing environment modeling technology used in this research to simulate monitored environments for implementing awareness visualization as a supporting component. Finally, the introduction of Web services is reviewed focusing on the technologies of Web services and Web applications, and the comparisons of attributes and characteristics of two Web frameworks.

CHAPTER 3 REQUIREMENT DEFINITION:

The main purpose of Chapter 3 is to define the common requirements of UWB RTLS applications, and then determine the requirements of the Web service for UWB RTLS according to these common requirements. In this research, the Web service is named ***UWB RTLS Web Service***. Firstly, this chapter discusses the general requirements of UWB RTLS applications. Secondly, it explores the specific requirements of three typical UWB RTLS applications (employee monitoring, fire rescue, and crane operation) by analyzing their divisions of problems and situations to define the common requirements of UWB RTLS applications. Thirdly, it discusses the advantages of Web services for UWB RTLS applications. Finally, it determines the requirements of the ***UWB RTLS Web Service***, which are categorized into four parts based on the common requirements: (1) requirements of technology integration and collaboration, (2) requirements of remote interaction, (3) requirements of awareness visualization, and (4) requirements of event detection. Finally, it analyzes six fundamental use cases of the ***UWB RTLS Web Service***: (1) get real-time location data, (2) get visualization data, (3) get event messages, (4) save real-time location data, (5) save event messages, and (6) collect monitoring result.

CHAPTER 4 WEB SERVICE DESIGN FOR UWB RTLS:

The main purpose of Chapter 4 is to design the ***UWB RTLS Web Service*** to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications. Firstly, this chapter discusses the four solutions which independently satisfy the requirements of the ***UWB RTLS Web Service***: (1) solution of technology integration and collaboration, (2) solution of remote interaction, (3) solution of awareness visualization, and (4) solution of event detection. Then, it presents design details of the ***UWB RTLS Web Service*** including the namespace decision and interface design.

CHAPTER 5 IMPLEMENTATION:

The main purpose of Chapter 5 is to present the integrated components, the programming platform, and the three programming modules whose names are: (1) ***UWB RTLS Web Service***, which is the Web service developed to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications; (2) ***Tag Positioning Web Application***, which is an example of applications using the services provided by the ***UWB RTLS Web Service***; and (3) ***BIM Visualization Application***, which is an example of plug-in applications of Autodesk Revit Architecture visualizing real-time location data files in BIM.

CHAPTER 6 TESTS:

The main purpose of Chapter 6 is to test the ***UWB RTLS Web Service*** in three simulated tests and apply it in two practical tests by using the ***Tag Positioning Web Application***. Firstly, this chapter briefly describes three simulated tests which simulate the three aforementioned typical UWB RTLS applications to evaluate the capabilities of the ***UWB RTLS Web Service*** in terms of how well its requirements were satisfied. Then, it introduces two practical tests (tracking workers in a construction project and tracking the motion of a crane) which were completed by our research group.

CHAPTER 7 CONCLUSIONS & FUTURE WORK:

The main purpose of Chapter 6 is to summarize the present work, highlight the contributions of this research, and suggest recommendations for future research.

CHAPTER 2 LITERATURE REVIEW

2.1 Background

In this research, there are three primary components: (1) UWB RTLS (a type of positioning system); (2) WMS (a type of environment modeling technology) for the simulation of monitored environments in awareness visualization; and (3) Web technologies for Web services developments. The main purpose of this chapter is to review the general philosophies of the different types of positioning system technologies, environment modeling technologies, and Web Services. Firstly, different types of positioning systems are compared and particular emphasis is placed on UWB RTLS, the type of IPS used in this research. Then, three different types of environment modeling technologies are introduced focusing on WMS application, a prevailing environment modeling technology used in this research to simulate monitored environments for implementing awareness visualization as a supporting component. Finally, the introduction of Web services is reviewed focusing on the technologies of Web services and Web applications, and the comparisons of attributes and characteristics of two Web frameworks.

2.2 Positioning Systems

Positioning systems are a mechanism for determining the locations of objects in monitored sites. In this research, UWB RTLS, one type of positioning systems, is used to locate and track tagged objects moving indoors.

2.2.1 Global Positioning System (GPS)

After the incident of Korean Air Lines Flight 007 on September 1st, 1983, American President, Ronald Reagan, ordered the U.S. military to make GPS available for civilian use, free of charge. GPS, an American space-based global navigation satellite system, provides reliable positioning, navigation, and timing services to worldwide users on a continuous basis in all weather conditions. These services can be received at any location, on or near the Earth, which has an unobstructed view of four or more GPS satellites. GPS consists of three main components: (1) the space component ─ GPS satellite, (2) the control station component, and (3) the user component ─ GPS receiver. There are close to thirty GPS satellites orbiting the earth and broadcast radio signals from space. Radio signals, indicating time, are processed by GPS receivers to provide 3D location (latitude, longitude, and altitude) and precise current time (GPS, 2010). When a radio signal is received by a terrestrial GPS receiver, the difference between its transmission time and arrival time is noted (Cerruti et al., 2008). The distance from the receiver to the GPS satellite can then be calculated from that time difference. This calculation is performed using signals from at least four GPS satellites-for pinpointing a location on the earth. This working principle is illustrated in Figure 2-1. In addition, the installation and uninstallation of GPS receivers are simple on equipment from different manufacturers.

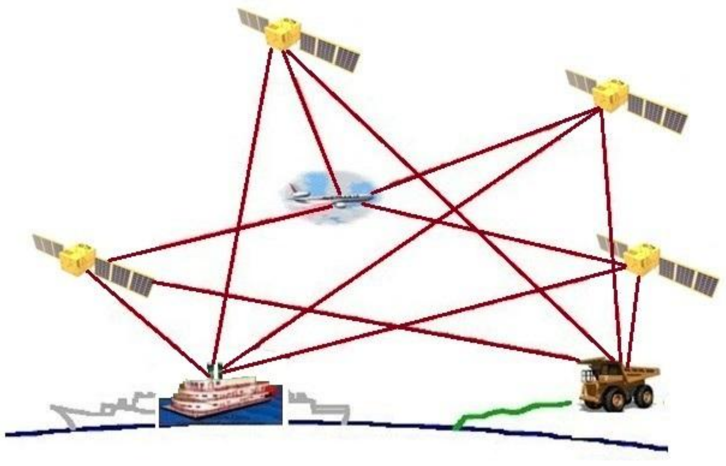


Figure 2-1 GPS Navigation

In modern industries and daily life, there are lots of various indoor activities where position information is very useful. Unfortunately, GPS can only be used in areas with a clear view of the sky. GPS receivers are unable to receive radio signals blocked by obstacles. Therefore, GPS has limited accuracy in indoor environments due to the excess loss of signals and multipath effect (Teizer et al., 2008).

2.2.2 Indoor Positioning Systems (IPSs)

IPSs overcome the limitations of GPS to locate and track objects indoor. IPSs are able to continuously determine the real-time locations of objects in a given physical space for navigation and data collection (Gu et al., 2009). Monitored objects need to be tagged with labels, tags, tokens or transponders to enable locating or positioning. IPSs have been used in various cases in industry and daily life. For example, the health care industry can use IPS to monitor patients under care or hospital equipment, the inventory management can use IPS to track cargo or material handling equipment in warehouses, and emergency services can use IPS to locate survivors confined in burning buildings (Reuters, 2009).

Indoor Messaging System (IMES)

Recently, IMES, with a new specification and technology to implement IPS, was developed by the Japan Aerospace Exploration Agency (JAXA) to provide accurate positioning indoors where reception of GPS is blocked or unreliable. IMES uses extremely low power radio as defined by radio regulations and thus does not require a license to deploy or operate. As illustrated in Figure 2-2, IMES provides seamless location-information infrastructure which links indoors and outdoors during normal times and emergency times (Hitachi, 2010).

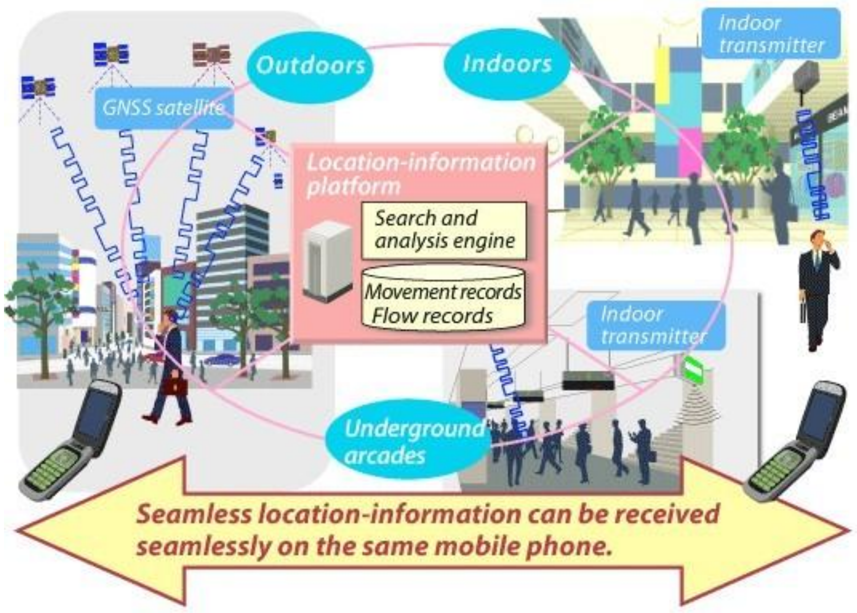

Outdoors
Indoors
Indoor transmitter
GNSS satellite
Location-information platform
Search and analysis engine
Movement records
Flow records
Indoor transmitter
Underground arcades
Seamless location-information can be received seamlessly on the same mobile phone.

Figure 2-2 Seamless Location - Information Infrastructure (Hitachi, 2010)

Transmitters are mounted indoors to create seamless positioning. Location information of indoor transmitters (including latitude, longitude, altitude, floor number, etc.) is transmitted as a message via a radio signal. When GPS receivers receive a radio signal from one of those transmitters, the positions of GPS receivers can be calculated from location information. In addition, with a certain amount of software modification, GPS receivers enable seamless positioning between indoors and outdoors (Hitachi, 2010). The overview of IMES is illustrated in Figure 2-3.

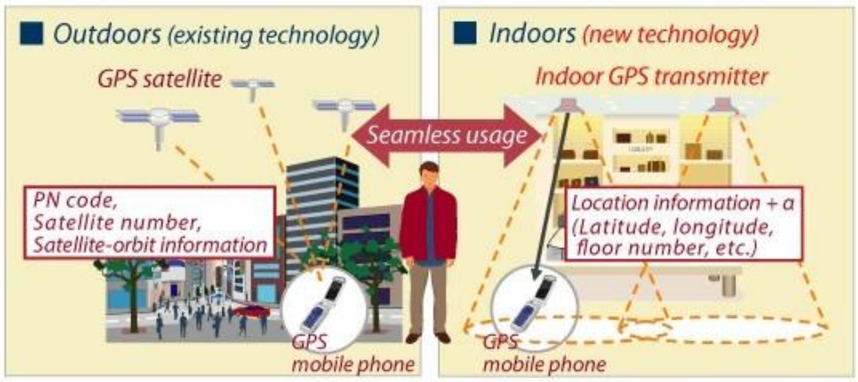

Outdoors (existing technology)
GPS satellite
PN code, Satellite number, Satellite-orbit information
GPS mobile phone
Seamless usage
Indoors (new technology)
Indoor GPS transmitter
Location information + α (Latitude, longitude, floor number, etc.)
GPS mobile phone

Figure 2-3 Overview of IMES (Hitachi, 2010)

The advantage of IMES is that the same GPS equipment can be used for indoor and outdoor positioning. IMES provides the same service as GPS satellites for all people with GPS receivers. Therefore, regardless of being inside or outside, objects can be located and even given navigation support with GPS receivers. On the other side, the disadvantage of IMES is that it is not designed for object monitoring and monitoring result sharing. Because data are calculated and collected by GPS receivers, only the holders of GPS receivers have access to the data. Therefore, although IMES is a good system for indoor navigation, the technology is not suitable to monitor objects and share data with others.

2.2.3 Real-Time Location Systems (RTLSs)

RTLSs are designed to track and identify the position of objects in real-time. Those objects are commonly tagged by simple and inexpensive tags (badges). These tags receive and transmit wireless information to a set of sensors (readers) to determine their locations. RTLS typically refers to systems which provide passive or active (automatic) collection of location information (Clarinox, 2009).

Radio Frequency Identification (RFID)

RFID is a technology used for identifying, locating, and tracking objects by exchanging data between readers and electronic tags attached to the objects via radio frequency waves. RFID has found a rapidly growing market by improving productivity and safety in an increasing variety of enterprises. Its competitive advantage has made RFID become one of the most widespread indoor positioning technologies of the 21st century.

During the 70s, several research laboratories and universities, such as the Los Alamos Scientific Laboratory and Northwestern University, became deeply involved in RFID research (Landt, 2005). In 1987, the leading commercial RFID application was developed and implemented in Norway. This application was then improved by the Dallas North Turnpike in the United States in 1989 (Domdouzis et al., 2007).

As illustrated in Figure 2-4, RFID is comprised of two basic components: (1) transceiver (also known as readers or antenna) and (2) tags (also known as labels).

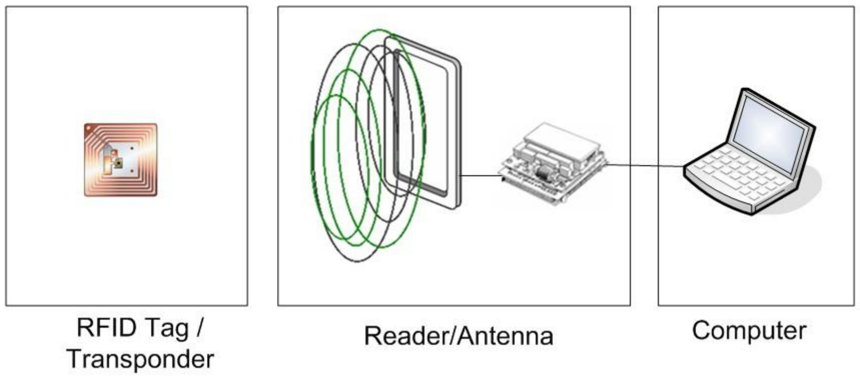


Figure 2-4 RFID Components (Motamedi, 2009)

RFID tags must contain at least two components: (1) an integrated circuit, which is designed to process and store information, modulate and demodulate radio-frequency (RF) signals and other specialized functions; and (2) an antenna, which is mounted to receive and transmit RF signals. There are generally three types of RFID tags: (1) active RFID tags, (2) passive RFID tags, and (3) battery assisted passive (BAP) RFID tags. Active tags contain an internal battery to supply power and can transmit RF signals autonomously in a longer read range compared with passive tags. Some of them even have memory to memorize data and configuration setting. They are commonly bigger, more expensive and have a limited operational lifetime based on operating temperatures and battery type.-Passive RFID tags have no battery so they require an external source or operating power generated from the transceiver to provoke signal transmission. They are lighter, less expensive and offer a theoretically unlimited operational lifetime. BAP RFID tags require an external source to wake up but have significantly higher forward link capability which provides greater range. Recently, much smaller RFID tags have been proposed. RFID tags are built in the form of labels and adhered to the objects which are going to be managed and tracked.

Ultra-Wideband (UWB)

UWB, a radio technology, proposes a new approach for carrying high data rates (480 to 500 Mbps) at very low power levels (less than 0.5 mill watts) for short-range (over 70 meters operating range) high-bandwidth communications with little interference by using a large portion of the radio spectrum (Zhang et al., 2009). A great number of applications are already envisioned or being created for UWB to target precision locating, sensor data collection, and tracking applications in diverse realms, such as emergency services, asset tracking, search and rescue, manufacturing inventory management, and medical patient monitoring (Colwell, 2008).

UWB has several distinct advantages over other traditional technologies. One major benefit is that UWB signals are transmitted across a much wider frequency than other traditional systems. The amount of spectrum occupied by a UWB signal ─ the bandwidth of the signal ─ is at least 25% of the center frequency. Thus, a UWB signal centered at 2 GHz would have a minimum bandwidth of 500 MHz and the minimum bandwidth of a UWB signal centered at 4 GHz would be 1 GHz (Malik, 2009). The most common technique for generating a UWB signal is to transmit pulses with durations less than 1 nanosecond. Therefore, UWB signals go undetected and are secure, because of their low frequency.

Distance or Angle Measurements

As illustrated in Figure 2-5, employing UWB technology with RTLS results in effective measurement of distance or angles between known sensor locations and unknown tagged objects. There are four common distance or angle computations distinguished by a variety of input data, which is used either alone or in combination for estimating the distance or angle between the tag and the reader (Clarinox Technologies Pty Ltd, 2009): (1) Angle of Arrival (AOA); (2) Time Difference of Arrival (TDOA); (3) Received Signal Strength Indicator (RSSI); and (4) Time of Arrival (TOA) (Correal et al., 2003). The comparison of AOA, TDOA, and RSSI can be found in Appendix A. The specific UWB RTLS used in this research is the Ubisense system, which employs both AOA and TDOA. These two methods are described briefly in the following.

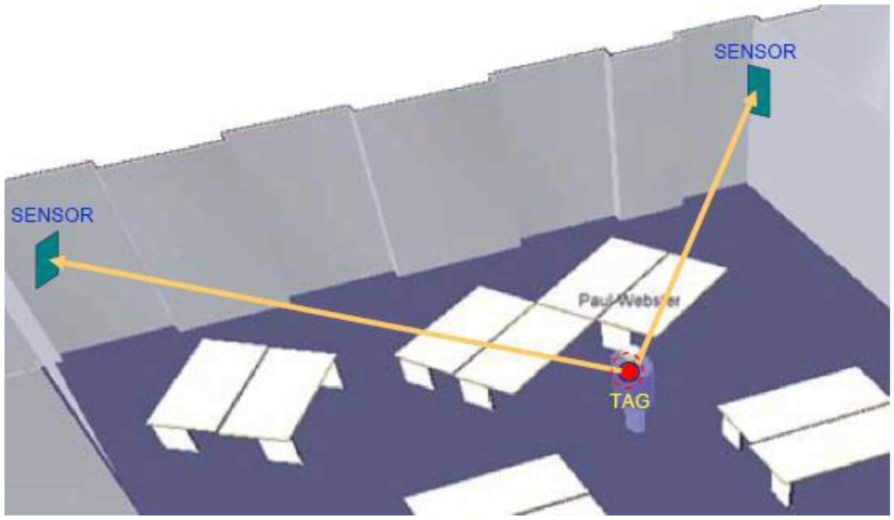


Figure 2-5 UWB Technology Employed in RTLS

In the AOA method, the angle of arrival of the signal sent by the object to be positioned is measured at several stationary receivers. Each measurement forms a radial line from the receiver to the object to be positioned. In 2D positioning, the position of the object is defined at the intersection of 2D lines of bearing. This method has the advantage of neither requiring synchronization of the receivers nor an accurate timing reference. On the other hand, receivers require regular calibration in order to compensate for temperature variations and mismatches (Ghavami et al., 2004). A triangulation method may be used to form a location estimate of the tag at the intersection of these lines. In theory, direction-finding systems require only two receiving sensors to locate a tag. However, in practice, there is often a need for more than two references to improve accuracy, finite angular resolution, and reduce multipath and noise (Muñoz et al., 2009). The AOA positioning principle for three sensors is illustrated in Figure 2-6.

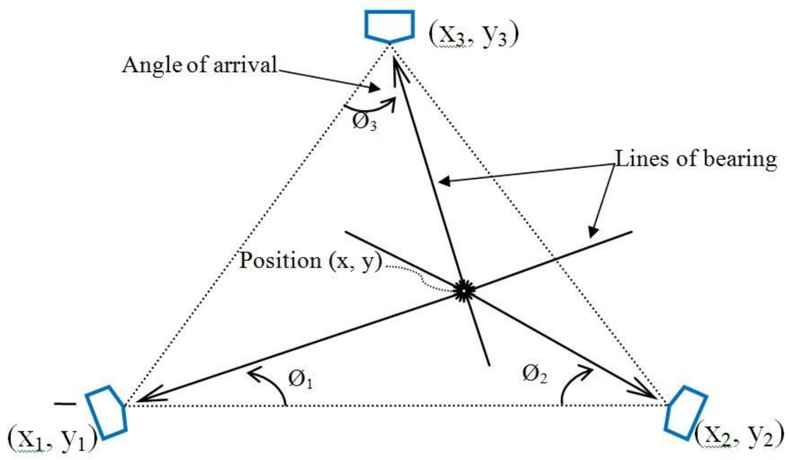


Figure 2-6 AOA Positioning Principle (adapted from Muñoz et al., 2009)

In the TDOA method, the difference in time at which the signal from the object to be positioned arrives at two different receivers is measured. The difference is then converted into a hyperboloid with a constant distance difference between the two receivers. In two dimensional positioning, at least two pairs of receivers are required; the position is the intersection of the two corresponding hyperboloids. This technique requires synchronization of the receivers' clocks. The position is found by solving equations if the coordinates of the three receivers are known (Ghavami et al., 2004). The TDOA positioning principle for three sensors is illustrated in Figure 2-7.

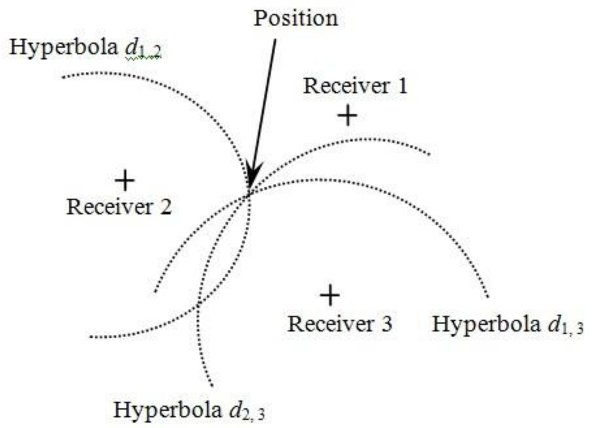


Figure 2-7 TDOA Positioning Principle (Ghavami et al., 2004)

AOA and TDOA are used in UWB RTLSs to locate tags based on trilateration. However, these methods are affected by obstacles or reflections in indoor environments. In practice, if the AOA method is used individually, although only two sensors are theoretically required to locate a tag in 3D, more sensors are needed to reduce the influence of multipath and noise for accuracy improvement (Muñoz et al., 2009). On the other hand, if the TDOA method is used individually, at least three sensors are required for 2D positioning and four sensors for 3D positioning (Ghavami et al., 2004). Using a combination of AOA and TDOA can obtain a high accuracy in location measurements (Abdul-Latif et al., 2007). The comparison of the methods of distance or angle measurements and their combination is summarized in Table 2-1.

Table 2-1 Comparison of AOA, TDOA, and their Combination (Rodriguez, 2010)

Location Method	**Number of Sensors Detecting Tag**	**Other Information Required**	**Result**
AOA	1	known height of tag	2D horizontal position (known height)
AOA	2 or more	none	3D position
TDOA	3	known height of tag	2D horizontal position (known height)
TDOA	4 or more	none	3D position
AOA + TDOA	2 or more	none	3D position (highest accuracy)

2.3 Environment Modeling Technologies

Environment modeling technologies are used to simulate monitored environments and provide a reference model coordinate system with 3D coordinates to uniquely determine the positions of architectural models or geometric elements.

2.3.1 Geographic Information System (GIS)

As illustrated in Figure 2-8, GIS is the integration of hardware, software, people, methods, and data for capturing, managing, analyzing, and displaying all forms of geographically referenced information. GIS allows users to view, understand, question, interpret, and visualize data in many approaches which reveal relationships, patterns, and trends in the form of maps, globes, reports, and charts. GIS also helps to answer questions and solve problems by looking at data in a way that is quickly understood and easily shared (ESRI, 2010).

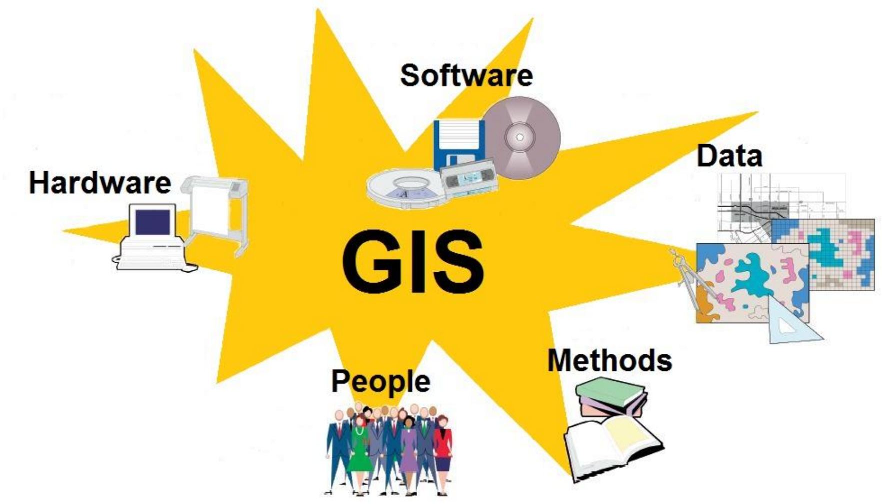
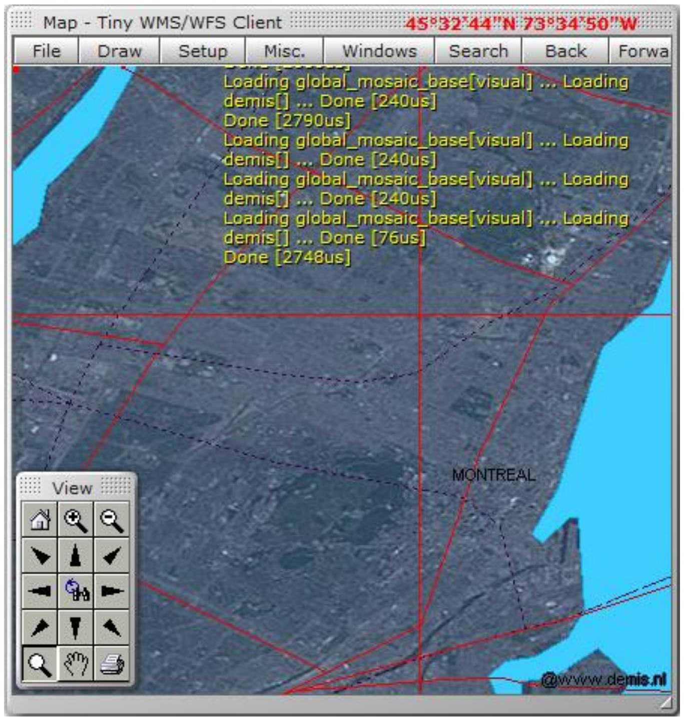


Figure 2-8 Architecture of GIS (adapted from City of Sheridan)

GIS Web Services

Users can conveniently obtain location data from GIS Web Services, which have the potential to revolutionize the way in which GIS is developed, accessed, and used. These services make it easier to share geographic data and functionality, and for GIS to be deeply integrated into other technologies. As described in Subsection 2.4.1, Web services are software components which can be accessed through the World Wide Web (WWW) and used by other applications. As illustrated in Figure 2-9, a Web application uses the services of a GIS Web service. GIS Web services provide spatial data or functionality on the WWW. They make it possible for users to access GIS data and functionality through the Web and to integrate them with their own systems and applications without the need to develop or host specific GIS tools and data sets themselves (Tang and Selwood, 2003).



```
Loading global_mosaic_base[visual] ... Loading
demis[] ... Done [240us]
Done [2790us]
Loading global_mosaic_base[visual] ... Loading
demis[] ... Done [240us]
Loading global_mosaic_base[visual] ... Loading
demis[] ... Done [240us]
Loading global_mosaic_base[visual] ... Loading
demis[] ... Done [76us]
Done [2748us]
```

Figure 2-9 GIS View in GEAbios (adapted from GEAbios, 2010)

2.3.2 Web Mapping Service (WMS)

WMS is a standard protocol and a widely supported format for serving geo-referenced map images generated by a map server using data from GIS databases over the Internet (Open Geospatial Consortium, 2009). The specification of WMS was developed and first published by the Open Geospatial Consortium in 1999, which is a non-profit, international, voluntary consensus standards organization leading the development of standards for geospatial and location based services (Scharl and Tochtermann, 2007).

As illustrated in Figure 2-10, the WMS application has the advantage of displaying 3D building models with the features to rotate and tilt the view angle, to pan, and to zoom. The 3D building model, which is based on a mathematical representation of any 3D surface and inner structure of architectural objects, can be displayed as a 2D image through the 3D rendering process or used in the computer simulation of physical phenomena (3D Modeling, 2010). In addition, the 3D building model is textured using composites of aerial photography in order to achieve near-photorealism.

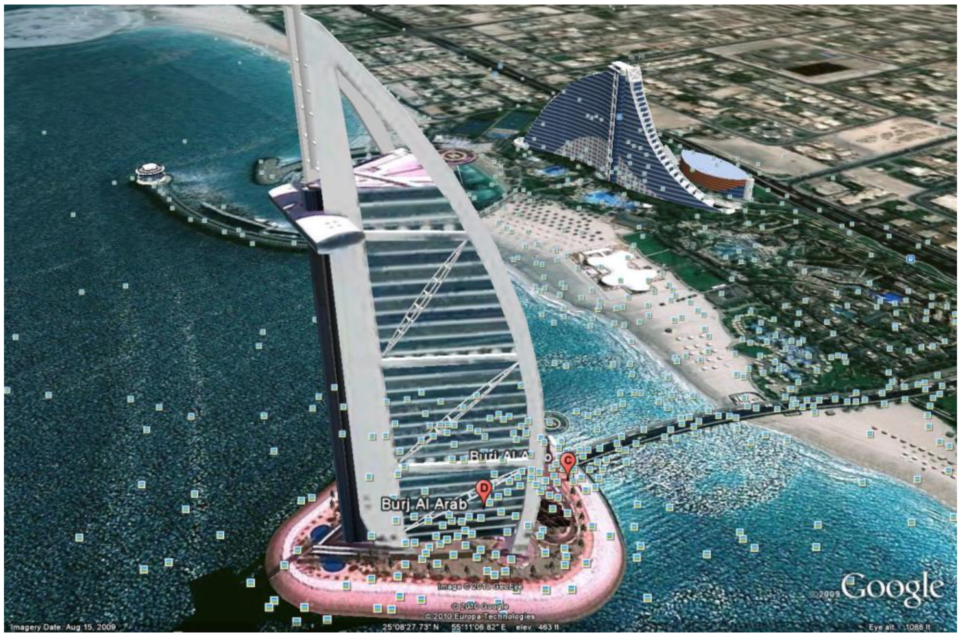


Figure 2-10 3D Building Model View in Google Earth (adapted from Google Earth, 2010)

In this research, the WMS application plays a supporting role for the Web service by simulating monitored environments and providing a reference model coordinate system for implementing awareness visualization.

2.3.3 Building Information Modeling (BIM)

As illustrated in Figure 2-11, BIM is the process of generating and managing architectural information data during the life cycle of a building (Lee et al., 2006). Typically it helps to increase productivity in building design and construction by using 3D, real-time, dynamic building modeling software (Gordon and Holness, 2008). BIM is an innovative method which seamlessly bridges communication within the architecture, engineering and construction industries. With BIM, architects and engineers can efficiently generate and exchange information, create 4D digital displays of all stages of the building process, and simulate real-world performance, such as streamlining workflow, increasing productivity and improving quality (Autodesk, 2010). By encompassing building geometry, spatial relationships, geographic information, and building component quantities and properties, BIM can provide an integrated visualization of 2D and 3D building models at any step of the building life cycle.

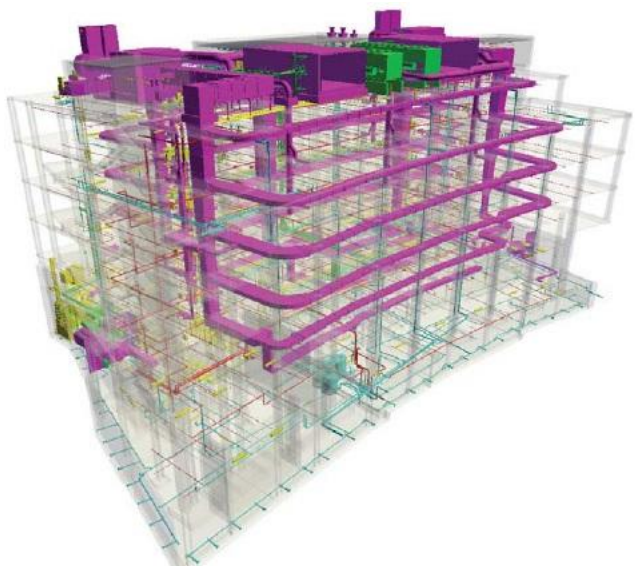


Figure 2-11 BIM View (Khemlani, 2007)

2.4 Web Services

2.4.1 Review of Web Services

Web services are self-describing, self-contained software modules which are available via public or private networks. Web services can complete tasks, solve problems, or conduct transactions on behalf of users or applications. Web services are comprised of a distributed computer infrastructure made up of many different interacting application modules trying to communicate over the Internet or intranets to virtually form a single logical system (Papazoglou, 2008). Web services are collections of functions which are packaged as a single entity and published to the network to be used by other applications. They build middleware for generating open distributed systems to allow companies and individuals to quickly and inexpensively make their digital assets available worldwide (Glass, 2000). Furthermore, they have the potential to serve as an environment for facilitating collaborative scenarios (Kerer et al., 2004). For example, if a group of developers have an existing application and want to make its services available to others ─ either only within their own organization or beyond it ─ they can use the technologies of Web services to provide a standard Web interface for the services. Under this method, Web services can be defined as middleware. Anyone can connect applications together no matter how each application is implemented or where it is located.

Web services are a typical type of Web application. Web applications are accessed over networks (both the Internet and intranets). Web applications are hosted in a browser-controlled environment (e.g. ASP.NET), coded in a browser-supported language (e.g. JavaScript), combined with a browser-rendered markup language (e.g. HTML), and reliant on a common Web browser to render the application executable.

2.4.2 Technologies of Web Services and Web Applications

Web Service Technologies

The technologies of Web services are able to connect by using open standards. Web services operate at a level of abstraction, which is similar to the Internet, in which they can work with any operating system, middleware, hardware platform, or Web-enabled programming language (IBM, 2005). Therefore, depending on the needs of the developer, there exists a wide array of choices for the creation, assemblies, and deployments of Web services. The following technologies are open industry initiatives which have gained the most industry acceptance for performance of Web services: (1) eXtensible Markup

Language (XML), (2) Universal Description, Discover, and Integration (UDDI), (3) Web Services Description Language (WSDL), and (4) Simple Object Access Protocol (SOAP).

a) eXtensible Markup Language (XML)

XML is a set of specifications for encoding documents electronically. XML enables heterogeneous computing environments to share information over the World-Wide Web (IBM, 2005).

b) Universal Description, Discover, and Integration (UDDI)

UDDI is a platform-independent and XML-based registry. UDDI helps users to find services by searching lists of businesses worldwide on the Internet. Developers can publish Web services and enable software to search for Web services offered by others (IBM, 2005).

c) Web Services Description Language (WSDL)

WSDL is an XLM-based language. WSDL provides a model for describing and defining Web services. Developers can use WSDL documents to describe Web services to others (IBM, 2005).

d) Simple Object Access Protocol (SOAP)

SOAP is a protocol specification for exchanging structured information in the implementation of Web services in computer networks. SOAP relies on XML format for representing parameters, and other application layer protocols for message negotiation and transmission (e.g. return values over HTTP). Developers can use SOAP to bind applications to Web services and invoke service operations (IBM, 2005).

As illustrated in Figure 2-12, there are six working processes of Web services: (1) a client queries the registry to locate a service; (2) the registry refers the client to a WSDL document; (3) the client accesses the WSDL document; (4) the WSDL provides data to interact with Web services; (5) the client sends SOAP message request; and (6) the Web service returns a SOAP message response (Janelli, 2005).

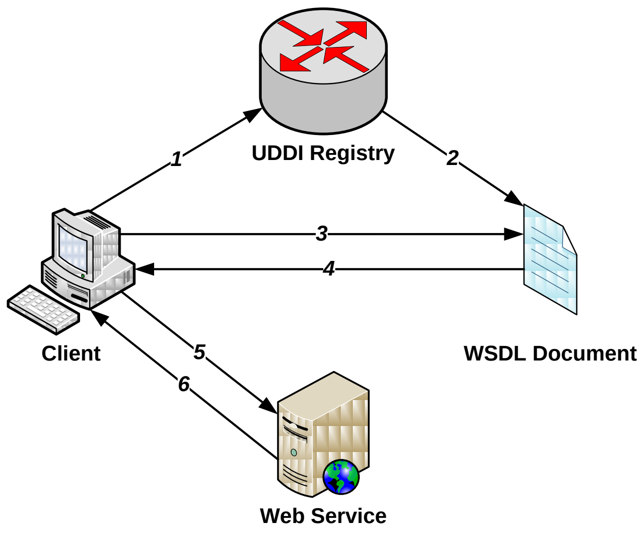


Figure 2-12 Processes of Web Services

Web Application Technologies

Recently, the technologies of Web applications have been developed to coordinate client-side scripting with server-side technologies (e.g. PHP and Ajax), which create more interactive experiences. Developers prefer client-side scripting to add functionality and especially create interactive Web applications which do not require page reloading. The following are interrelated Web development techniques: (1) Hypertext Preprocessor (PHP) and Asynchronous JavaScript and XML (Ajax).

a) Hypertext Preprocessor (PHP)

PHP is available as either a processor for most modern Web servers or a standalone interpreter on most operating systems and computing platforms. PHP code is embedded into the HTML source document and interpreted by a Web server with a PHP processor module generating the Web page document (IBM, 2005).

b) Asynchronous JavaScript and XML (Ajax)

Ajax leads to an increase in interactive or dynamic interfaces on Web pages by enabling Web applications to retrieve data from a server asynchronously in the background without interfering with the display and behavior of the existing page (IBM, 2005).

2.4.3 Web Framework Comparison

There are two Web frameworks of evolutions of existing application server technologies: (1) Java Platform, Enterprise Edition (Java EE) and (2) Microsoft's .NET Platform. For the sake of understanding Java EE and .NET by analogy, their features are discussed below.

Java EE

The architecture of Java EE is based on the Java programming language. The source code of Java needs to be converted into bytecode at compile-time, and then interpreted by Java Runtime Environment (JRE) as native executable for a specific machine at run-time. The Java EE application is hosted within a container. In the development model of Java EE, there is a container which is designed to provide necessary services for Web applications, such as transactions, security, maintenance, and persistence services. Enterprise JavaBeans (EJB) performs business process and data logic. There are various connection technologies of EJB: to databases is Java Database Connectivity (JDBC) and SQL/J; to existing systems is Java Connector Architecture (JCA); and to business partners with Java EE applications are some of Web services technologies mentioned above, such as SOAP, UDDI, WSDL, and Electronic Business using eXtensible Markup Language (ebXML) through the Java APIs for XML (JAX APIs). JAX API is used by a "servlet", a request/response oriented Java object, to perform Web services operations, such as acceptance of Web service requests from business partners. The Web services development model of Java EE is illustrated in Figure 2-13 (Oracle, 2009).

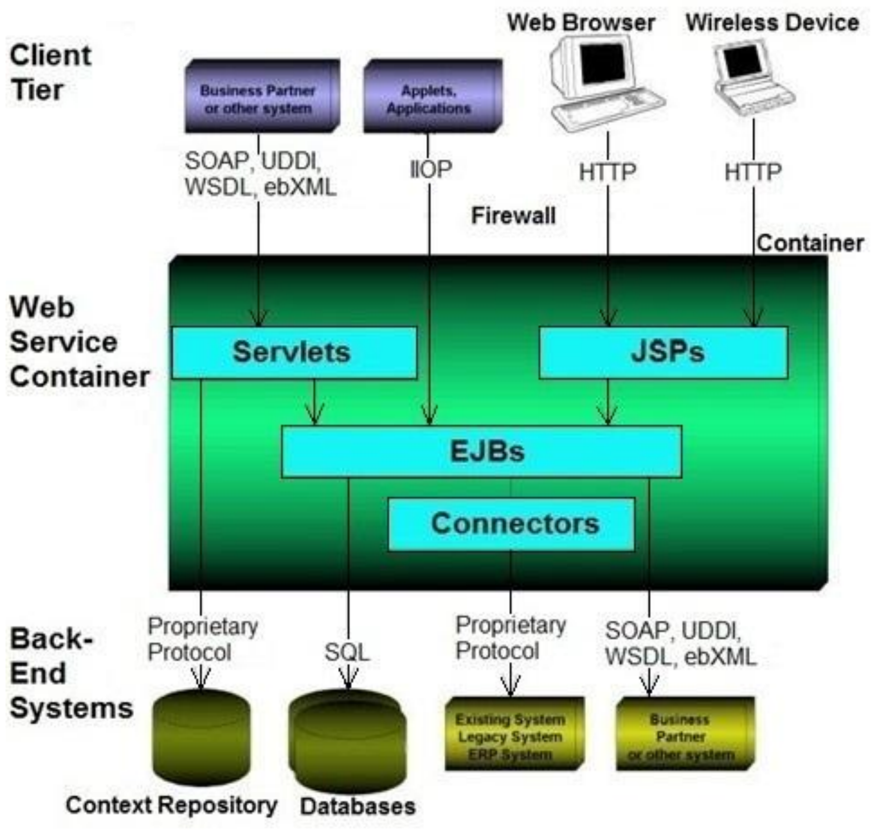


Figure 2-13 Web services development model of Java EE (Vawter and Roman, 2001)

.NET Framework

The most intriguing and fundamental features of the .NET platform are supporting independence and interoperability of language. The component of .NET can be written in Visual Basic.NET (VB.NET) and C#, both of which are object-oriented computer programming languages by Microsoft. The conversion, interpretations, and execution of the source code of .NET are analogous to Java EE framework. Instead of byte code in Java and JRE, its source code is converted into Microsoft's Intermediate Language (IL) at compile-time, and interpreted by Common Language Runtime (CLR) as native code at run-time. Similar to Java EE applications, .NET applications are also hosted within a container. The function of .NET Managed Components is similar to that of EJB's, which is to perform~~s~~ business processes and data logic. However, the connection technologies of .NET differ from Java EE: to databases is Active Data Objects (ADO.NET); to existing systems is Microsoft Host Integration Server 2000, such as the COM Transaction Integrator (COM TI); and to business partners with .NET applications are the same Web services technologies as Java EE. The Web services development model of .NET is illustrated in Figure 2-14 (Microsoft, 2009).

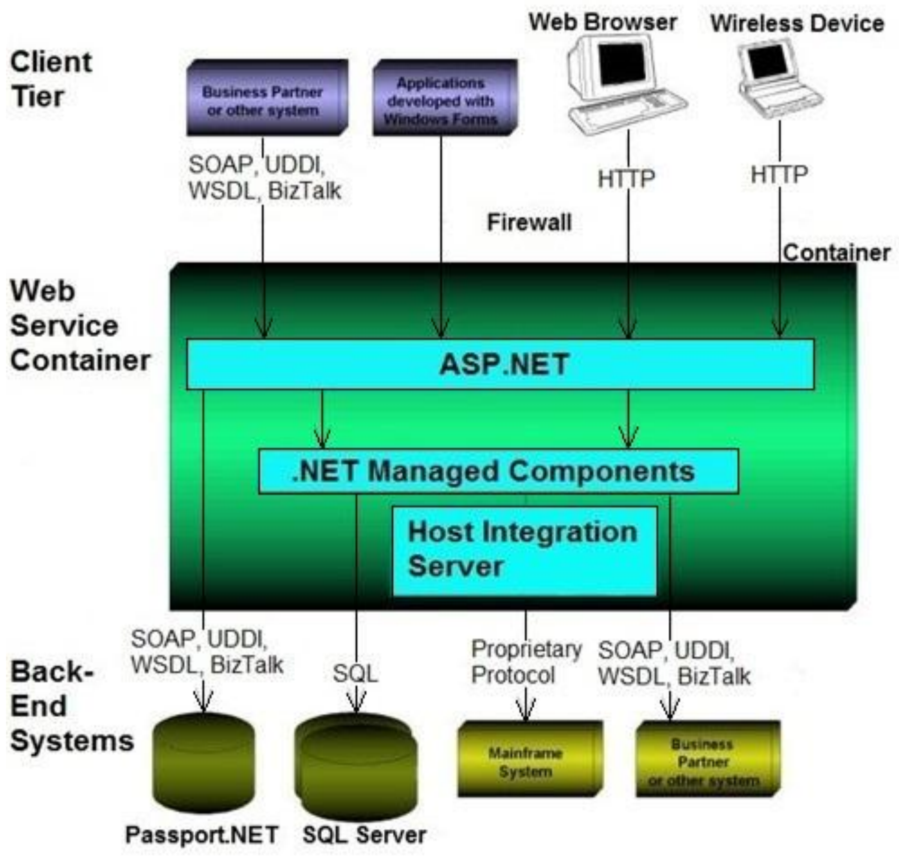


Figure 2-14 Web services development model of .NET (Vawter and Roman, 2001)

2.5 Summary

In this chapter, the three primary groups of components of this research are reviewed: (1) UWB RTLS, which is a type of positioning system employed to provide indoor positioning and navigation; (2) WMS application, which plays a supporting role for the Web Service by simulating monitored environments and providing a reference model coordinate system for implementing awareness visualization; and (3) Web services.

CHAPTER 3 REQUIREMENT DEFINITION

3.1 Introduction

The main purpose of this chapter is to define the common requirements of UWB RTLS applications, and then determine the requirements of the Web service for UWB RTLS according to these common requirements. In this research, the Web service is named ***UWB RTLS Web Service***. Firstly, this chapter discusses the general requirements of UWB RTLS applications. Secondly, it explores the specific requirements of three typical UWB RTLS applications (employee monitoring, fire rescue, and crane operation) by analyzing their divisions of problems and situations to define the common requirements of UWB RTLS applications. Thirdly, it discusses the advantages of Web services for UWB RTLS applications. Finally, it determines the requirements of the ***UWB RTLS Web Service***, which are categorized into four parts based on the common requirements: (1) requirements of technology integration and collaboration, (2) requirements of remote interaction, (3) requirements of awareness visualization, and (4) requirements of event detection. Finally, it analyzes six fundamental use cases of the ***UWB RTLS Web Service***: (1) get real-time location data, (2) get visualization data, (3) get event messages, (4) save real-time location data, (5) save event messages, and (6) collect monitoring result.

3.2 General Requirements of UWB RTLS Applications

This research discusses the following general requirements of various UWB applications for better productivity and safety: (1) accuracy requirements, (2) real-time location data filtering, (3) visibility requirements, (4) scalability and real-time requirements, (5) tag form factor requirements, (6) power requirements, and (7) networking requirements. The number of sensors and tags, location of sensors and tags, and orientation of sensors should also meet these requirements.

1) Accuracy Requirements

Accuracy is the most crucial requirement to guarantee that collected real-time location data are valuable. As described in Subsection 2.2.3, in order to get the highest possible accuracy, the combination of AOA and TDOA should be applied (Abdul-Latif et al., 2007). Two UWB RTLS sensors deliver a robust localization with an accuracy of up to 15 cm in ideal conditions. In practical applications, more sensors enable greater confidence in the accuracy and higher availability leading to a more robust solution (Rodriguez, 2010).

To gain accurate real-time location data, calibration of the sensors is essential. A local coordinate system is defined by the user based on the coordinates of each sensor which should be measured precisely using surveying tools, such as total stations. Each sensor should be levelled after the installation. A tag should be placed at a location with known coordinates in the local coordinate system. As a result, the pitch and yaw angles of each sensor can be calculated and recorded in the system.

2) Real-Time Location Data Filtering

Real-time location data filtering should be applied to improve the accuracy by decreasing errors in near real time. This filtering can validate the results of the individual AOA and TDOA measurements against predicted positions, and then calculate new estimated positions. In this case, the motion model for the filter has to be defined by specifying the constraints on the motion which tracked objects will undergo. For example, a tag could be free to move in 3D or constrained to move horizontally with a certain motion model of position and velocity and Gaussian noise on velocity. Filtering can be applied on real-time location data resulting from the trilateration (Rodriguez, 2010). For example, total accuracy can be improved by 25% after applying an error model using the Kalman smoother (Cho et al., 2010). However, in applying these filters, several assumptions are made about the motion model. These assumptions may not be very realistic for UWB RTLS applications.

3) Visibility Requirements

Sensors should be mounted in a way that utilizes their antenna pattern both in the azimuth and the elevation. The field of view may be different from one UWB system to another. The maximum range of sensors can be potentially up to 60 m. Therefore, a reasonable monitoring area should be defined considering the coverage of the cell. If the area to cover is large, more sensors should be mounted to cover this area using one or more cells. In addition, multiple tags can be attached to one object as a way to improve the visibility of the object by increasing the probability of detecting these tags on it.

4) Scalability and Real-Time Requirements

Each tag will be registered with its containing cell, and inserted into the schedule for this cell. The schedule determines when tags transmit UWB signals and are located by sensors. The schedule is arranged so that each tag is give optimal attention relative to its requested quality of service, and sufficient space is maintained in the schedule for new tags to register. When a tag transmits a UWB signal, this signal is picked up by one or more sensors in the cell (Wombacher, 2008).

The ideal number of tags in a cell should be based on the frequency of the UWB RTLS and the size of the cell. The number of timeslots per second depends on the operational frequency of the UWB RTLS. For example, the Ubisense system, a type of UWB RTLS, has a nominal operational frequency of R = 160 Hz, and each second is divided into 153 timeslots, which means each timeslot has a duration of 6.5 ms. Different slot intervals can be selected to determine how often the tags' locations are updated, and how often the system listens for data and schedule messages from the master sensor. As illustrated in Figure 3-1, the Ubisense system has a shortest slot interval of four timeslots, which means the update interval is 26 ms, corresponding to a maximum update rate per tag of approximately 38 Hz (Rodriguez, 2010). The relationship of slot interval, update interval, and update rate for a 160 Hz System is summarized in Table 3-1.

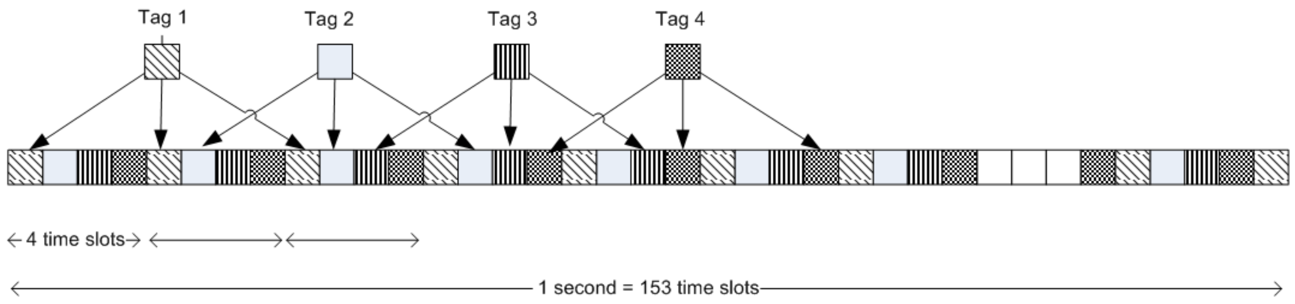


Figure 3-1 4 Timeslots of Slot Interval for a 160 Hz System (Rodriguez, 2010)

Table 3-1 Relationship of Slot Interval, Update Interval, and Update Rate for a 160 Hz System (Rodriguez, 2010)

Slot Interval	**Update Interval (ms)**	**Nominal Update Rate for Each Tag (Hz)**
4	26	38
8	52	19
16	104	10
32	208	5
64	416	2.4
128	832	1.2
…	…	…

The update rate of tags will be reduced as the number of tags increases in order to allow the UWB RTLS to cover all tags with the fixed total number of timeslots. On the other hand, the more tags in UWB RTLS, the bigger the slot interval should be selected, and the lower the update rate. A specific update rate can be set for an individual tag or a group of tags (Rodriguez, 2010). For example, if the timeslot is set to 4 and only 4 tags are in the cell, the four tags are updated every 26 ms (38 Hz). If more tags are detected in the UWB RTLS, such as 8 tags, the update rate will be reduced to 19 Hz. Setting the update rate also depends on the velocity of moving objects. Objects with high velocity need more frequent updates to accurately track them. Therefore, selecting a suitable number of tags with an appropriate update rate based on their velocity is essential for achieving the balance between the conflicting requirements of visibility and accuracy in near real time. The heuristic rule which maximizes the update rate can be found in (Zhang, 2010).

5) Tag Form Factor Requirements

Although the fundamental functionality of tags is the same, they represent different form factors. Some tags are basically designed to be attached to objects or assets as labels, and others are designed to be worn by persons as badges. In addition to their tracking capabilities, tags can be equipped with push-buttons to trigger button events and a buzzer to provide basic messaging capabilities.

6) Power Requirements

Not only sensors need to be connected to a stable power source for precise measurements, but also UWR RTLS tags require a battery. Battery lifetime depends upon the update rate established for the UWR RTLS. The update rate of tags can be dynamically and automatically varied based on the activities of the tags. For example, if tags are moving fast, their update rate will be increased for better tracking; if tags are moving slowly, their update rate will be reduced for better battery lifetime; if tags are stationary, they will go to sleep mode to conserve power, and built-in motion detectors ensure that the tags can transmit again as soon as they are moved.

7) Networking Requirements

Sensors can be connected by cables or wirelessly to a location server. Both data cables and timing cables are needed for a wired system. However, wireless communication is not fast enough to support the TDOA method. Only the AOA method can use a wireless connection. The type of network (wired vs. wireless) has a direct impact on accuracy (Cho et al., 2010).

3.3 Typical UWB RTLS Applications for Specific Requirements Elicitation

For applying UWB RTLSs to business enterprises, there are three stages of evolution: (1) better information; (2) process improvement; and (3) business innovation (Heinrich, 2005). The business enterprises which carry out UWB RTLS research have to decide which method to accept and then build a definition of UWB RTLS business posture into its strategic plan (Poirier and McCollum, 2006). The process of determining business justification is: forming a business justification team, determining potential application areas, building business cases, determining priorities, and creating roadmaps (Lahiri, 2005). Those business enterprises can provide more services to not only management for improving productivity and safety, but also collaboration with various businesses. Therefore, the primary objective of the specific requirements elicitation from some typical UWB RTLS applications is to define the common requirement of UWB RTLS applications which is used determine the requirements of the Web service for UWB RTLS. So as to define the common requirements of UWB RTLS applications, three typical UWB RTLS applications are examined with respect to their divisions of needs, conditions, and problems. These three applications are: (1) employee monitoring, (2) fire rescue, and (3) crane operation.

3.3.1 Employee Monitoring

As a result of new technologies, there are many reasons why employee monitoring is important in the indoor workplace. Some examples include human management, asset management, security, and so on. Employers have many alternatives for monitoring what employees do while at work indoor, including the applications of video cameras, audio monitoring, keystroke logging, or email filters. These methods may involve limitations and legal issues depending on certain situations. This research proposes the use of the UWB RTLS for employee monitoring.

Requirements

In a typical employee monitoring scenario, a company owns a large number of employees, movable instruments, and visitors. For better human management, employers want to record employee attendance, hours, and routes. Such information can help employers to figure out whether their employees are working on schedule, attending meetings, and so on. For better asset management, companies want to track movable instruments, so that they can find out where movable instruments are and who are using. For better security, security personnel need to identify and locate visitors, so that they can be alerted when a visitor is entering into an unauthorized area.

Basic Design

In order to satisfy these requirements, the UWB RTLS is mounted in the company building so that its signals cover the work places. Different tags are designed to be attached to a diverse array of tracked objects including: (1) every employee with a tagged employee ID card; (2) every visitor with a tagged visitor ID card which is worn while visiting; and (3) some movable instruments. This idea is illustrated in Figure 3-2.

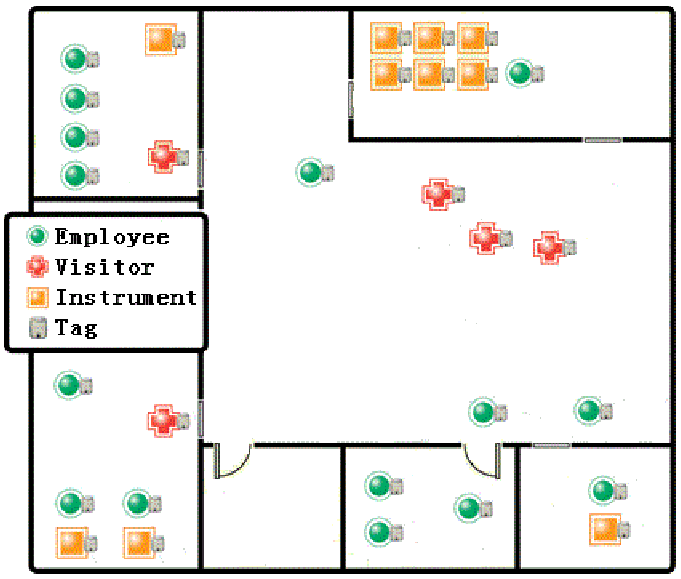


Figure 3-2 UWB RTLS Use in Employee Monitoring

After UWB RTLS detects the locations of employees, visitors, and instruments and generates real-time location data, their actions can be estimated. For example, when an employee is in his office, he is probably working; otherwise, if he is not in his office, he is probably taking a break outside; and when an employee is close to an instrument, he is probably using it. In addition, according to long term recording of the actions of employees, employers are able to understand employee work habits and efficiency which would help to plan a better strategy of human management.

The locations and actions of employees will be displayed under the four following major processes: (1) defining various actions of employees and visitors based on practical experience (such as coming, leaving, and using instruments); (2) attaching tags to employees, visitors, and instruments; (3) locating these tags by using the UWB RTLS; and (4) detecting useful events by logical analysis (such as an employee is coming when the tag on him/her appears in the monitored site, leaving when it disappears, or using an instrument when it gets close to another tag attached to the instrument).

3.3.2 Fire Rescue

There is no doubt that fire rescue can save a lot of lives in fire emergencies. Fire emergencies can result in devastating damages, traumatizing personal injuries, and even fatalities. Every year, billions of dollars in property and asset damages occur as a result of fire emergencies. Victims of fire emergencies may suffer serious harm, including burn injuries to their entire bodies. Deaths from fire emergencies can be caused not only from burns but also from smoke inhalation and toxic gases. The Centers for Disease Control and Prevention note that deaths from fires and burns are the fifth most common cause of unintentional injury deaths in the US and third leading cause of fatal home injury (Lawyers and Settlements, 2010).

According to the US Fire Administration (USFA) reports, 118 firefighters were killed on duty. Of those incidents, approximately 48 percent were outside or other fires, 16 percent were vehicle fires, and approximately 36 percent were structure fires (Lawyers and Settlements, 2010).

Requirements

In a typical fire rescue scenario, firefighters need to enter a burning building to locate fire root, remove dangerous items, or search for victims. For better command, the rescue commanders need to be aware of the position and situation of firefighters, such as where firefighters are searching for victims, how they can approach targets, whether they are facing danger, or whether they are injured. Such information can help the rescue commanders to better supervise firefighters, assign backup to specific positions, and save injured firefighters from danger.

Basic Design

In order to satisfy these requirements, the sensors of UWB RTLS may be equipped on fire engines to monitor the burning building. Two tags are attached to a firefighter: one on his helmet, and one on his belt. Furthermore, some tags can be attached to the fire-fighting tools, such as water jets, axes, fire extinguishers. This idea is illustrated in Figure 3-3.

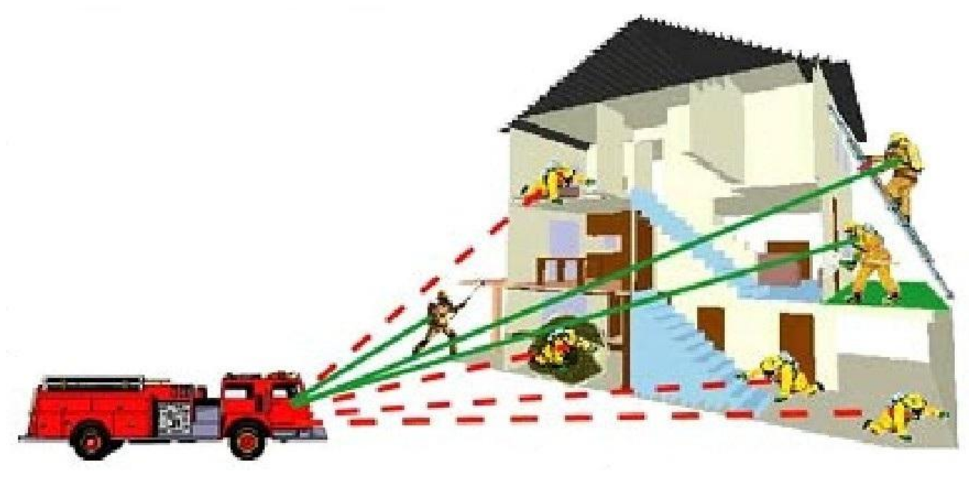


Figure 3-3 UWB RTLS Using in Fire Rescue (adapted from Fire Government, 2009)

When a firefighter enters into a burning building, the two tags attached to him can indicate his location to the rescue commander. When the two tags are in a near vertical position, it can be assumed that the firefighter is standing, walking, or jogging. On the other hand, when the two tags are in a near horizontal position, it can be assumed that the firefighter is lying or creeping, or might be injured. This approach can help the rescue commander to better obtain condition information, plan a rescue strategy, manage human resources and control unexpected situations.

The locations and actions of firefighters will be displayed under the four following major processes: (1) defining various actions of firefighters based on practical experience (such as walking, running, standing up, or lying down); (2) attaching tags to different parts of the firefighters (e.g. hardhats, belts, boots, and tools) in order to get relative positions and movements of the tags with respect to each other; (3) locating these tags by using the UWB RTLS; and (4) detecting useful events by logical analysis, mathematical calculation, and physical estimation (such as a firefighter is running when tags on him are moving faster or a firefighter is lying down when the tag on his helmet and the tag on his boot are almost at the same level).

3.3.3 Crane Operation

Crane operation is a good example to demonstrate an application of the UWB RTLS in the heavy construction industry. The crane is a lifting machine, generally equipped with a winder, wire rope or chains and sheaves, which can be used both to lift and lower materials and to move them horizontally. Cranes are ~~also~~ commonly used in ports for the movement of materials and in the manufacturing industry for assembling heavy equipment.

In the United States, there were 323 deaths related to crane operations during the period from 1992 to 2006 (Updated Crane Report, 2008). In Canada, the province of British Columbia had 72 accidents related to tower cranes and 142 accidents related to mobile cranes and boom trucks from 2005 to September 2008 (Work Safe BC, 2009). In the province of Quebec, there were 23 accidents with injuries, 26 accidents with death, and 13 accidents with material damage related to crane operations during the period from 1974 to 2002 (CSST, 2009). Furthermore, the number of reported accidents and the resulting deaths related to crane operations have been increasing during the past 10 years

(Crane Accident, 2009). Majority of these accidents were caused by contact with overhead power lines, workers struck by booms/jibs, struck by crane load, or caught in between objects.

Requirements

In a typical crane operations scenario, cranes are used in confined areas, such as close to buildings, or near another crane. For accident prevention, crane operators and workers need to be alerted by software applications in real time when their operations are approaching a potential collision, such as boom collisions or workers within the crane operation range (Carbonari et al., 2009). Hence, different parts of a crane need to be located for the simulation of the crane's actions.

Basic Design

In order to satisfy these requirements, several tags are attached to the boom, hook, and outriggers of a crane on different sides of the joints. Tags on the boom are used to measure the extended length and vertical or horizontal angle of the boom. Tags on the hook are used to measure the height of a lifted object. And tags on the outriggers are used to measure the position and direction of the crane. This idea is illustrated in Figure 3-4.

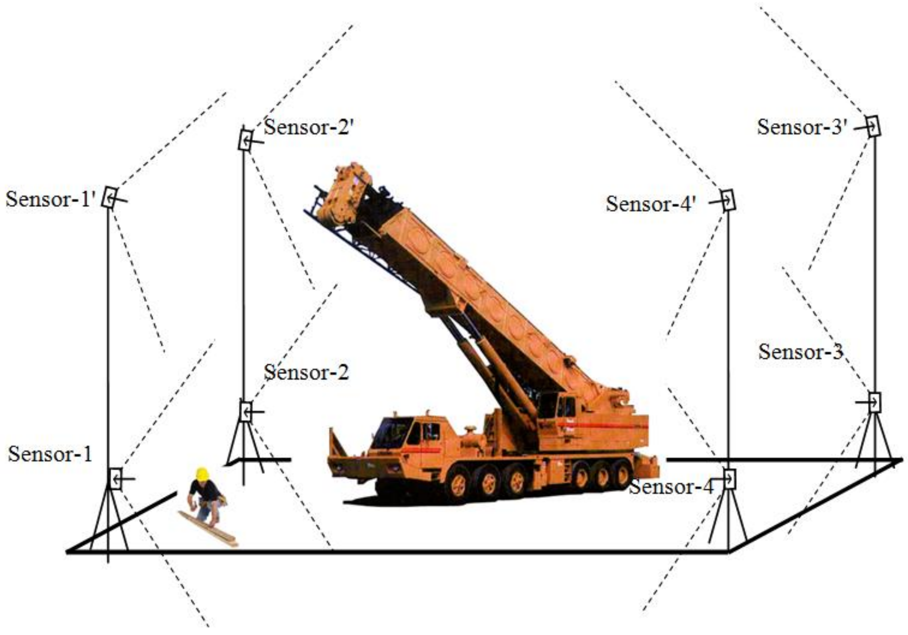


Figure 3-4 UWB RTLS Used in Crane Operation (Zhang, 2010)

When the relative positions and movements of tags on the crane are detected, the following functions can be performed: (1) operation control, which ensures that the operator properly follows the procedure manual, such as recording the operational processes and alerting the crane operator when he is operating inappropriately; and (2) collision prevention, which eliminates dangers resulting from potential collisions, such as giving an alarm when a boom gets too close to an obstacle, people are under the hook, or other potential collisions. The alarm may be signals from software applications running on mobile devices or buzzers in tags.

Alerts will activate when dangers are detected under the following four major processes: (1) defining various events of potential accidents based on practical experience (such as potential collisions with the boom of a crane); (2) attaching tags to different parts of cranes (e.g. boom, hook, and outriggers), workers, and other objects in order to get the relative positions and movements of tags with respect to each other; (3) locating these tags by using UWB RTLS; and (4) detecting useful events by logical analysis, mathematical calculation, and physical estimation (such as the X, Y coordinates of tags on a hook and a person are almost the same, and Z coordinate of the tag on the hook is higher than the tag's on the person, which means that the person is under the hook).

3.4 Advantages of Web Services for UWB RTLS

Developing Web services for UWB RTLS is considered as the best approach for enhancing its capabilities and satisfying the common requirements of UWB RTLS applications described in Section 3.3. Web services have the following advantages for UWB RTLS: (1) technology integration and collaboration, (2) remote interaction, (3) facilitating collaborative scenarios, and (4) low cost.

1) Technology Integration and Collaboration

Web services are able to bring together the components of UWB RTLS with other discrete systems by utilizing a variety of technologies in one system and ensuring that they function together as a system. Web services also provide a good organization of UWB RTLS to enhance its capabilities. Although some significant layers in the Web services stack have not been fully standardized, the current state of support for Web services provides an excellent application integration and collaboration technology (Ferris and Farrell, 2003).

2) Remote Interaction

Web services can support remote interaction by providing APIs for UWB RTLS to execute it on a remote server system hosting the requested services and access it via

HTTP. The W3C defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network. As described in Subsection 2.3.1, a Web service has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards (Haas and Brown, 2002).

3) Facilitating Collaborative Scenarios

One of the limitations of UWB RTLS is a lack of services for environments which facilitate collaborative scenarios, such as actual work support and Web applications combining these services. Fortunately, Web services have the potential to serve as a key enabling technology for collaborative scenarios in UWB RTLS applications. Although the Web services framework is not a new information technology, it is able to extend the scope of UWB RTLS applications and support them (e.g. the three typical UWB RTLS applications described in Section 3.3) as a front-end. Web services can also provide a runtime environment for applications with various infrastructural services (Shepard, 2005).

4) Low Cost

Web services make UWB RTLS applications more economical, because companies or individuals are able to connect to UWB RTLS by using the services from Web services rather than developing new applications from scratch. Web services can also decrease the complexity of an application connection, which could reduce the cost of maintenance and updating (Huang and Tang, 2008).

3.5 Web Services Requirements for UWB RTLS

The ***UWB RTLS Web Service*** is designed to satisfy the common requirements of UWB RTLS applications which are defined from the specific requirements elicitation mentioned in Session 3.3, as well as other potential extended UWB RTLS applications. It supports other application developments used in UWB RTLS applications by providing its services. The requirements of the ***UWB RTLS Web Service*** are categorized into four parts: (1) requirements of technology integration and collaboration, (2) requirements of remote interaction, (3) requirements of awareness visualization, and (4) requirements of event detection.

3.5.1 Technology Integration and Collaboration Requirements

Users need to visualize real-time location data in building models and save them into files and databases. However, these functions are not supported by the UWB RTLS. Web services are able to integrate and collaborate with various applications in a loosely coupled way to achieve the goals of this research (Guan et al., 2005). Hence, the objective of technology integration is to integrate UWB RTLS with other technologies and applications (e.g. WMS applications, databases, and time-management Web applications) with a simple and friendly interface, instead of the specific and complicated software of UWB RTLS. In order to develop a Web service satisfying the common requirements of UWB RTLS applications, there are two requirements of technology integration and collaboration: (1) integration of UWB RTLS and WMS application and (2) integration of UWB RTLS and data storage.

1) Integration of UWB RTLS and WMS Applications

The integration of UWB RTLS and WMS applications implements awareness visualization, which will be further described in Subsection 3.5.3. Using the services from WMS applications can visualize real-time location data in a 4D digital display.

2) Integration of UWB RTLS and Data storage

Integration of UWB RTLS and data storage implements an organized collection of real-time location data and event messages in a digital form. Real-time location data can be stored in files and databases, and event messages can be stored in time-management Web applications. Analyzing a mass of saved real-time location data in data storage helps to improve process control by using other provided tools.

3.5.2 Remote Interaction Requirements

Users need to instantly and remotely monitor objects, manage UWB RTLS, and get real-time location data through the Internet in a way that is convenient and accessible anytime anywhere. Web services can be executed on a remote server system hosting the requested services and accessed via HTTP. The objective of remote access is not only to perform the instant remote operation on most of the functions for the management of UWB RTLS, but also to share real-time location data on the Internet. In order to develop a Web service to satisfy the common requirements of UWB RTLS applications, there are three requirements of remote interaction: (1) instant remote monitoring; (2) monitoring result sharing; and (3) remote UWB RTLS control.

1) Instant Remote Monitoring

Instant remote monitoring allows users to monitor tags or objects from on-line computers though the Internet anywhere in the world. It is similar to camera monitoring seen in offices or on streets, but it takes advantages of providing a 4D digital display and it is easier to for computers to process.

2) Monitoring Result Sharing

Monitoring result sharing allow users to obtain real-time location data generated by UWB RTLS and event messages generated by applications though the Internet. Without the need to install any software, users only need a Web browser to monitor and download real-time location data and event messages.

3) Remote UWB RTLS Control

Remote UWB RTLS control offers to operate UWB RTLS though the Internet, which has a similar interface, functions, and work-flows as the software of UWB RTLS.

3.5.3 Awareness Visualization Requirements

Awareness visualization is the visual tool which enables users to gain greater awareness on locations of tags or objects in their monitored environment by increasing each person’s ability to figure out tags or objects and their tracks in the 4D digital display for monitoring tasks.

Real-time location data can amply describe conditions of tag movements, but it is not suitable for general application developers. Real-time location data are too complicated to be understood from document files or database tables. Consequently, awareness visualization is considered as the best approach for helping users to understand real-time location data. In order to develop a Web service to satisfy the common requirements of UWB RTLS applications, there are two requirements of awareness visualization: (1) indoor monitoring and (2) indoor navigation.

1) Indoor Monitoring

Indoor monitoring is used for building safety and security. Instead of images from camera monitoring, the real-time location data from UWB RTLS are more conveniently understood by computer calculations. In addition, awareness visualization can be generated by visualizing real-time location data with dynamic features in a 4D digital display which is easier to understand.

2) Indoor Navigation

Indoor navigation needs the vector-based maps of buildings for route calculation and directions. This feature is similar to GPS navigation. Indoor navigation can show room numbers or names on every floor in a map, route information, and step-by-step routing direction, often in spoken form with a feature called “text to speech”.

3.5.4 Event Detection Requirements

Users need to log events, get an alarm when certain events occur, and seek out useful information such as which kind of events happen frequently. Therefore, the objective of event detection is to save event records with their attributes in database storages, generate alarms to respond to certain events, and apply data mining on event records to transform real-time location data into an informational advantage. In order to develop a Web service to satisfy the common requirements of UWB RTLS applications, there are two requirements of event detection: (1) event response, and (2) pattern prediction.

1) Event Response

Event response is the ability to respond after specific events are caught based on event attributes. For example, an equipment will be turned on when a tag button is pressed; an event will be recorded when an employee leaves a meeting room; an alarm will activate when two crane booms get close.

2) Pattern Prediction

Pattern prediction is a relevant technology of data-mining which extracts patterns from real-time location data. It predicts the routines of specific objects based on data-mining of events. For example, common movement patterns of customers may help form a new arrangement of items in a store; knowledge of common patterns and habits of employees in offices may help to improve productivity and safety.

3.6 Use Case Analysis

The use case is a description of the interactions and responsibilities of a system under discussion or design with external actors who may be a person, a group of people, or a computer system. It is associated with the goal of one particular actor, who is called the primary actor for that use case. The use case describes not only the various sets of interactions which can occur between the various external actors when the primary actor is in pursuit of that goal, but also the responsibilities of the system without getting into implementation techniques or system components. It collects together all scenarios which are possible sequences of interactions related to the goal of that primary actor (Cockburn, 2001).

In order to capture the behavioral requirements of the Web service from an actor's point of view, by detailing scenario-driven threads through the functional requirements, six fundamental use cases of the ***UWB RTLS Web Service*** are defined and analyzed: (1) get real-time location data; (2) get visualization data; (3) get event messages; (4) save real-time location data; (5) save event messages; and (6) collect monitoring result. The use case analysis comprises of not only UWB RTLS and other relative components, but also the externally visible properties and the relationships among them. As described in Section 3.5.1, the ***UWB RTLS Web Service*** is designed to emerge as a standards-based platform for technology integration and collaboration between UWB RTLS, WMS application, database, and time-management Web application though either intranets or the Internet and different companies. These fundamental use cases and their relationships can be demonstrated by the following use case diagram for the ***UWB RTLS Web Service*** illustrated in Figure 3-5 and the sequence diagram for the ***UWB RTLS Web Service*** is illustrated in Figure 3-6.

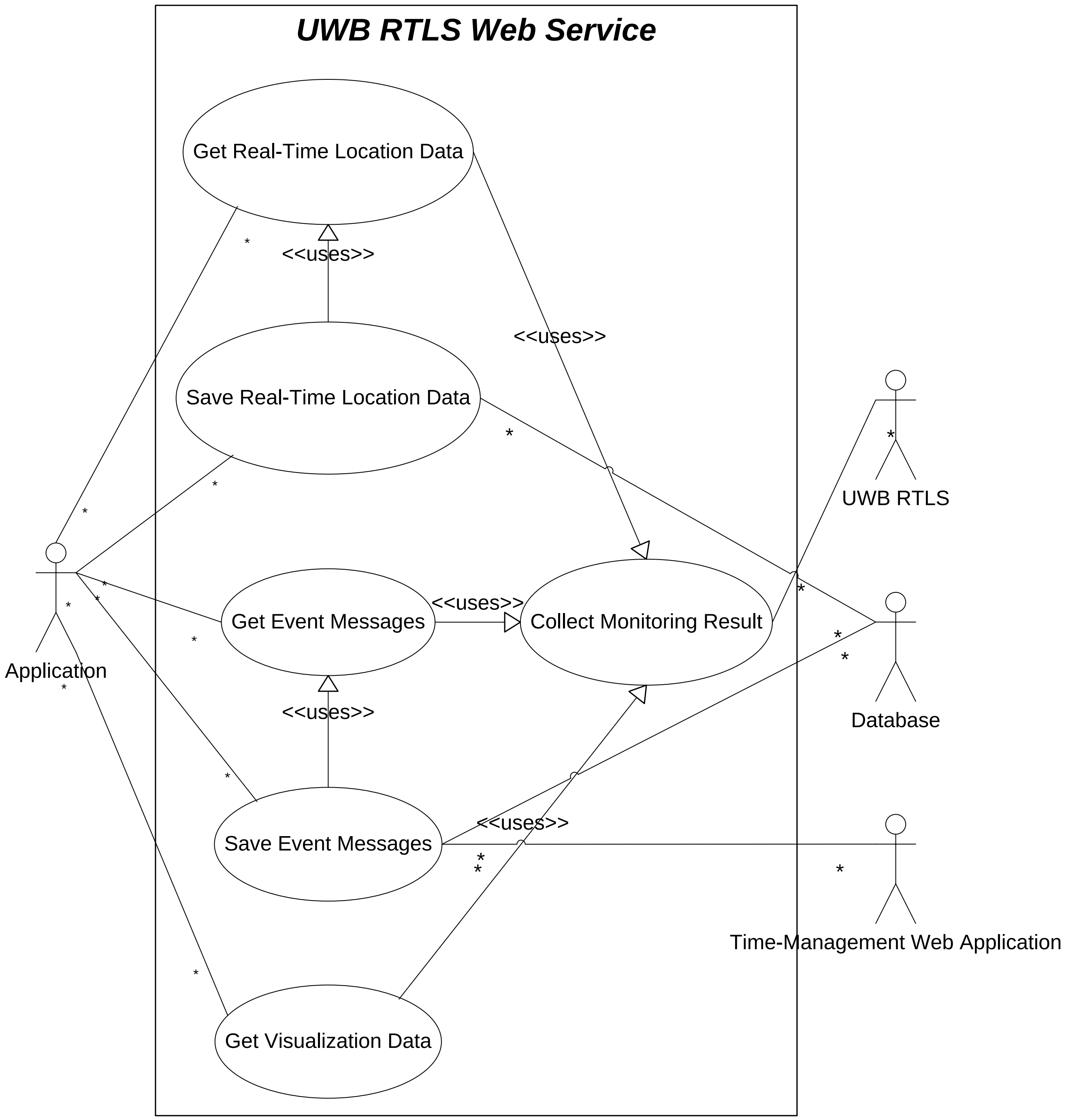


Figure 3-5 Use Case Diagram for UWB RTLS Web Service

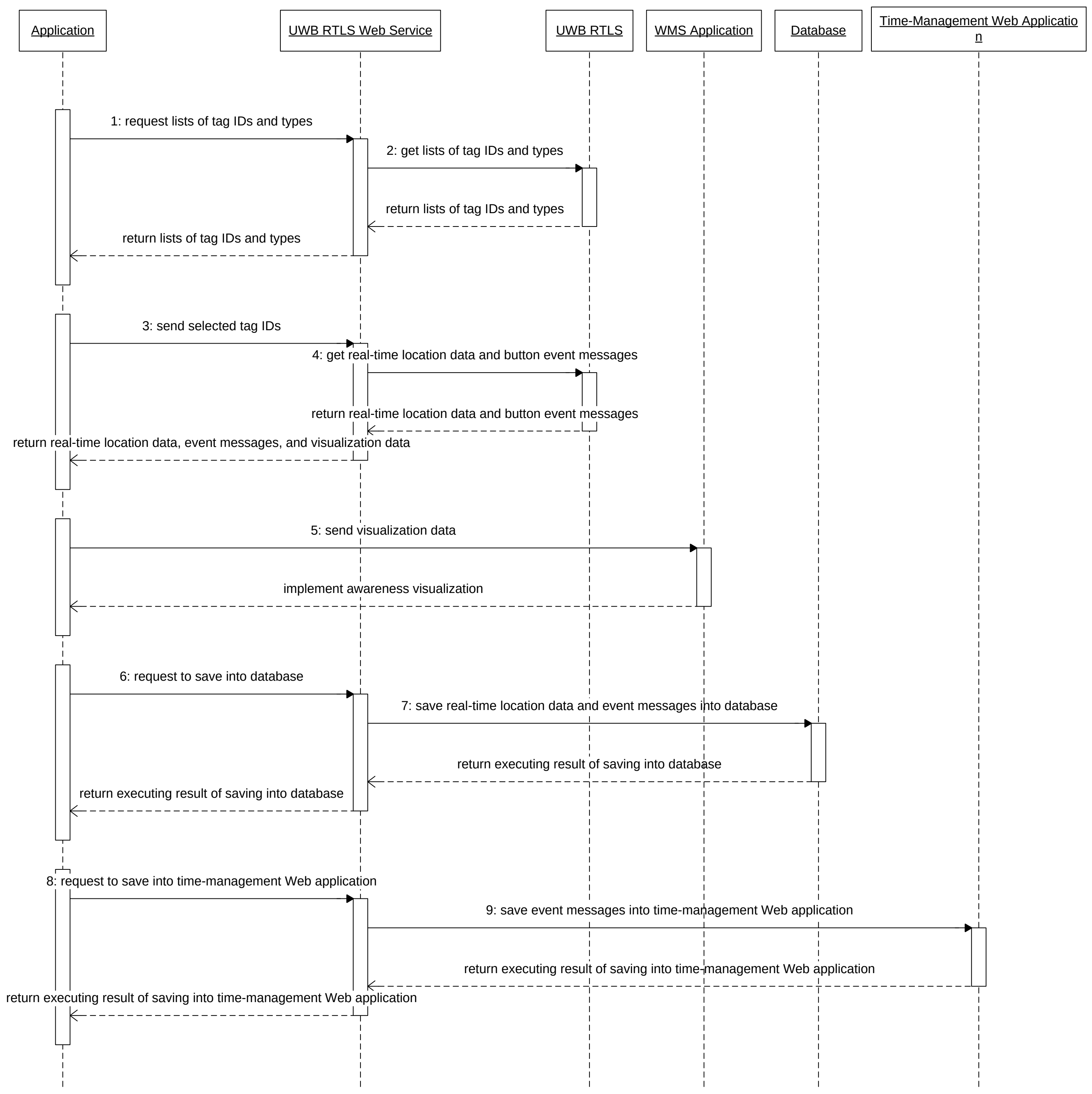

Application
UWB RTLS Web Service
UWB RTLS
WMS Application
Database
Time-Management Web Application
1: request lists of tag IDs and types
2: get lists of tag IDs and types
return lists of tag IDs and types
return lists of tag IDs and types
3: send selected tag IDs
4: get real-time location data and button event messages
return real-time location data and button event messages
return real-time location data, event messages, and visualization data
5: send visualization data
implement awareness visualization
6: request to save into database
7: save real-time location data and event messages into database
return executing result of saving into database
return executing result of saving into database
8: request to save into time-management Web application
9: save event messages into time-management Web application
return executing result of saving into time-management Web application
return executing result of saving into time-management Web application

Figure 3-6 Sequence Diagram for UWB RTLS Web Service

3.6.1 Use Case: Get Real-Time Location Data

Brief Description:

This use case is used by applications to get real-time location data.

Actor:

Application which is designed to record real-time location data.

Precondition:

1. UWB RTLS has been executed in the server.
2. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The application sends the IDs of the tags which are going to be monitored.
2. The application requests to start monitoring.
3. The ***UWB RTLS Web Service*** collects real-time location data from the UWB RTLS.
4. The application requests to get the latest real-time location data.
5. The ***UWB RTLS Web Service*** sends the latest real-time location data to the application.
6. The application receives the latest real-time location data and other related.

Alternative:

- The application cannot get real-time location data of a tag when it is not moving.

3.6.2 Use Case: Get Visualization Data

Brief Description:

This use case is used by applications to get visualization data in a specific file format of geographic data used in a WMS application (e.g. *kml* file used in Google Earth).

Actor:

Application which is designed to implement awareness visualization.

Precondition:

1. The plug-in of a WMS application has been installed.
2. UWB RTLS has been executed in the server.
3. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The ***UWB RTLS Web Service*** collects real-time location data from the UWB RTLS.
2. The application requests to get visualization data.
3. The ***UWB RTLS Web Service*** converts the real-time location data into visualization data and sends them to the application.
4. The application receives the visualization data.

Alternative:

- The application uses the WMS application by invoking the visualization data as a parameter and its JavaScript functions to implement awareness visualization.

- The application saves the visualization data into a file which is opened by the WMS application to implement awareness visualization (e.g. *kml* file is opened by Google Earth).
- The application cannot get visualization data used in a WMS application when there are no real-time location data.

3.6.3 Use Case: Get Event Messages

Brief Description:

This use case is used by applications to get event messages which include button event messages and movement event messages.

Actor:

Application which is designed to record event messages.

Precondition:

1. UWB RTLS has been executed in the server.
2. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The application sends the IDs of the tags which are going to be monitored.
2. The application requests to start monitoring.
3. The ***UWB RTLS Web Service*** collects real-time location data and button event messages from the UWB RTLS.
4. The application requests to get event messages.

5. The ***UWB RTLS Web Service*** analyzes real-time location data to detect movement events and generate movement event messages.
6. The ***UWB RTLS Web Service*** sends event messages (both button and movement event messages) to the application.
7. The application receives the event messages.

Alternative:

- The application cannot get movement event messages when there are no real-time location data.

3.6.4 Use Case: Save Real-Time Location Data

Brief Description:

This use case is used by applications to save real-time location data into files (e.g. *txt* files) in clients' computers and a database (e.g. MySQL) in a server.

Actor:

Application which is designed to record real-time location data.

Precondition:

1. UWB RTLS has been executed in the server.
2. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The ***UWB RTLS Web Service*** gets real-time location data.

2. The application requests to stop monitoring.
3. The ***UWB RTLS Web Service*** stops monitoring.
4. The application requests to save the real-time location data.
5. The ***UWB RTLS Web Service*** saves the real-time location data into files in clients' computers and a database in a server.

Alternative:

- The ***UWB RTLS Web Service*** cannot save the real-time location data into the database when it is not running.

3.6.5 Use Case: Save Event Messages

Brief Description:

This use case is used by applications to save event messages into a database (e.g. MySQL) in a server and a time-management Web application (e.g. Google Calendar). Event messages consist of button event messages and movement event messages.

Actor:

Application which is designed to record event messages.

Precondition:

1. UWB RTLS has been executed in the server.
2. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The ***UWB RTLS Web Service*** gets event messages.
2. The application requests to stop monitoring.
3. The ***UWB RTLS Web Service*** stops monitoring.
4. The application requests to save the event messages.
5. The ***UWB RTLS Web Service*** saves the event messages into a database in a server and a time-management Web application.

Alternative:

- The ***UWB RTLS Web Service*** cannot save the event messages into the database when it is not running.
- The ***UWB RTLS Web Service*** cannot save the event messages into the time-management Web application when its account's password is not correct.

3.6.6 Use Case: Collect Monitoring Result

Brief Description:

This use case is used in the ***UWB RTLS Web Service*** to collect monitoring result including real-time location data and button event message from UWB RTLS.

Actor:

The ***UWB RTLS Web Service***.

Precondition:

1. UWB RTLS has been executed in the server.
2. The ***UWB RTLS Web Service*** has been executed in the server.

Main Flow of Events:

1. The ***UWB RTLS Web Service*** determines the IDs of the tags which are going to be monitored.
2. The ***UWB RTLS Web Service*** starts monitoring.
3. The ***UWB RTLS Web Service*** collects real-time location data and button event messages from UWB RTLS.

Alternative:

- The application cannot get real-time location data of a tag when it is not moving and button event messages of a tag without pressing its buttons.

3.7 Summary and Conclusions

After defining common requirements of UWB RTLS applications by exploring the specific requirements of the three typical UWB RTLS applications, developing Web services for UWB RTLS is considered as the best approach to enhance its capabilities and satisfy these common requirements. In order to design the Web service for UWB RTLS in Chapter 4, the requirements of the ***UWB RTLS Web Service*** are determined based on the common requirements, and the five fundamental use cases of the ***UWB RTLS Web Service*** are analyzed.

CHAPTER 4 WEB SERVICE DESIGN FOR UWB RTLS

4.1 Introduction

The requirements of the ***UWB RTLS Web Service*** are determined based on the common requirements of UWB RTLS applications, and the five fundamental use cases of the ***UWB RTLS Web Service*** are analyzed in Chapter 3. The major purpose of this chapter is to design the ***UWB RTLS Web Service*** to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications. Firstly, this chapter discusses the four solutions which independently satisfy the requirements of the ***UWB RTLS Web Service***: (1) solution of technology integration and collaboration, (2) solution of remote interaction, (3) solution of awareness visualization, and (4) solution of event detection. Then, it presents design details of the ***UWB RTLS Web Service*** including the namespace decision and interface design.

4.2 Overall Web Service Design

The objective of the ***UWB RTLS Web Service*** design is to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications described in Section 3.3, as well as other potential extended UWB RTLS applications. As illustrated in Figure 4-1, the system is based on the Web services infrastructure as a common system platform which aligns with the goals and strategic vision of the UWB RTLS applications. However, a one-size-fits-all strategy is generally impossible, which means that different applications require their own strategies (Lahiri, 2005). As described in Section 3.5, the requirements of the ***UWB RTLS Web Service*** are categorized into technology integration, remote interactions, awareness visualization, and event detection. The solutions which address each of the four requirements are built based on the common system platform.

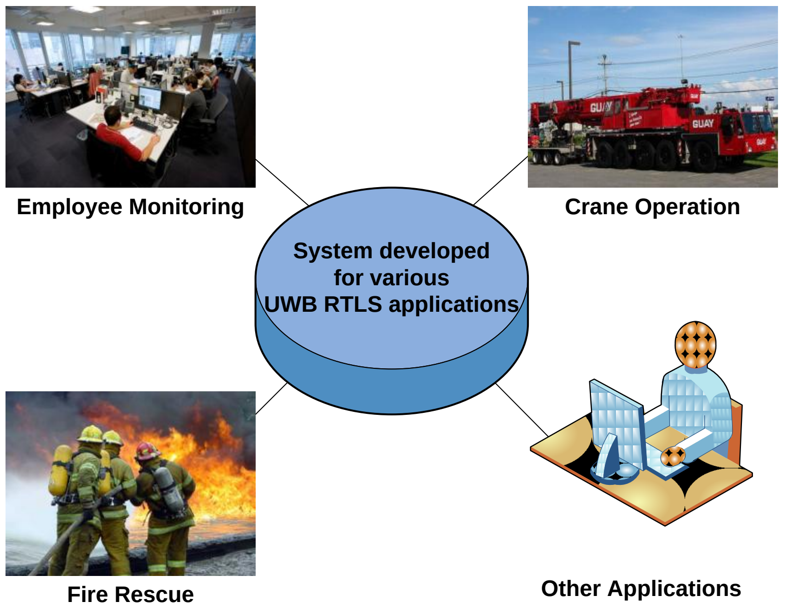


Figure 4-1 Common System Platform for Various UWB RTLS Applications

4.2.1 Technology Integration and Collaboration Solution

To satisfy the requirements of technology integration and collaboration, the ***UWB RTLS Web Service*** is designed to serve as middleware which integrates UWB RTLS, WMS applications, databases, and time-management Web applications together in a system and collaborates their services. The Web service collects real-time location data from UWB RTLS, generates visualization data in a specific file format of geographic data used in a WMS application, saves real-time location data and event messages into a database, and saves event messages into a time-management Web application. In this research, the

UWB RTLS is Ubisense system, the WMS application is Google Earth, the database is MySQL, and the time-management Web application is Google Calendar. The relationship of integrated technologies is illustrated in Figure 4-2.

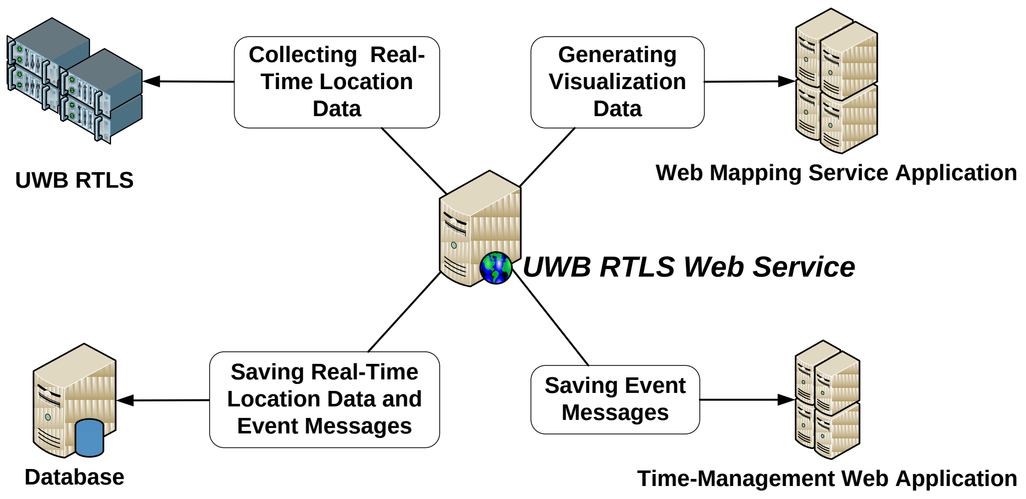

Collecting Real-Time Location Data
Generating Visualization Data
UWB RTLS
Web Mapping Service Application
UWB RTLS Web Service
Saving Real-Time Location Data and Event Messages
Saving Event Messages
Database
Time-Management Web Application

Figure 4-2 Relationship of Integrated Technologies

4.2.2 Remote Interaction Solution

To satisfy the requirements of remote interaction, the ***UWB RTLS Web Service*** is designed to provide services invoked by other applications. The Web service is on a remote server system hosting the requested services to customized applications developed by users. Those applications can access the Web service though the WWW and execute remote interaction using the services provided by the Web service to achieve instant remote monitoring, monitoring result sharing, and remote UWB RTLS control.

In terms of system implementation, accessible servers are deployed to execute remote interaction programs, such as UWB RTLS, WMS applications, databases, and time-management Web applications, which operate as socket listeners on the Internet or intranets. Users with thin client devices can interact with the servers using the services provided by the ***UWB RTLS Web Service***. A thin client can be any devices which can connect to the Internet (e.g. computers, laptops, cell phones, PDAs, and monitors).

4.2.3 Awareness Visualization Solution

To satisfy the requirements of awareness visualization, the ***UWB RTLS Web Service*** is designed to convert real-time location data into visualization data in a specific file format of geographic data used in a WMS application. Awareness visualization, required in this research, includes the feature of 4D digital display on general application platforms (e.g. Web browser) for UWB RTLS indoor monitoring. The design of awareness visualization explores variations on how to best to represent a range of features to provide efficiency in further monitoring and analysis. In this research, the file format for visualization data is Keyhole Markup Language (KML) which is used in several WMS applications, such as Google Earth and Bing.

Visualizing Real-Time Location Data

Visualizing real-time location data is an important requirement of positioning systems. The basic need is to visualize real-time location data to represent location and movement of tags, such as displaying groups of people in a small area or the movement of people throughout a monitored environment. The approach for visualizing real-time location data consists of three steps: (1) converting real-time location data into visualization data; (2) compiling the visualization data in a WMS application; and (3) generating dots representing locations of tags and lines representing traces of tags in the view of the WMS application. However, a mass of messy dots and lines crowding in a small screen makes it hard to distinguish tags, so dots and lines of different tags are printed in different colors. Users can easily follow the lines in the same color to search dots one by one. Instead of complicated interpretation of real-time location data, users are able to get valuable information about respective tags from their visual perception which can help to identify activities of tagged objects. These features are similar to those of vehicle navigation systems. For example, moving dots represent locations of objects, lines represent their traces, and other features represent conditions of objects movements, such as their speeds and directions.

Simulating Monitored Environments

Simulating monitored environments becomes one of the crucial components of awareness visualization in order to provide information of the internal structure of a building and a better orientation in complex buildings. Monitored environments can be constructed using 3D building models in WMS applications, BIM, or other technologies to represent their information, such as the construction details of a monitored building. A UWB RTLS can support both 2D and 3D digital display of tracked objects within monitored environments. The "cell" is a term used in UWB RTSL software. However, a UWB RTLS has limited visualization functions using simple architectural objects, which makes it difficult to model large and complex buildings and constructions.

As described in Section 2.3, there are two approaches to create visual models to simulate monitored environment in this research: (1) using 3D drafting software to create building models which can be viewed in a WMS application (e.g. *skp* files are built using Google SketchUp and browsed in Google Earth); and (2) using building design software which support the standard ~~of~~ BIM format (e.g. *ifc* (Industry Foundation Classes) files are built by and browsed in Autodesk Revit Architecture).

4.2.4 Event Detection Solution

To satisfy the requirements of event detection, the ***UWB RTLS Web Service*** is designed to detect events and generate event messages for monitoring purposes and further analysis. Event messages are either presented in Web pages or saved into databases and time-management Web applications. In this research, two kinds of events are defined: (1) button events and (2) movement events.

Button Events

Button events derive from presses on tag buttons, such as the Ubisense tags will be described in Subsection 5.2.1. The button event records when, where, and which button on which tag is pressed.

Movement Events

Movement events derive from analyzing real-time locations of tags attached to objects. For a single tag on an object, movement events are based on locations, appearance, and disappearance of tags in the monitored site. For multiple tags on an object, movement events are based on the relative positions and movements of the tags. The movement event records when, where, and which object is having certain conditions related to the positions of the tags attached to it. The three processes of movement event generation are summarized in the following: (1) movement event definition; (2) approach for attaching tags to objects; and (3) movement event message generation.

1) Movement Event Definition

Firstly, movement event definition considers the differences of positions and movements of tags to represent certain movement event based on different monitoring purposes. For example, a tag moving into or exiting from a specific area triggers the movement event of the tagged object coming or leaving the area; the distance between two tags on different objects becoming shorter or longer than a specific threshold triggers the movement event of the two tagged objects getting closer to or farther away from each other.

2) Approach for Attaching Tags to Objects

Secondly, there are two methods of attaching tags to objects according to the different needs of UWB RTLS applications: (1) single tag on one object; and (2) multiple tags on one object.

Single Tag on One Object

In general cases, an object tagged by only one tag is adequate for positioning indoor, such as a tag mounted in an employee ID card. According to the location, appearance, or disappearance of a tag in the monitored site, UWB RTLS can position the tagged object and detect its coming or leaving events. The application of employee monitoring is considered as a representative case using this approach as explained in Subsection 3.3.1.

Multiple Tags on One Object

In some cases, an object can be tagged by multiple tags attached to different parts of the object. Multiple tags on one object improves the visibility of an object, as described in Section 3.2, and helps to reasonably estimate the actions of the object based on the changes of the relative positions and movements of the tags. To infer the actions of an object, tags need to be attached to crucial parts, so that their relative positions and movements will change during certain actions. For example, when an object is tagged on two sides and the two tags are orbiting with respect to each other, the system will assume that the object is spinning; when the two tags get farther away from each other, it will be assume that the object is extending. Based on the visualization of real-time location data, 3D model designers can generate a vivid object model under real-time actions. This technique is similar to body motion tracking used in video game productions to capture the movements of basketball or soccer players. The applications of fire rescue and crane operation are considered as representative cases using this approach as described in Subsections 3.3.2 and 3.3.3.

3) Movement Event Message Generation

Thirdly, movement event message generation is based on movement event definition and the calculation of positions and movements of tags from collected real-time location data.

4.3 Namespace Decision

The Web service namespace is used for providing a uniquely named Web service. According to the URL of the server of our research group, the namespace of the ***UWB RTLS Web Service*** is *http://montreal.ciise.concordia.ca/Jian/UWBRTLSWebService/*.

4.4 Interface Design

The Web service interface design needs to cover all the important functionalities in the application domain. To reduce the complexity for application development, the number of provided operations should be adjusted carefully (Hong and Feuerlicht, 2008). The four main categories in a set of Web service interface design principles ─ interface orientation, interoperability, autonomy and modularity, and business suitability ─ must be taken into account (Legner and Vogel, 2007).

The interfaces of the ***UWB RTLS Web Service*** are designed to create and publish Web services which allow Web communities to create an open architecture for the Web services requirements described in Section 3.5. The fundamental communication and description for the Web service are designed according to the collaborative scenarios of the three typical UWB RTLS applications described in Section 3.3.

Communication: SOAP

As described in Subsection 2.4.2, SOAP offers basic communication for Web Services as a simple messaging protocol at the basic functionality level to make communication mechanisms platform-independent, standard, secure, and as lightweight as possible (Curbera et.al, 2002). There are fifteen SOAP messages in the ***UWB RTLS Web Service***: (1) return the list of tag IDs; (2) return the list of tag types; (3) send the selected tag IDs; (4) return the selected tag IDS; (5) return the verifying result of whether or not real-time location data have been updated ; (6) return completed real-time location data; (7) return button event messages; (8) send tags IDs; (9) return movement event messages; (10) send monitoring parameters; (11) return visualization data for monitoring phase; (12) send monitoring parameters; (13) return visualization data for parsing phase; (14) return the executing result of saving into database; and (15) return the executing result of saving into time-management Web application.

Return the List of Tag IDs

Applications can invoke the XML Web service method of string[] getIDsList() to return the list of tag IDs as a SOAP response message, illustrated in Figure 4-3. In the following figures about SOAP messages, only the part of SOAP:Body will be shown.

```xml
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <getIDsListResponse xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
   <getIDsListResult>
    C000
    C001
    ...
   </getIDsListResult>
  </getIDsListResponse>
 </soap:Body>
</soap:Envelope>
```

Figure 4-3 SOAP Response Message to Return the List of Tag IDs

Return the List of Tag Types

Applications can invoke the XML Web service method of string[] getTypesList() to return the list of tag types as a SOAP response message, illustrated in Figure 4-4.

```xml
<soap:Body>
 <getTypesListResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getTypesListResult>
   Person
   Furniture
   ...
  </getTypesListResult>
 </getTypesListResponse>
</soap:Body>
```

Figure 4-4 SOAP Response Message to Return the List of Tag Types

Send the Selected Tag IDs

Applications can invoke the XML Web service method of void setCheckedIDsList(string[] checkedIDsList) to send the selected tag IDs as a SOAP request message, illustrated in Figure 4-5.

```
<soap:Body>
 <setCheckedIDsList xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <checkedIDsList>
   C000
   C001
   ...
  </checkedIDsList>
 </setCheckedIDsList>
</soap:Body>
```

Figure 4-5 SOAP Request Message to Send the Selected Tag IDs

Return the Selected Tag IDs

Applications can invoke the XML Web service method of string[] getCheckedIDsList() to return the selected tag IDs as a SOAP response message, illustrated in Figure 4-6.

```
<soap:Body>
 <getCheckedIDsListResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getCheckedIDsListResult>
   C000
   C001
   ...
  </getCheckedIDsListResult>
 </getCheckedIDsListResponse>
</soap:Body>
```

Figure 4-6 SOAP Response Message to Return the Selected Tag IDs

Return the Verifying Result of Whether or Not Real-Time Location Data Have Been Updated

Applications can invoke the XML Web service method of bool areUpdated() to verify whether or not real-time location data have been updated and return the verifying result as a SOAP response message, illustrated in Figure 4-7.

```
<soap:Body>
 <areUpdatedResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <areUpdatedResult>
   true
  </areUpdatedResult>
 </areUpdatedResponse>
</soap:Body>
```

Figure 4-7 SOAP Response Message to Return the Verifying Result of Whether or Not Real-Time Location Data Have Been Updated

Return Completed Real-Time Location Data

Applications can invoke the XML Web service method of string[] getCompletedData() to return completed real-time location data as a SOAP response message, illustrated in Figure 4-8.

```
<soap:Body>
 <getCompletedDataResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getCompletedDataResult>
   C000 2010/01/01 00:00:00:000 2.00 1.00 0.00 False 0 0 1
   C001 2010/01/01 00:00:00:000 5.00 4.00 3.00 False 0 0 1
   …
  </getCompletedDataResult>
 </getCompletedDataResponse>
</soap:Body>
```

Figure 4-8 SOAP Response Message to Return Completed Real-Time Location Data

Return Button Event Messages

Applications can invoke the XML Web service method of string[] getButtonEvents() to return button event messages as a SOAP response message, illustrated in Figure 4-9.

```
<soap:Body>
 <getButtonEventsResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getButtonEventsResult>
   C000 (2010/01/01 00:00:00:000) button 0 was pressed
   C001 (2010/01/01 00:00:01:000) button 1 was pressed
   …
  </getButtonEventsResult>
 </getButtonEventsResponse>
</soap:Body>
```

Figure 4-9 SOAP Response Message to Return Button Event Messages

Send Tags IDs and Return Movement Event Messages

Applications can invoke XML Web service method of string[] getMovementEvents(string link) to send tags IDs as a SOAP request message, illustrated in Figure 4-10, and return movement event messages as a SOAP response message, illustrated in Figure 4-11.

```
<soap:Body>
 <getMovementEvents xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <link>C000 C001</link>
 </getMovementEvents>
</soap:Body>
```

Figure 4-10 SOAP Request Message to Send Tags IDs

```
<soap:Body>
 <getMovementEventsResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getMovementEventsResult>
   C000&C001 (2010/01/01 00:00:00:000) lie down
   C000&C001 (2010/01/01 00:00:01:000) stand up
   …
  </getMovementEventsResult>
 </getMovementEventsResponse>
</soap:Body>
```

Figure 4-11 SOAP Response Message to Return Movement Event Messages

Send Monitoring Parameters and Return Visualization Data for Monitoring Phase

Applications can invoke XML Web service method of string getMonitoringKML(string latitude, string longitude, string height, string angle, string link, string type) to send monitoring parameters as a SOAP request message, illustrated in Figure 4-12, and return recent visualization data as a SOAP response message, illustrated in Figure 4-13, for monitoring phase.

```
<soap:Body>
 <getMonitoringKML xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <latitude>2</latitude>
  <longitude>1</longitude>
  <height>0</height>
  <angle>45</angle>
  <link>C000 C001</link>
  <type>AllTracks</type>
 </getMonitoringKML>
</soap:Body>
```

Figure 4-12 SOAP Request Message to Send Monitoring Parameters for Monitoring Phase

```
<soap:Body>
 <getMonitoringKMLResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getMonitoringKMLResult>
   <?xml version="1.0" encoding="UTF-8"?>
   <kml xmlns="http://www.opengis.net/kml/2.2">
    <Document>
     <name>Trace of tags</name>
     <description>Show traces of tag in Google Earth by using the data</description>
     …
    </getMonitoringKMLResult>
 </getMonitoringKMLResponse>
</soap:Body>
```

Figure 4-13 SOAP Response Message to Return Visualization Data for Monitoring Phase

Send Monitoring Parameters and Return Visualization Data for Parsing Phase

Applications can invoke XML Web service method of string getParsingKML(string latitude, string longitude, string height, string angle) to send monitoring parameters as a SOAP request message, illustrated in Figure 4-14, and return completed visualization data as a SOAP response message, illustrated in Figure 4-15, for parsing phase.

```
<soap:Body>
 <getParsingKML xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <latitude>2</latitude>
  <longitude>1</longitude>
  <height>0</height>
  <angle>45</angle>
 </getParsingKML>
</soap:Body>
```

Figure 4-14 SOAP Request Message to Send Monitoring Parameters for Parsing Phase

```
<soap:Body>
 <getParsingKMLResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <getParsingKMLResult>
   <?xml version="1.0" encoding="UTF-8"?>
   <kml xmlns="http://www.opengis.net/kml/2.2">
    <Document>
     <name>Trace of tags</name>
     <description>Show traces of tag in Google Earth by using the data</description>
     …
    </getParsingKMLResult>
 </getParsingKMLResponse>
</soap:Body>
```

Figure 4-15 SOAP Response Message to Return Visualization Data for Parsing Phase

Return the Executing Result of Saving into Database

Applications can invoke XML Web service method of `bool saveIntoMySQL()` to save real-time location data and event messages into a database and return the executing result as a SOAP response message, illustrated in Figure 4-16.

```
<soap:Body>
 <saveIntoMySQLResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <saveIntoMySQLResult>
   true
  </saveIntoMySQLResult>
 </saveIntoMySQLResponse>
</soap:Body>
```

Figure 4-16 SOAP Response Message to Return the Executing Result of Saving Real-Time Location Data and Event Messages into a Database

Return the Executing Result of Saving into Time-Management Web Application

Applications can invoke XML Web service method of `bool saveIntoGoogleCalendar()` to save event message into a time-management Web application and return the executing result as a SOAP response message, illustrated in Figure 4-17.

```
<soap:Body>
 <saveIntoGoogleCalendarResponse
xmlns="http://montreal.ciise.concordia.ca/jian/UWBRTLSWebService/">
  <saveIntoGoogleCalendarResult>
   true
  </saveIntoGoogleCalendarResult>
 </saveIntoGoogleCalendarResponse>
</soap:Body>
```

Figure 4-17 SOAP Response Message to Return the Executing Result of Save Event Message into a Time-Management Web Application

Description: WSDL

As described in Subsection 2.4.2, WSDL describes Web services as collections of communication end points which can exchange certain messages. Messages, whose elements are defined as aggregations of parts, provide an abstract type data definition sent to and from the services (Curbera et.al, 2002). The fifteen messages which might appear during the interaction of the ***UWB RTLS Web Service*** are illustrated in Figure 4-18.

```
<message name="getIDsListResponse">
 <part name="getIDsListResult" type="xsd:string[]"/>
</message>

<message name="getTypesListResponse">
 <part name="getTypesListResult" type="xsd:string[]"/>
</message>

<message name="setCheckedIDsList">
 <part name="checkedIDsList" type="xsd:string[]"/>
</message>

<message name="getCheckedIDsListResponse">
 <part name="getCheckedIDsListResult" type="xsd:string[]"/>
</message>

<message name="areUpdatedResponse">
 <part name="areUpdatedResult" type="xsd:bool"/>
</message>

<message name="getCompletedDataResponse">
 <part name="getCompletedDataResult" type="xsd:string[]"/>
```

```
</message>

<message name="getButtonEventsResponse">
  <part name="getButtonEventsResult" type="xsd:string[]"/>
</message>

<message name="getMovementEvents">
  <part name="link" type="xsd:string"/>
</message>

<message name="getMovementEventsResponse">
  <part name="getMovementEventsResult" type="xsd:string[]"/>
</message>

<message name="getMonitoringKML">
  <part name="latitude" type="xsd:string"/>
  <part name="longitude" type="xsd:string"/>
  <part name="height" type="xsd:string"/>
  <part name="angle" type="xsd:string"/>
  <part name="link" type="xsd:string"/>
  <part name="type" type="xsd:string"/>
</message>

<message name="getMonitoringKMLResponse">
  <part name="getMonitoringKMLResult" type="xsd:string"/>
</message>

<message name="getParsingKML">
  <part name="latitude" type="xsd:string"/>
  <part name="longitude" type="xsd:string"/>
  <part name="height" type="xsd:string"/>
  <part name="angle" type="xsd:string"/>
</message>

<message name="getParsingKMLResponse">
  <part name="getParsingKMLResult" type="xsd:string"/>
</message>

<message name="saveIntoMySQLResponse">
  <part name="saveIntoMySQLResult" type="xsd:bool"/>
</message>

<message name="saveIntoGoogleCalendarResponse">
  <part name="saveIntoGoogleCalendarResult" type="xsd:bool"/>
</message>


<portType name="UWBRTLSWebServicePortType">
```

```
<operation name="getIDsList">
  <output message="getIDsListResponse"/>
</operation>

<operation name="getTypesList">
  <output message="getTypesListResponse"/>
</operation>

<operation name="setCheckedIDsList">
  <input message="setCheckedIDsList"/>
</operation>

<operation name="getCheckedIDsList">
  <output message="getCheckedIDsListResponse"/>
</operation>

<operation name="areUpdated">
  <output message="areUpdatedResponse"/>
</operation>

<operation name="getCompletedData">
  <output message="getCompletedDataResponse"/>
</operation>

<operation name="getButtonEvents">
  <output message="getButtonEventsResponse"/>
</operation>

<operation name="getMovementEvents">
  <input message="getMovementEvents"/>
  <output message="getMovementEventsResponse"/>
</operation>

<operation name="getMonitoringKML">
  <input message="getMonitoringKML"/>
  <output message="getMonitoringKMLResponse"/>
</operation>

<operation name="getParsingKML">
  <input message="getParsingKML"/>
  <output message="getParsingKMLResponse"/>
</operation>

<operation name="saveIntoMySQL">
  <output message="saveIntoMySQLResponse"/>
</operation>

<operation name="saveIntoGoogleCalendar">
```

```
   <output message="saveIntoGoogleCalendarResponse"/>
  </operation>
</portType>
```

Figure 4-18 WSDL Abstract Description

4.5 Summary and Conclusions

The ***UWB RTLS Web Service*** is designed to cover the solutions of technology integration and collaboration, remote interaction, awareness visualization, and event detection in order to satisfy the common requirements of UWB RTLS applications. The design details of the ***UWB RTLS Web Service***, such as its namespace decision and interface design, are also presented. In accordance with the design of the ***UWB RTLS Web Service***, its programming modules and other applications using its services are implemented in detail in Chapter 5.

CHAPTER 5 IMPLEMENTATION

5.1 Introduction

The ***UWB RTLS Web Service*** is designed at a system level in Chapter 4. The main purpose of this chapter is to present the integrated components, the programming platform, and the three programming modules whose names are: (1) ***UWB RTLS Web Service***, which is the Web service developed to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications; (2) ***Tag Positioning Web Application***, which is an example of applications using the services provided by the ***UWB RTLS Web Service***; and (3) ***BIM Visualization Application***, which is an example of plug-in applications of Autodesk Revit Architecture visualizing real-time location data files in BIM.

5.2 Integrated Components

5.2.1 Ubisense System

Applying the Ubisense system as the UWB RTLS in this research is the core approach for detecting and tracking tags, and generating real-time location data. The Ubisense system, founded in 2003 by Cambridge, is a manufacturer of RTLS solutions based on UWB technology.

The Ubisense system is the combination of hardware ─ RTLS sensor network, and software ─ Location Platform and Developer Software. The hardware and software can detect precise real-time location, bring visibility and control to previously intractable business processes, and deliver value on an industrial scale (Ubisense, 2009).

Hardware

The Ubisense hardware solution consists of the Ubisense series sensors, and two types of tags ─ slim tags and compact tags. It uses standard network infrastructure and protocols to network devices which take advantage of the latest advances in both wired and wireless technologies. Throughout the area to be monitored, these types of Ubisense tags are located by a Ubisense sensors network (Ubisense, 2009).

Ubisense Sensors

Ubisense sensors calculate the locations of tags based on UWB signals from the tags. Ubisense Series 7000 sensors are illustrated in Figure 5-1 and its specifications can be found in Appendix B. They use the UWB signals from 6 GHz to 8 GHz and a conventional bidirectional 2.4 GHz radio as a control and telemetry channel. A cell is constructed by several Ubisense sensors connected together into a single operating unit, which captures the location of tracked objects. Each sensor is assembled with a UWB radio receiver which receives and evaluates UWB signals from the tags. It is able to determine AOA of the UWB signals from the tags, and the TDOA information through the timing cables between sensors. The sensors are supplied power by a Power over Ethernet (PoE) switch and connected by timing cables to synchronize the UWB signals from a tag to different sensors by using a timing signal (distributed by cables or wirelessly) from each sensor to the timing source (Ubisense, 2009).

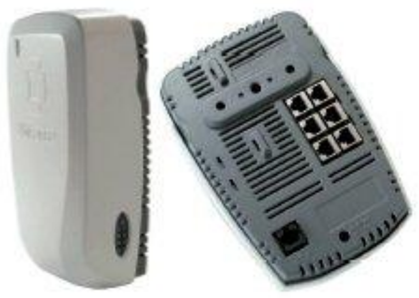


Figure 5-1 Ubisense Series 7000 Sensors (Ubisense, 2009)

One of sensors within a cell, called the master sensor, is defined to receive and synchronize the timing data from the other sensors. Other sensors, called slave sensors, decode the UWB signal and send AOA and TDOA information back to the master sensor through an Ethernet connection. Installation of Ubisense sensors is illustrated in Figure 5-2. The master sensor accumulates all sensed data and computes the location based on triangulation. The master sensor also represents the master function for the whole system, which is to collect and process the data of the other sensors and generate location events to the software — Location Engine Configuration over an existing IP network using the UDP protocol, called the Ubisense on the Wire (OTW) protocol (Wombacher, 2008). Commonly, four sensors are mounted at four corners in a roughly rectangle shape cell. The procedures of the installation and calibration of the cell network are defined in Appendix C. Once the sensors installation is completed, tags attached to objects can be positioned and tracked in the cell.

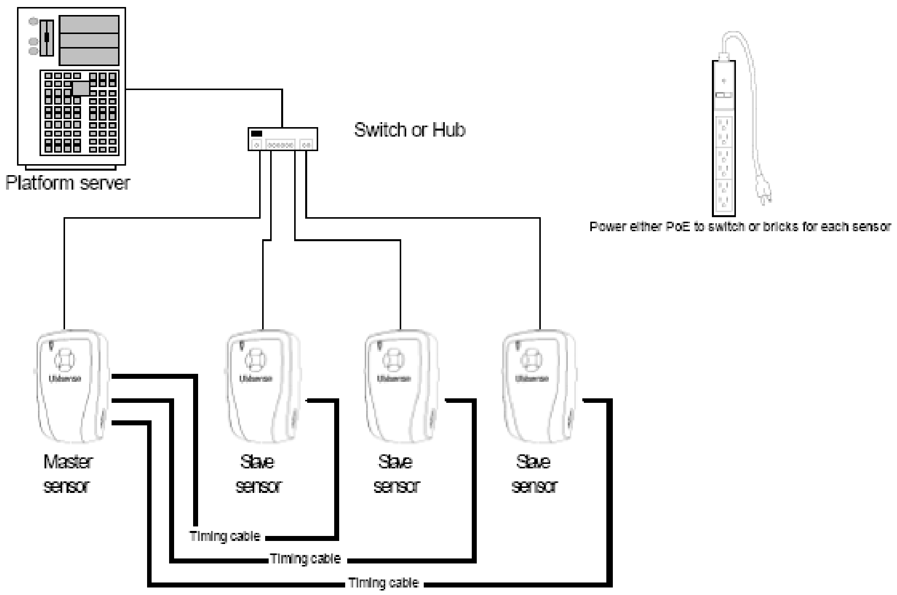
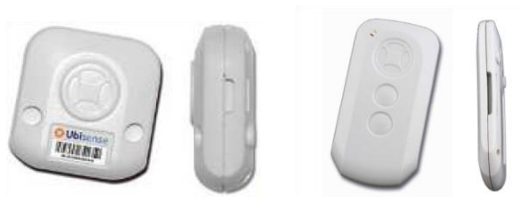


Figure 5-2 Installation of Ubisense Sensors (Ubisense, 2009)

Ubisense Tags

Ubisense active (battery powered) tags attach to objects and emit UWB pulses for positioning their location. The compact tag is especially suitable for harsh industrial environments. One of its advanced features is a Light Emitting Diode (LED) for easy identification. The slim tag is easily worn by people. One of its advanced features is with a motion detector to instantly activate a stationary tag and a push button to trigger events (Ubisense, 2009). The two types of Ubisense tags are illustrated in Figure 5-3 and their specifications can be found in Appendix D.



Compact Tag *Slim Tag*

Figure 5-3 Ubisense Tags (Ubisense, 2009)

Software

The Ubisense system offers a set of modular software products which cater to integrating the sensor system with 3rd party software platforms, integrating the software platform with 3rd party sensor systems, and providing a complete end-to-end location systems solution. The architecture of Ubisense software is illustrated in Figure 5-4. There are three main modules of software (Ubisense, 2009): (1) Location Engine Configuration, (2) Location Platform and Developer, and (3) Ubisense Simulator.

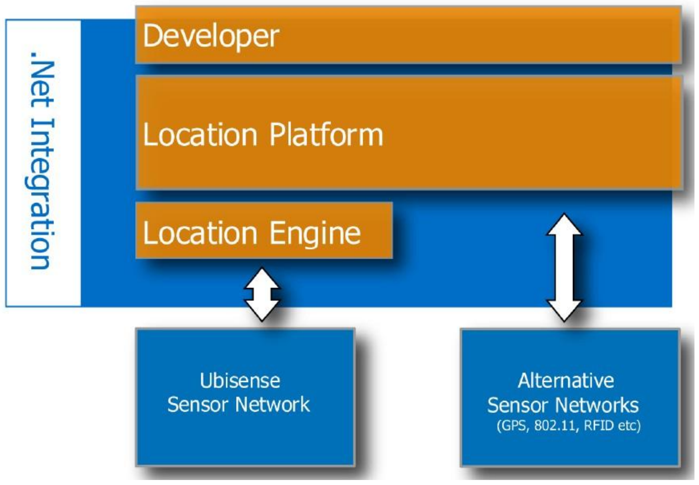


Figure 5-4 Architecture of Ubisense Software (Ubisense, 2009)

Location Engine Configuration

The Location Engine Configuration software runs with the Ubisense hardware and includes all the Ubisense software needed to install and tune a Ubisense sensor network and track tags precisely in real-time (Ubisense, 2009).

In Location Engine Configuration, Ubisense sensors are configured and logically installed together in a cell. The software provides different kinds of calibration and filters and displays the 3D view of a cell, illustrated in Figure 5-5. The Location Engine

Configuration software plays the leading role in installation and operation of the whole system.

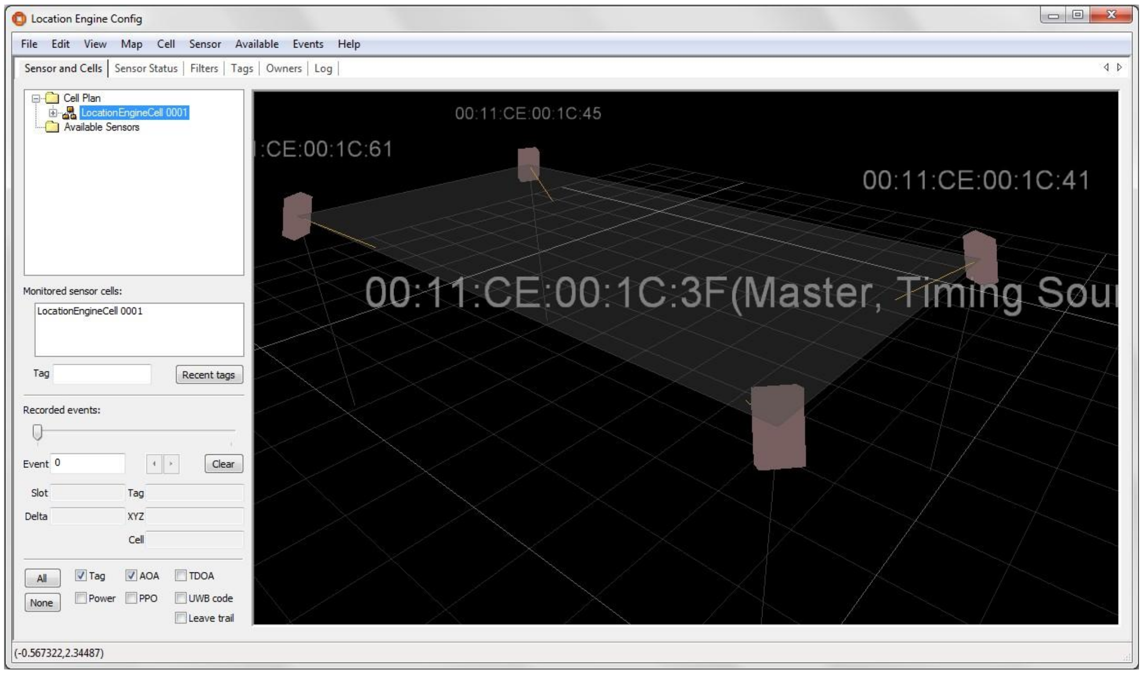


Figure 5-5 3D View of a Cell in Ubisense Location Engine Configuration

Location Platform and Developer

The products of Location Platform and Developer include all the software needed to build robust, scalable, extensible real-time location applications using the Ubisense system or 3rd party sensor systems (Ubisense, 2009). The architecture of Location Platform and Developer is illustrated in Figure 5-6.

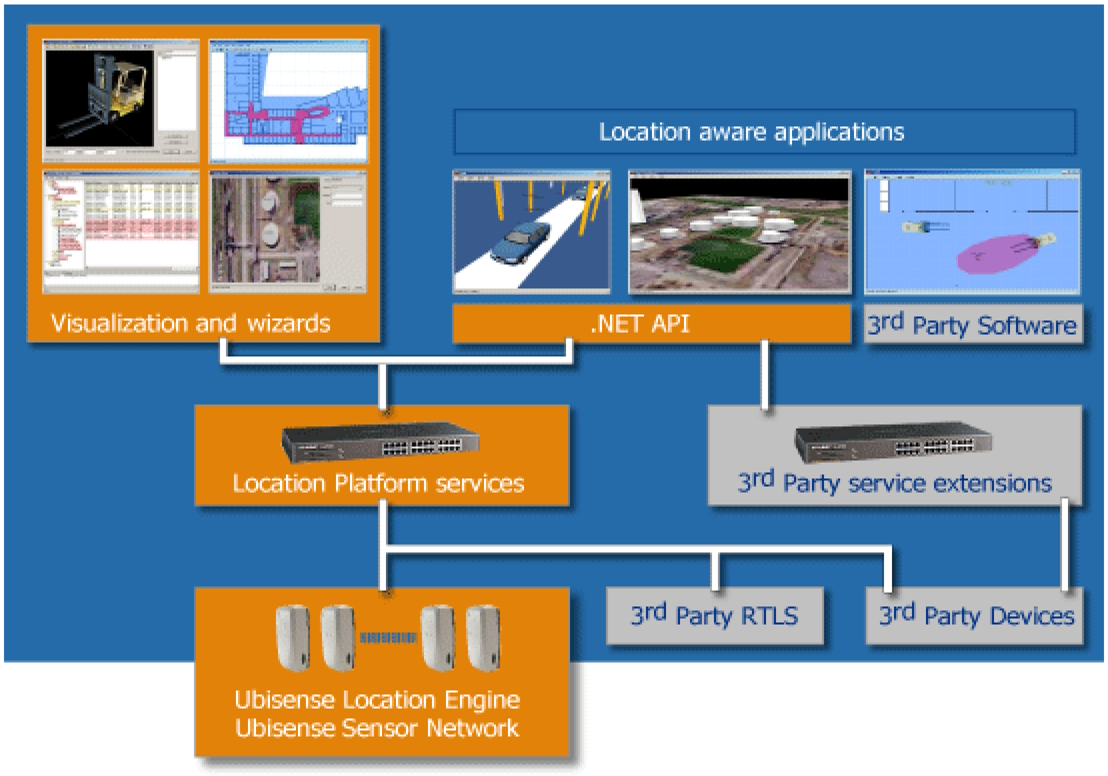

Visualization and wizards
Location aware applications
.NET API
3rd Party Software
Location Platform services
3rd Party service extensions
3rd Party RTLS
3rd Party Devices
Ubisense Location Engine
Ubisense Sensor Network

Figure 5-6 Architecture of Location Platform and Developer (Ubisense, 2009)

The core server, where Location Platform and Developer are installed, manages the deployment of services to the different controllers on the Ubisense sensors network and assigns duties to each other.

Ubisense Simulator

As illustrated in Figure 5-7, the Ubisense Simulator software is designed to add interactive simulation to the Ubisense system for testing and demonstrating location-aware applications without needing a sensor system. It is able to visualize application behavior, obtain quick feedback on location-aware applications, and create test scripts to automate the testing of applications.

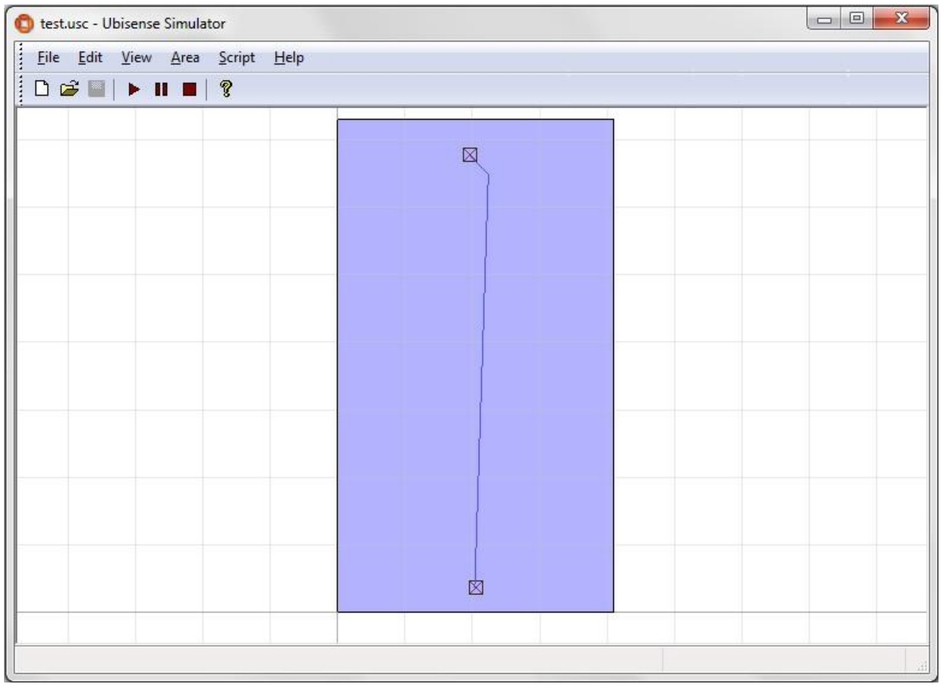


Figure 5-7 Ubisense Simulator

There are three main features of the Simulator: (1) Sensor Simulation, which represents the movements of the objects through the Ubisense Domain Object Model; (2) Motion Scripting, which scripts the movements of the objects; and (3) Motion Viewing, which views the movements of objects according to the scripting. The combination of these features and other components in the Ubisense Architecture is illustrated in Figure 5-8.

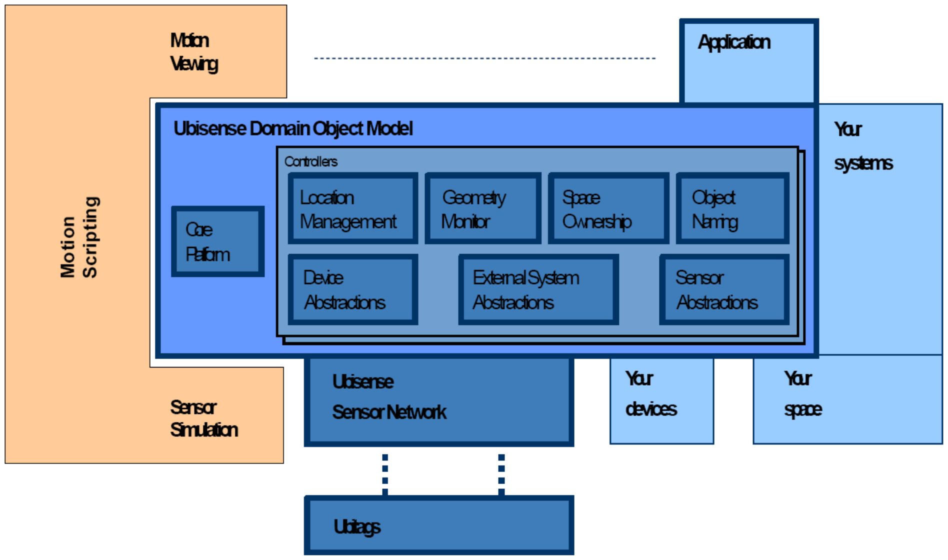


Figure 5-8 Ubisense Architecture (Ubisense, 2009)

Ubisense .NET API

Ubisense .NET API is the interface implemented by Ubisense which provides a simplified application programming interface (API) to key Ubisense components. By using this API, users can implement extended applications, such as query of current state and notification of state changes for location of objects, naming of objects, Ubisense tag notifications and button presses, and visualizing spatial events. In order to achieve this, it interfaces with several core components of the Domain Object Model (Ubisense, 2009).

5.2.2 Visualizing Real-Time Location Data in Google Earth

Applying Google Earth as the WMS application in this research is the first approach for simulating monitored environments for awareness visualization. Users can explore rich geographical content, save toured places, and share with others by flying anywhere on the earth through the view of satellite imagery, maps, terrain, and 3D buildings (Google Earth, 2010). Google Earth uses KML, which is an international standard maintained by the Open Geospatial Consortium (OGC), as a file format to display geographic data. Users can create their own *kml* files to pinpoint locations, add image overlays, and expose rich data in new ways (Google Code, 2010).

There are two important features of Google Earth for using it in this research: (1) users can view 3D building models in bird's eye view, so that it offers a conspicuous observation of buildings; and (2) developers can build their own 3D building models and post them into Google Earth to share with others. Consequently, in comparison with the second approach, will be described in Subsection 5.2.3, 3D buildings in Google Earth have a better capability for sharing information of buildings and their surrounding environments on maps.

Many visual building models of certain buildings and structures from around the world in Google Earth from around the world now have detailed 3D structures and are available via Google's 3D Warehouse and other Websites. Users can also generate their own visual building models in 3D modeling software and publish them into Google Earth.

The earth's surface is not perfectly round, but ellipsoid in shape. It means the semi-major axis is not perfect equal to the semi-minor axis. These geographic terms are illustrated in Figure 5-9. Google Earth uses WGS84 data: (1) semi-major axis = 6,378,137.0; and (2) semi-minor axis = 6,356,752.3142 (Google Earth, 2010). This geodetic data needs to be known for calculating precise locations and displaying them on the surface of Google Earth.

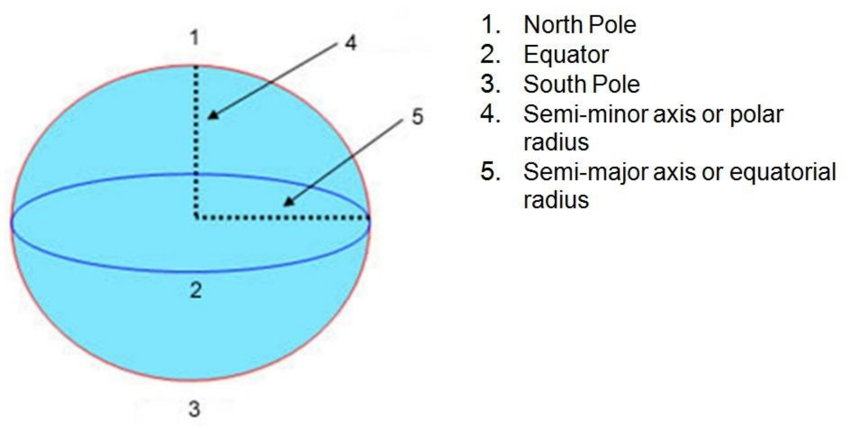

1
4
5
2
3
1. North Pole
2. Equator
3. South Pole
4. Semi-minor axis or polar radius
5. Semi-major axis or equatorial radius

Figure 5-9 Geographic Terms (Google Earth, 2010)

Google Earth Plug-in and its JavaScript API is the interface which embeds it into other Web pages. By using the Google Earth API, users can draw markers and lines, drape images over the terrain, add 3D models, load *kml* files, and consequently enable the development of building sophisticated 3D map applications (Google Code, 2010).

5.2.3 Visualizing Real-Time Location in Autodesk Revit Architecture

Applying Autodesk Revit Architecture as the BIM software in this research is the second approach for simulating monitored environments for awareness visualization on clients' computers. Autodesk Revit Architecture is a building design software purpose-built for BIM applied in architecture (Autodesk, 2010). Developers can develop plug-in applications of Autodesk Revit Architecture to implement additional functions.

In comparison with the first approach, as described in Subsection 2.3.3, BIM in Autodesk Revit Architecture takes advantage of offering more information and detail of complex and elaborate architectural objects.

Revit.NET API is the interface implemented by Autodesk Revit Architecture which enables it to interact with other programs and software. Focused on integrating applications of architectural analysis and visualization, Revit.NET API allows to program with any .NET compliant language including VB.NET, C#, and managed C++ (Autodesk, 2010).

5.3 Programming Platform

In this research, the technology of .NET framework is the programming platform to build the Web service and applications for UWB RTLS, C# is the programming language used in .NET framework, and Microsoft Visual Studio is the Integrated Development Environment (IDE) from Microsoft.

C# is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. The integrated components, described in Section 5.2, provide their own APIs in C# programming language. Therefore, programming in C# can have a significant advantage for integrating those components easily.

.NET framework is a prevalent platform from Microsoft. The technology of .NET has two primary advantages: (1) Active Server Pages (ASP) .NET, a Web application framework running on the Internet Information Services (IIS), can simplify application development and maintenance, and implement a great executing efficiency; and (2) Microsoft's Visual Studio and other third providers provide several developing and debugging tools for developers.

5.4 Programming Modules

This section presents the three programming modules developed in this research: (1) the ***UWB RTLS Web Service*** is the Web service developed to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications; (2) the ***Tag Positioning Web Application*** is an example of applications using the services provided by the ***UWB RTLS Web Service***; and (3) the ***BIM Visualization Application*** is an example of plug-in applications of Autodesk Revit Architecture visualizing real-time location data files in BIM. The major challenge related to the implementation of these three programming modules is the clear interpretation of the ASP.NET framework, Ajax framework, Ubisense Client API, Google Earth API, MySQL API, Google Calendar API, and Autodest Revit Architecture API.

5.4.1 Architecture of the Three Programming Modules

As described in Subsection 4.2.1, the WMS application used in this research is Google Earth, the database is MySQL, and the time-management Web application is Google Calendar. The ***UWB RTLS Web Service*** is deployed on a server to communicate with the Ubisense system on the intranet, and MySQL and Google Calendar on the Internet. The ***Tag Positioning Web Application*** is deployed on a server to use the services provided by the ***UWB RTLS Web Service*** via the intranet and the services from Google Earth via the Internet. The ***BIM Visualization Application*** is installed in Autodesk Revit Architecture in a machine. The architectural relationship of the three programming modules is illustrated in Figure 5-10 with interaction scenarios labeled.

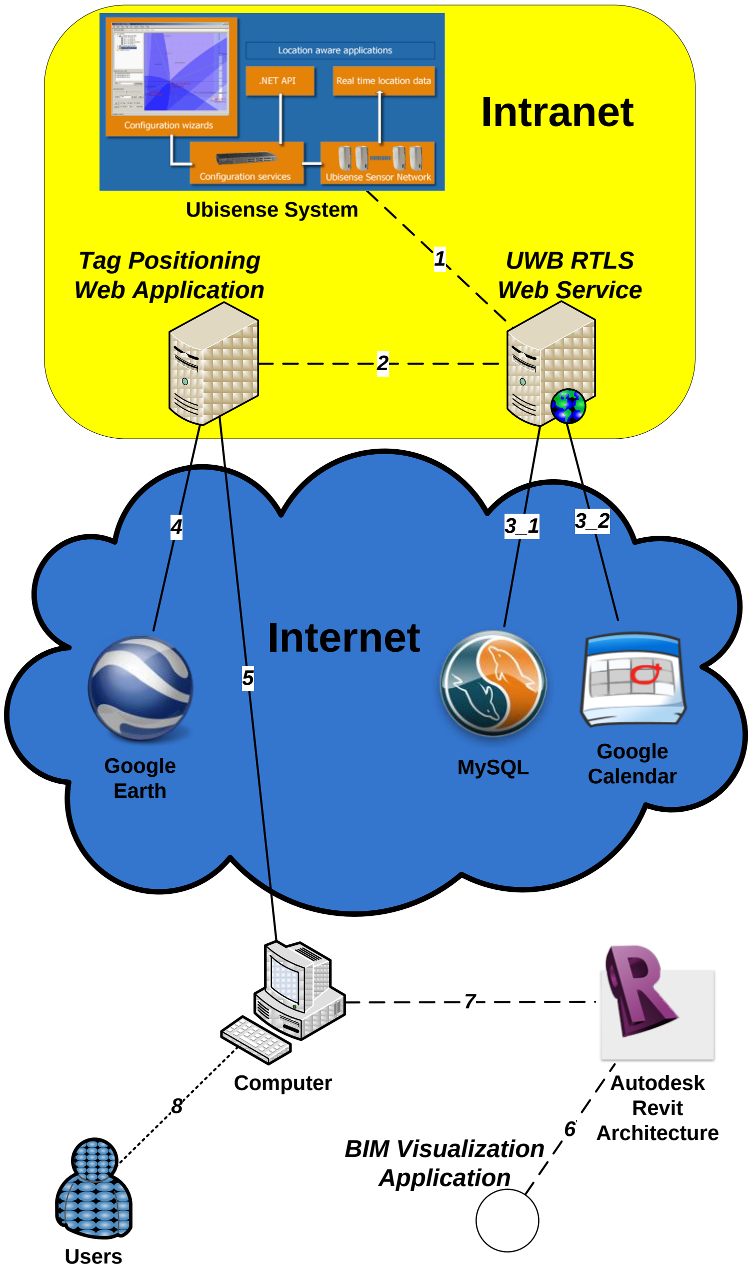


Figure 5-10 Architectural Relationship of Three Programming Modules

In Scenario 1, the ***UWB RTLS Web Service*** communicates with the Ubisense system to collect real-time location data and button event messages on the intranet.

In Scenario 2, the ***Tag Positioning Web Application*** uses the services provided by the ***UWB RTLS Web Service*** on the intranet.

In Scenario 3, the ***UWB RTLS Web Service*** saves real-time location data and event messages into data storage. In Scenario 3_1, real-time location data and event messages are saved into MySQL. In Scenario 3_2, event messages are saved into Google Calendar.

In Scenario 4, the ***Tag Positioning Web Application*** implements visualization of real-time location data by invoking the Google Earth APIs to visualize visualization data in KML obtained from the ***UWB RTLS Web Service***.

In Scenario 5, a computer accesses the ***Tag Positioning Web Application*** as a thin client.

In Scenario 6, the ***BIM Visualization Application*** is installed in Autodesk Revit Architecture as a plug-in application to visualize real-time location data files downloaded from the ***Tag Positioning Web Application*** in the BIM of a monitored site.

In Scenario 7, a computer runs Autodesk Revit Architecture to open the BIM of a monitored site.

In Scenario 8, users monitor indoor tags on the computer.

5.4.2 UWB RTLS Web Service

The primary objective of the programming module of the ***UWB RTLS Web Service*** is to implement a Web service developed to enhance the capabilities of UWB RTLS and satisfy the common requirements of UWB RTLS applications, which adopts a service-oriented approach. Instead of the complex structure and platform limitations of Ubisense API invocation, users can invoke simple services published by this Web service on a variety of platforms. In the following, the specific implementations of awareness visualization and event detection are introduced in this programming module.

Awareness Visualization

This Web service is able to implement awareness visualization. As described in Subsection 4.2.3, real-time location data are converted into visualization data in KML which can be used in Google Earth.

In practical tests of UWB RTLS applications, the monitored site is simulated as a cell in the Ubisense system. X, Y, and Z coordinates are used to determine the unique position of a point in the cell. In order to convert real-time location data into visualization data in KML, the latitude, longitude, and altitude of the point on Earth need to be calculated from its X, Y, and Z coordinates in the cell. There are four parameters which need to be measured during the preparation of a test: (1) the latitude, longitude, and altitude in Google Earth, which position the origin set by the Ubisense system (where X, Y, and Z coordinates equal zero in the cell); and (2) the angle between the x axis of a cell and the east direction, which is illustrated in Figure 5-11 as α.

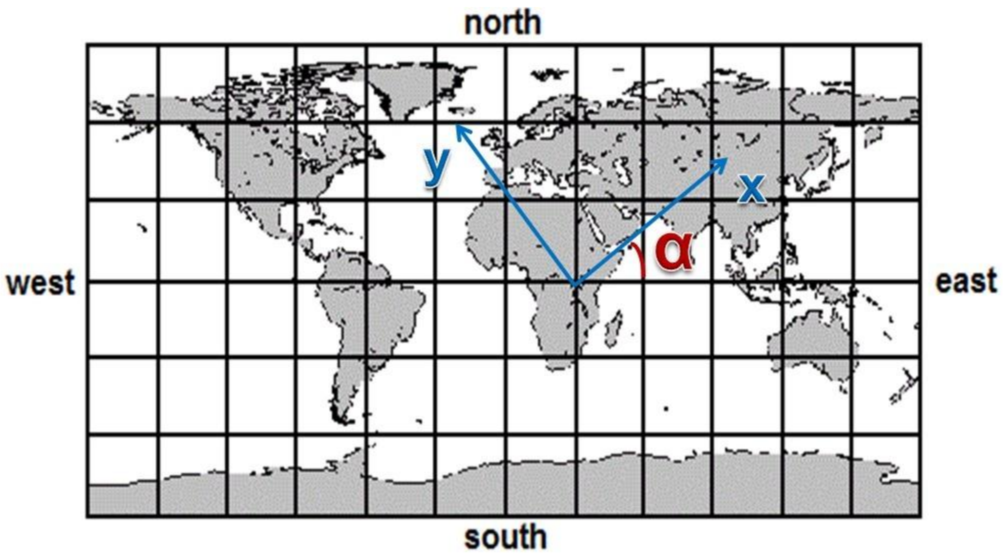


Figure 5-11 Angle between the X Axis of a Cell and the East Direction

Event detection

This Web service is able to detect the events on selected tags. As described in Subsection 4.2.4, there are two different methods for button events and movement events.

Button Events Detection Method

Button events can be directly detected by the Ubisense system. When a button is pressed on a selected tag, the Ubisense system detects the button event, and this Web service then generates a button event message to record it.

Movement Events Detection Method

Movement events are detected by analyzing real-time locations of the tags attached to objects. When selected tags perform a specific movement, this Web service generates a movement event message to record it by following the processes of movement event generation described in Subsection 4.2.4.

Programming Details

Service.cs

The class *Service.cs* is the code file behind the Web service page ─ *Service.asmx*.

GeographicCalculator.cs

The class *GeographicCalculator.cs* calculates geographic data (latitude, longitude, and height) used in Google Earth from X, Y, and Z coordinates generated by Ubisense system. As described in Subsection 5.2.3, it uses the WGS84 datum.

GoogleCalendarOperator.cs

The class *GoogleCalendarOperator.cs* saves event message into Google Calendar. The event includes tag ID, location, and time.

GoogleEarthKMLConvertor.cs

The class *GoogleEarthKMLConvertor.cs* converts real-time location data into visualization data in KML using in Google Earth. Visualization data in KML create dots representing tags and lines representing their tracks in the view of Google Earth. Visualization data in KML describe these main parameters of geographical information including name, color, description, time, and coordinates.

MovementEvents.cs

The class *MovementEvents.cs* is a movement event handler which can be modified based on different monitoring purposes and further analysis. The movement event handler catches movement events, described in Subsection 4.2.4, when tags are moving in specific situations. Movement event data refer to attributes of tag ID, date (year, month, and day), time (hour, minute, second, and millisecond), and movement event description. Following the processes of movement event generation described in Subsection 4.2.4, there are three valuable movement events which are defined and detected for the three typical UWB RTLS applications described in Section 3.3: (1) objects coming or leaving; (2) objects standing or lying; and (3) two objects getting closer to or farther away from each other.

MySQLOperator.cs

The class *MySQLOperator.cs* saves real-time location data into the table of Realtime_Location_Data, button events into the table of Button_Events, and movement events into the table of Movement_Events in MySQL.

UbisenseConnector.cs

The class *UbisenseConnector.cs* connects to Ubisense system and collect real-time location data. It sets up three schemas declarations: (1) naming_schema queries for names of tags or objects; (2) build_contents_schema provides controls for the building area; and (3) data_schema collects real-time location data generated by Ubisense system. There are two event handlers designed for the two kinds of events: (1) button event handler; and (2) real-time location event handler.

Button Event Handler: The button event handler catches button events, described in Subsection 4.2.4, when tag buttons are pressed. Button event data refer to the attributes of tag ID, date (year, month, and day), time (hour, minute, second, and millisecond), and button numbers.

Real-Time Location Event Handler: The real-time location event handler catches real-time location events when selected tags are moving in cells. Real-time location event data refer to attributes of tag ID, date (year, month, and day), time (hour, minute, second, and millisecond), X, Y, and Z coordinates, accuracy validity, accuracy GDOP, accuracy error, and recording count.

Limitations

(1) Limitation of storage volume restricts the amount of real-time location data which can be stored by this Web service at any given time. This restriction is based on how long or how many tags are monitored.

(2) Limitation of a high visitor volume restricts many applications from accessing the services of this Web service at the same time. This limitation is based on the probability of multi-thread crush of the server, because each application has its own HTTP sessions.

(3) Limitation on time range is one second, which is the smallest time range of Google Earth and *kml* files. Therefore, it is impossible to get a higher exactitude for this Web service. Fortunately, this limitation only affects simulated tests and seldom affects practical tests.

5.4.3 Tag Positioning Web Application

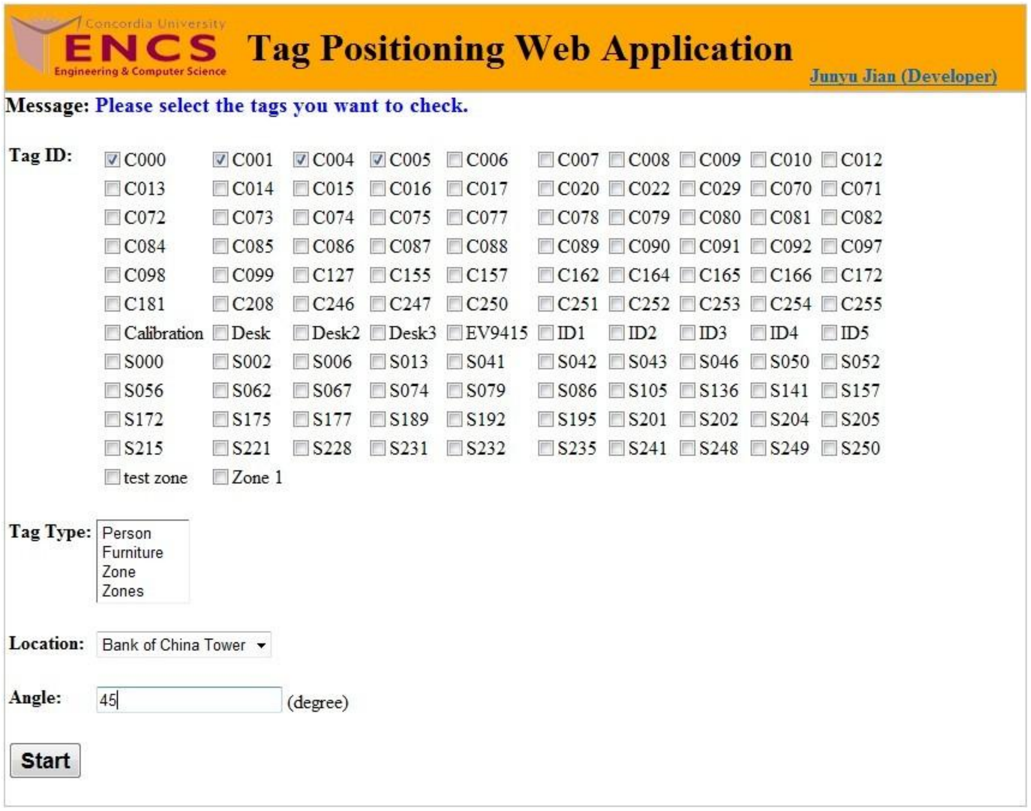
The primary objective of the programming module of the ***Tag Positioning Web Application*** is to implement a Web application using the services provided by the ***UWB RTLS Web Service***, which demonstrates the Web service benefits. Instead of having to install any software or hardware on users' machines, this Web application can be easily and conveniently accessed in a Web browser from remote computers for various UWB RTLS applications (e.g. the three typical UWB RTLS applications described in Section 3.3). For better quality, this Web application is designed with a friendly user interface and excellent awareness visualization. In addition, its environment is able to handle multiple threads (at least 5 simultaneous accesses) and offer moderate periods of monitoring time (at least 30 minutes for each data recording HTTP session) without obvious delay in terms of Web page display.

Steps for Using

There are three steps for using this Web application: (1) configuring, (2) monitoring, and (3) parsing.

1) Configuring

Before starting monitoring, this Web application needs to be configured. As illustrated in Figure 5-12, the user should (1) choose the tag IDs to monitor from the Tag ID check-box; (2) input the angle degree into the Angle text-box to set up the angle between the X axis of a cell and the east direction. The angle between the X axis of a cell and the east direction needs to be measured while preparing for a test; (3) choose the monitored location in the Location drop-list-box. For current tests, there are four locations of monitored sites: Guay Company, EV Building, JMSB Building, and Bank of China Tower, whose data are in *configuration.xml*. Each location includes its name, latitude, longitude, and height; and (4) click the Start button to start the monitoring phase.



Tag Positioning Web Application

Junyu Jian (Developer)

Message: Please select the tags you want to check.

Tag ID:

✓ C000	✓ C001	✓ C004	✓ C005	C006	C007	C008	C009	C010	C012
C013	C014	C015	C016	C017	C020	C022	C029	C070	C071
C072	C073	C074	C075	C077	C078	C079	C080	C081	C082
C084	C085	C086	C087	C088	C089	C090	C091	C092	C097
C098	C099	C127	C155	C157	C162	C164	C165	C166	C172
C181	C208	C246	C247	C250	C251	C252	C253	C254	C255
Calibration	Desk	Desk2	Desk3	EV9415	ID1	ID2	ID3	ID4	ID5
S000	S002	S006	S013	S041	S042	S043	S046	S050	S052
S056	S062	S067	S074	S079	S086	S105	S136	S141	S157
S172	S175	S177	S189	S192	S195	S201	S202	S204	S205
S215	S221	S228	S231	S232	S235	S241	S248	S249	S250
test zone	Zone 1								

Tag Type: Person / Furniture / Zone / Zones

Location: Bank of China Tower

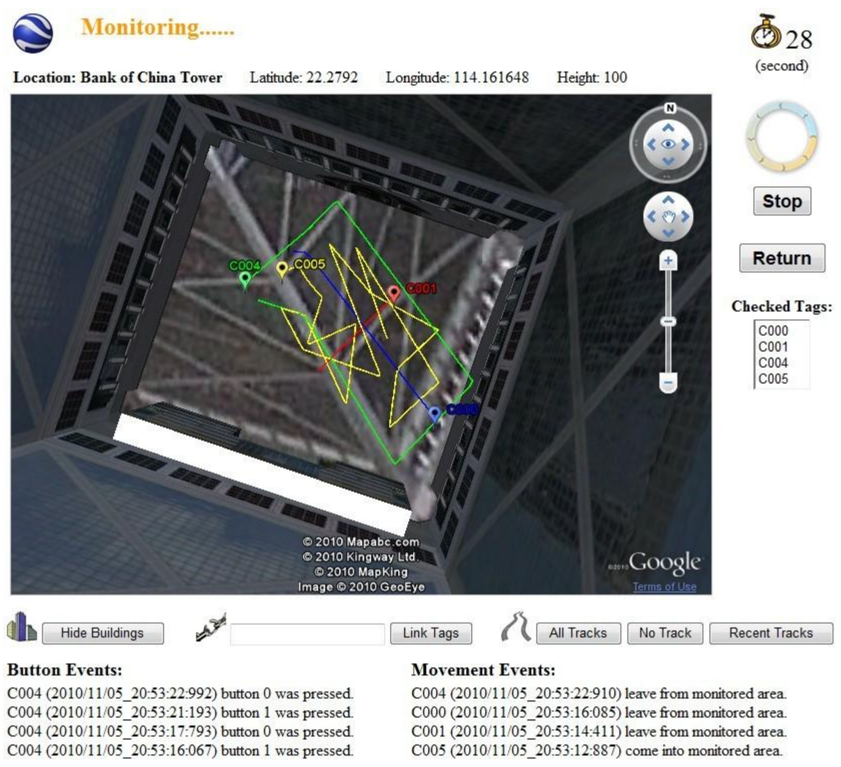
Angle: 45 (degree)

Figure 5-12 User Interface of Tag Positioning Web Application

2) Monitoring

After monitoring has started, this Web application will collect real-time location data and event messages from the ***UWB RTLS Web Service***. Real-time location data are dynamically updated and visualized in Google Earth. The user should click the Stop button to stop monitoring and continue to the parsing phase or click the Return button to return back and prepare for monitoring again.

In the view of Google Earth as illustrated in Figure 5-13, each tag gets one color (in dots and lines). Moving dots represent real-time locations of tags and lines linking dots represent the traces of tags.



Location: Bank of China Tower Latitude: 22.2792 Longitude: 114.161648 Height: 100

Checked Tags:

```
C000
C001
C004
C005
```

Button Events:

```
C004 (2010/11/05_20:53:22:992) button 0 was pressed.
C004 (2010/11/05_20:53:21:193) button 1 was pressed.
C004 (2010/11/05_20:53:17:793) button 0 was pressed.
C004 (2010/11/05_20:53:16:067) button 1 was pressed.
```

Movement Events:

```
C004 (2010/11/05_20:53:22:910) leave from monitored area.
C000 (2010/11/05_20:53:16:085) leave from monitored area.
C001 (2010/11/05_20:53:14:411) leave from monitored area.
C005 (2010/11/05_20:53:12:887) come into monitored area.
```

Figure 5-13 Monitoring in the View of Google Earth

Sometimes, traces of tags appear inside a building model. For the sake of easier observation, as illustrated in Figure 5-14, the view of Google Earth needs to be adjusted by hiding building models. The user should click the Hide Building or Show Building buttons to hide or show building models in the view of Google Earth.

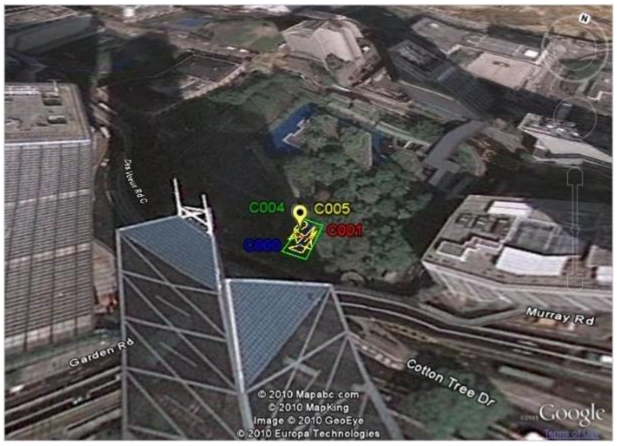


Figure 5-14 Monitoring in the View of Google Earth without Building Models

3) Parsing

After parsing has started, as illustrated in Figure 5-15, the user should click the Play button in the controlling bar of the view of Google Earth to replay recorded movements of tags.

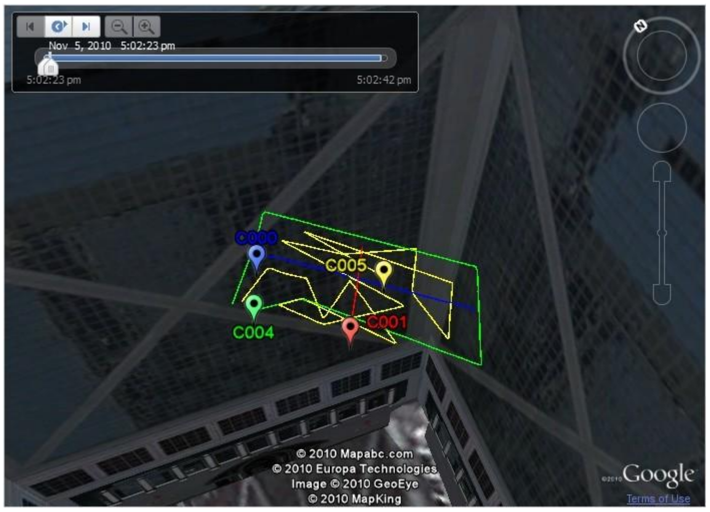
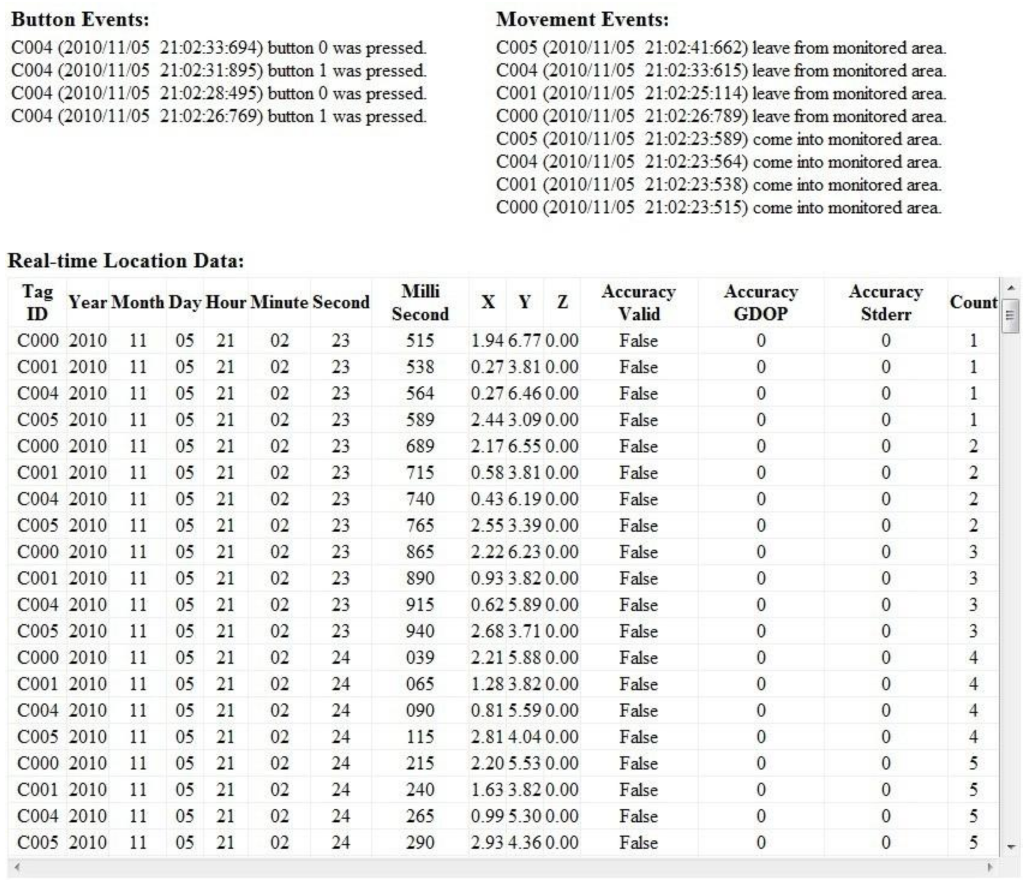


Figure 5-15 Parsing in the View of Google Earth

As illustrated in Figure 5-16, the real-time location data table enumerates the collected real-time location data from the monitoring. Every row in the table of real-time location data presents the observation on a tag in a position at a given moment and is sorted by time. The columns from left to right are attributes of tag ID, date (year, month, and day), time (hour, minute, second, and millisecond), X, Y, and Z coordinates, accuracy validity, accuracy (GDOP), accuracy error, and recording count.



Button Events:

```
C004 (2010/11/05  21:02:33:694) button 0 was pressed.
C004 (2010/11/05  21:02:31:895) button 1 was pressed.
C004 (2010/11/05  21:02:28:495) button 0 was pressed.
C004 (2010/11/05  21:02:26:769) button 1 was pressed.
```

Movement Events:

```
C005 (2010/11/05  21:02:41:662) leave from monitored area.
C004 (2010/11/05  21:02:33:615) leave from monitored area.
C001 (2010/11/05  21:02:25:114) leave from monitored area.
C000 (2010/11/05  21:02:26:789) leave from monitored area.
C005 (2010/11/05  21:02:23:589) come into monitored area.
C004 (2010/11/05  21:02:23:564) come into monitored area.
C001 (2010/11/05  21:02:23:538) come into monitored area.
C000 (2010/11/05  21:02:23:515) come into monitored area.
```

Real-time Location Data:

Tag ID	Year	Month	Day	Hour	Minute	Second	Milli Second	X	Y	Z	Accuracy Valid	Accuracy GDOP	Accuracy Stderr	Count
C000	2010	11	05	21	02	23	515	1.94	6.77	0.00	False	0	0	1
C001	2010	11	05	21	02	23	538	0.27	3.81	0.00	False	0	0	1
C004	2010	11	05	21	02	23	564	0.27	6.46	0.00	False	0	0	1
C005	2010	11	05	21	02	23	589	2.44	3.09	0.00	False	0	0	1
C000	2010	11	05	21	02	23	689	2.17	6.55	0.00	False	0	0	2
C001	2010	11	05	21	02	23	715	0.58	3.81	0.00	False	0	0	2
C004	2010	11	05	21	02	23	740	0.43	6.19	0.00	False	0	0	2
C005	2010	11	05	21	02	23	765	2.55	3.39	0.00	False	0	0	2
C000	2010	11	05	21	02	23	865	2.22	6.23	0.00	False	0	0	3
C001	2010	11	05	21	02	23	890	0.93	3.82	0.00	False	0	0	3
C004	2010	11	05	21	02	23	915	0.62	5.89	0.00	False	0	0	3
C005	2010	11	05	21	02	23	940	2.68	3.71	0.00	False	0	0	3
C000	2010	11	05	21	02	24	039	2.21	5.88	0.00	False	0	0	4
C001	2010	11	05	21	02	24	065	1.28	3.82	0.00	False	0	0	4
C004	2010	11	05	21	02	24	090	0.81	5.59	0.00	False	0	0	4
C005	2010	11	05	21	02	24	115	2.81	4.04	0.00	False	0	0	4
C000	2010	11	05	21	02	24	215	2.20	5.53	0.00	False	0	0	5
C001	2010	11	05	21	02	24	240	1.63	3.82	0.00	False	0	0	5
C004	2010	11	05	21	02	24	265	0.99	5.30	0.00	False	0	0	5
C005	2010	11	05	21	02	24	290	2.93	4.36	0.00	False	0	0	5

Figure 5-16 Tables of Event Messages and Real-Time Location Data

Real-time location data have been saved in an HTTP session and will be lost after the HTTP session is disposed. As illustrated in Figure 5-17, there are four different kinds of permanent storages for saving monitoring result: (1) *txt* file, (2) *kml* file, (3) MySQL, and (4) Google Calendar.

Figure 5-17 Four permanent Storages for Saving Monitoring Result

By clicking the icon of txt file and inputting a saving path in the saving window, the real-time location data will be saved into a *txt* file on a client computer. In the *txt* file, the real-time location data are arranged in a similar way as the Real-Time Location Data table. The *txt* file can be found in Appendix E.

By clicking the icon of kml file and inputting a saving path in the saving window, the real-time location data will be converted into visualization data in KML and saved into a *kml* file. In the *kml* file, the visualization data in KML are codified using XML as an encoding scheme. The *kml* file can be found in Appendix F. Users can open *kml* files in Google Earth to replay.

By clicking the icon of MySQL, real-time location data and event messages will be saved into MySQL. In MySQL, real-time location data are stored in a table, named Realtime_Location_Data with the same rows and column as the *txt* file, and button events and movement events are stored in two tables named Button_Events and Movement_Events respectively.

By clicking the icon of Google Calendar, event messages will be saved into Google Calendar. In Google Calendar, event messages are summarized in a calendar with basic detail, instead of enumerating every row.

Programming Details

UWBRTLSWebServiceConnector.cs

The class *UbisenseWebServiceConnector.cs* connects to the ***UWB RTLS Web Service*** and obtains real-time location data and other services by invoking the ***UWB RTLS Web Service*** API. Under the Web reference name — UWBRTLSWebService described in Subsection

5.3.2, the class named Service creates the object of the service of the ***UWB RTLS Web Service*** which has the Web methods.

Limitations

(1) Limitation of a high visitor volume restricts numerous clients from accessing this Web application at the same time. Because each client has his/her own HTTP session, this limitation is based on the probability of multi-thread crush of the server.

(2) Limitation of compatibility comes from this Web application being only supported by Internet Explorer. After some tests, it was found that real-time location data cannot be updated dynamically in Web pages opened by FireFox or Google Chrome. The reason is that only Internet Explorer fully supports Ajax.

5.4.4 BIM Visualization Application

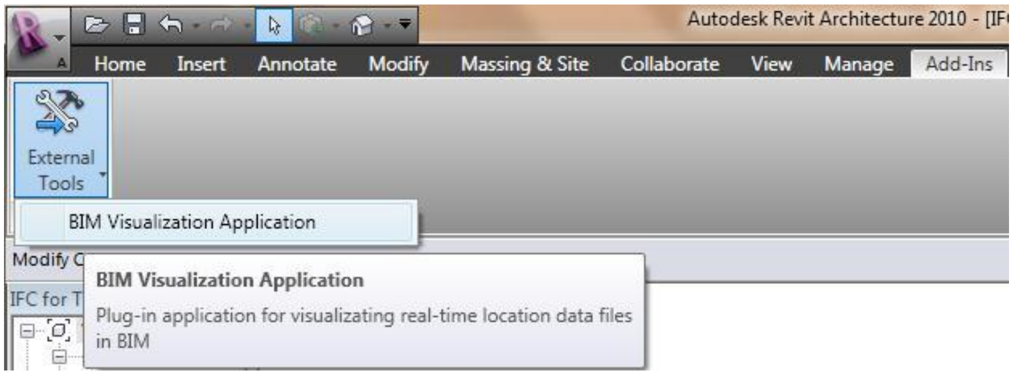
The primary objective of the programming module of the ***BIM Visualization Application*** is to implement a plug-in application of Autodesk Revit Architecture visualizing real-time location data files in BIM, which offers another approach for awareness visualization by using BIM instead of WMS applications.

Steps for Using

There are three steps for using this plug-in application: (1) executing in Autodesk Revit Architecture; (2) importing real-time location data; and (3) visualizing in BIM.

1) Executing in Autodesk Revit Architecture

Before executing this plug-in application in Autodesk Revit Architecture, the user should (1) click the tab of Add-Ins in the tool bar of Autodesk Revit Architecture, as illustrated in Figure 5-18; and (2) click the BIM Visualization Application tab which will appear in the External Tools menu to execute the plug-in application. The window of this plug-in application will pop out.

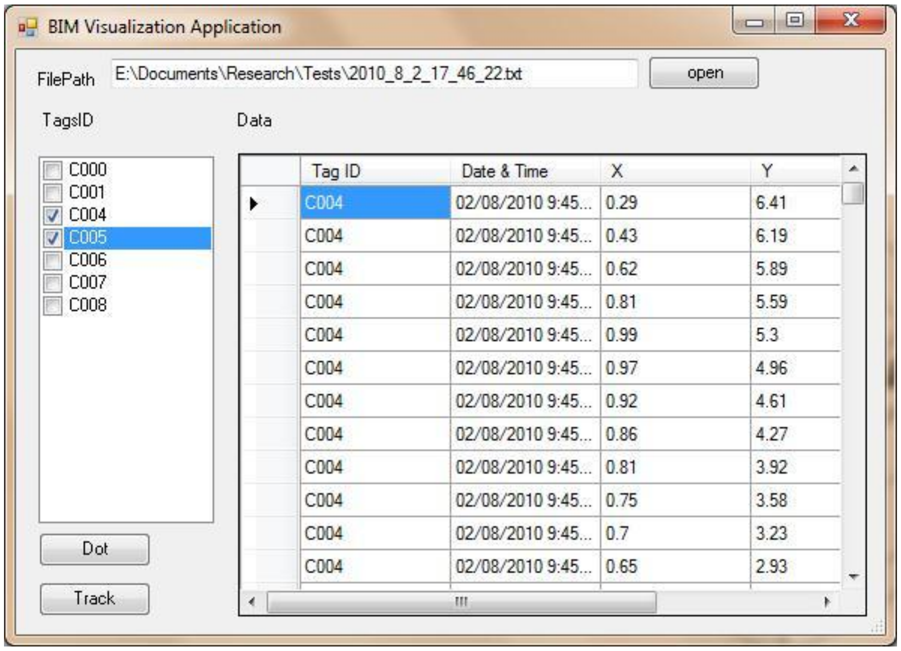


BIM Visualization Application
Plug-in application for visualizating real-time location data files in BIM

Figure 5-18 Tab of BIM Visualization Application

2) Importing Real-Time Location Data

After downloading *txt* files from the ***Tag Positioning Web Application*** as described in Subsection 5.3.3, the user should click the Open button to import real-time location data from the *txt* file and list the tag IDs in the Tag IDs checklist box. This program is able to select specific tags depending on the purpose. By selecting the tag IDs, for example C004 and C005, their data will be shown in the Data grid view. By clicking on the head of a column, for example the Tag ID tab, the rows will be sorted by tag IDs. The result is illustrated in Figure 5-19.



FilePath: E:\Documents\Research\Tests\2010_8_2_17_46_22.txt

TagsID:
- [ ] C000
- [ ] C001
- [✓] C004
- [✓] C005
- [ ] C006
- [ ] C007
- [ ] C008

Tag ID	Date & Time	X	Y
C004	02/08/2010 9:45...	0.29	6.41
C004	02/08/2010 9:45...	0.43	6.19
C004	02/08/2010 9:45...	0.62	5.89
C004	02/08/2010 9:45...	0.81	5.59
C004	02/08/2010 9:45...	0.99	5.3
C004	02/08/2010 9:45...	0.97	4.96
C004	02/08/2010 9:45...	0.92	4.61
C004	02/08/2010 9:45...	0.86	4.27
C004	02/08/2010 9:45...	0.81	3.92
C004	02/08/2010 9:45...	0.75	3.58
C004	02/08/2010 9:45...	0.7	3.23
C004	02/08/2010 9:45...	0.65	2.93

Figure 5-19 Window of BIM Visualization Application

3) Visualizing in BIM

The user should click the Track button to read real-time location data from a data table and draw the traces of tags in the BIM. For example as illustrated in Figure 5-20, there are two traces of tags C004 and C005 appearing in the BIM. Each green dot represents the position of a tag at a specific time. Linking consecutive dots represents the trace of a tag in a recording period. Therefore, the awareness visualization though BIM is legible for observation, which can help users to distinguish different movements of tags and avoid confusion.

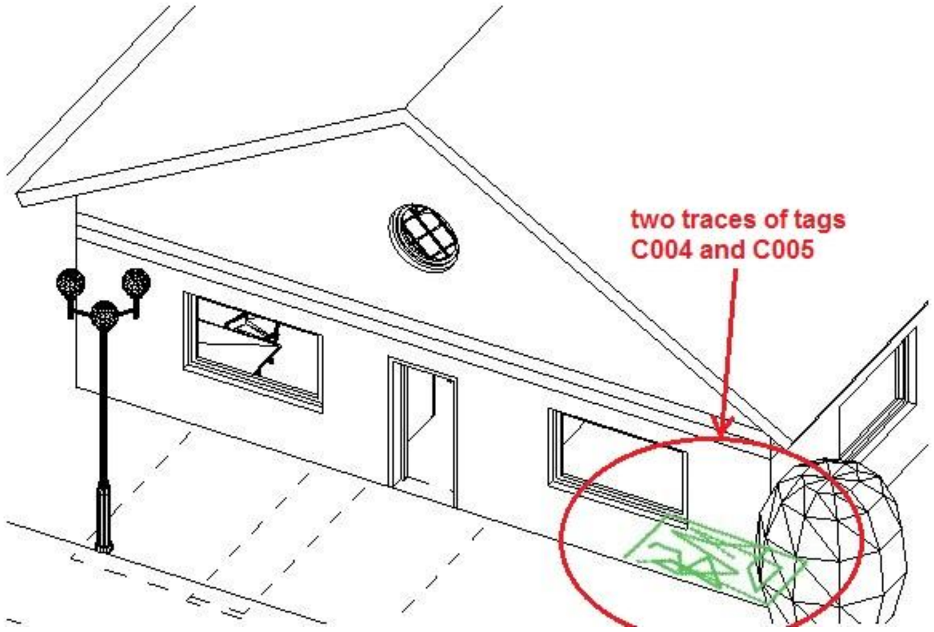


Figure 5-20 Awareness Visualization in BIM

Programming Details

Command.cs

The class *Command.cs* opens the plug-in application window of the ***BIM Visualization Application*** when it is executed in Autodesk Revit Architecture. It includes the first user-written function, which is similar to the main function in some programming languages. It needs to be defined in *Revit.ini* as the class name:

```
ECClassName1=Tag_Track_BIM_Visualization_Application.Command
```

Creator.cs

The class *Creator.cs* creates model lines and dots in the view of Autodesk Revit Architecture. It creates model lines by linking two consecutive locations in the building model to represent traces of tags and dots by crossed model lines in the building model to represent locations of tags.

DataManager.cs

The class *DataManager.cs* instances its object to store all real-time location data from the selected file. The real-time location data of selected tag IDs have been transformed and stored in the object of DataTable. The type of Tag ID is string, the type of time is DataTime, and the types of X, Y, and Z coordinates are float. Those data will be used by this plug-in application of Autodesk Revit Architecture.

FileReading.cs

The class *FileReading.cs* imports all real-time location data from the selected file.

MainForm.cs

The class *MainForm.cs* frames the window of this external tool program. The user interface is mainly designed in this class.

Limitations

(1) Limitation of colors makes it impossible to distinguish different tags and their traces in different colors. Autodesk Revit Architecture .NET APIs does not allow to the switching of model line colors. In some tests, a mass of messy dots crowding in a corner of BIM does not allow this plug-in application to provide a legible and efficient awareness visualization to users.

(2) Limitation of view makes it impossible to change the view angle and zoom in or zoom out when real-time location data are visualized as model lines in BIM.

5.5 Summary and Conclusions

In order to implement the Web service design in Chapter 4, the integrated components and the programming platform are reviewed, and three programming modules are presented in detail. The ***UWB RTLS Web Service*** is tested in three simulated tests and applied in two practical tests in Chapter 6.

CHAPTER 6 TESTS

6.1 Introduction

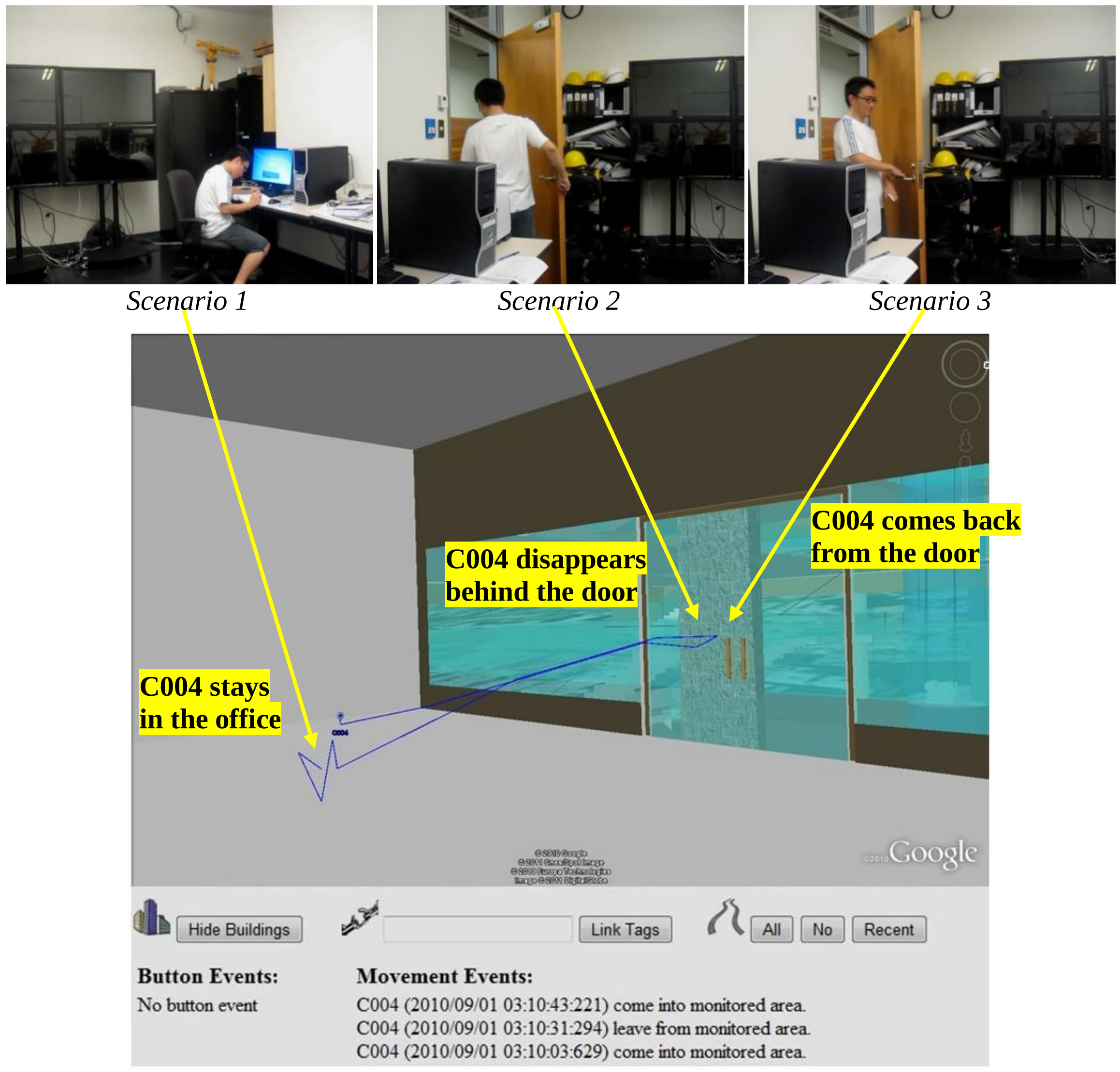
The three programming modules for UWB RTLS are implemented in Chapter 5. The main purpose of this chapter is to test the ***UWB RTLS Web Service*** in three simulated tests and apply it in two practical tests by using the ***Tag Positioning Web Application***. Firstly, this chapter briefly describes three simulated tests which simulate the three aforementioned typical UWB RTLS applications to evaluate the capabilities of the ***UWB RTLS Web Service*** in terms of how well its requirements were satisfied. Then, it introduces two practical tests (tracking workers in a construction project and tracking the motion of a crane) which were completed by our research group.

6.2 Simulated Tests

In order to test the capabilities of the ***UWB RTLS Web Service*** in terms of how well its requirements described in Section 3.5 were satisfied, three simulated tests are designed and executed in our lab based on the three typical UWB RTLS applications described in Section 3.3.

6.2.1 Employee Monitoring

In the simulated test for employee monitoring, a researcher acted as an employee working in the office, and a tag was chosen for tracking: C004 attached to the employee ID card carried by the employee. As illustrated in Figure 6-1 (C004 in blue), the three scenarios are: (1) the employee working in the office; (2) the employee gets a phone call and leaves the office for a while; and (3) the employee returns to the office and resumes work. The system can record all these scenarios.



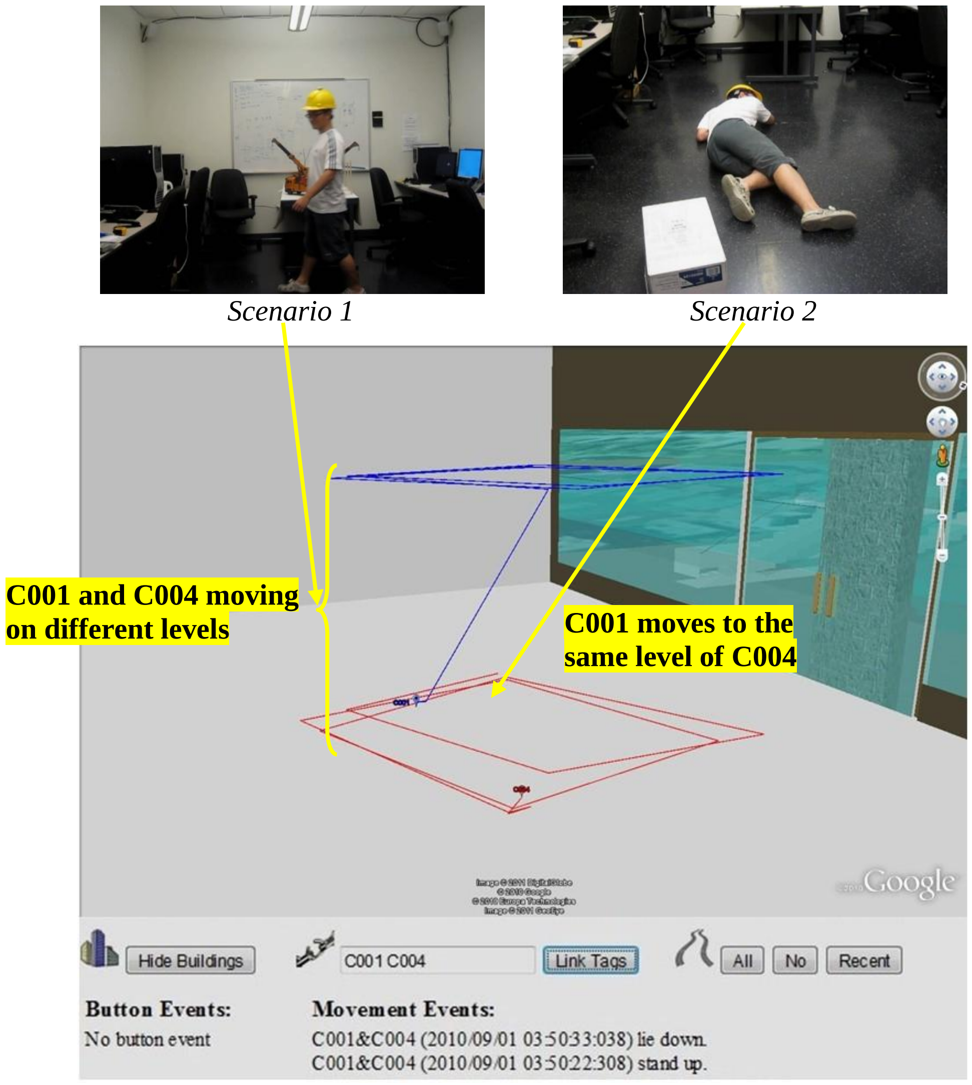
Button Events:
No button event

Movement Events:
C004 (2010/09/01 03:10:43:221) come into monitored area.
C004 (2010/09/01 03:10:31:294) leave from monitored area.
C004 (2010/09/01 03:10:03:629) come into monitored area.

Figure 6-1 Visualization of Simulated Test of Employee Monitoring

6.2.2 Fire Rescue

In the simulated test for a fire rescue, a researcher acted as a firefighter searching for victims in a burning building, and two tags were chosen for tracking: (1) C001 attached to the hardhat of the firefighter; and (2) C004 attached to the boot of the firefighter. As illustrated in Figure 6-2 (C001 in blue and C004 in red), the two scenarios are: (1) the firefighter walking in the burning building; and (2) the firefighter gets injured and lies down on the ground. The system can record all these scenarios.



Button Events:
No button event

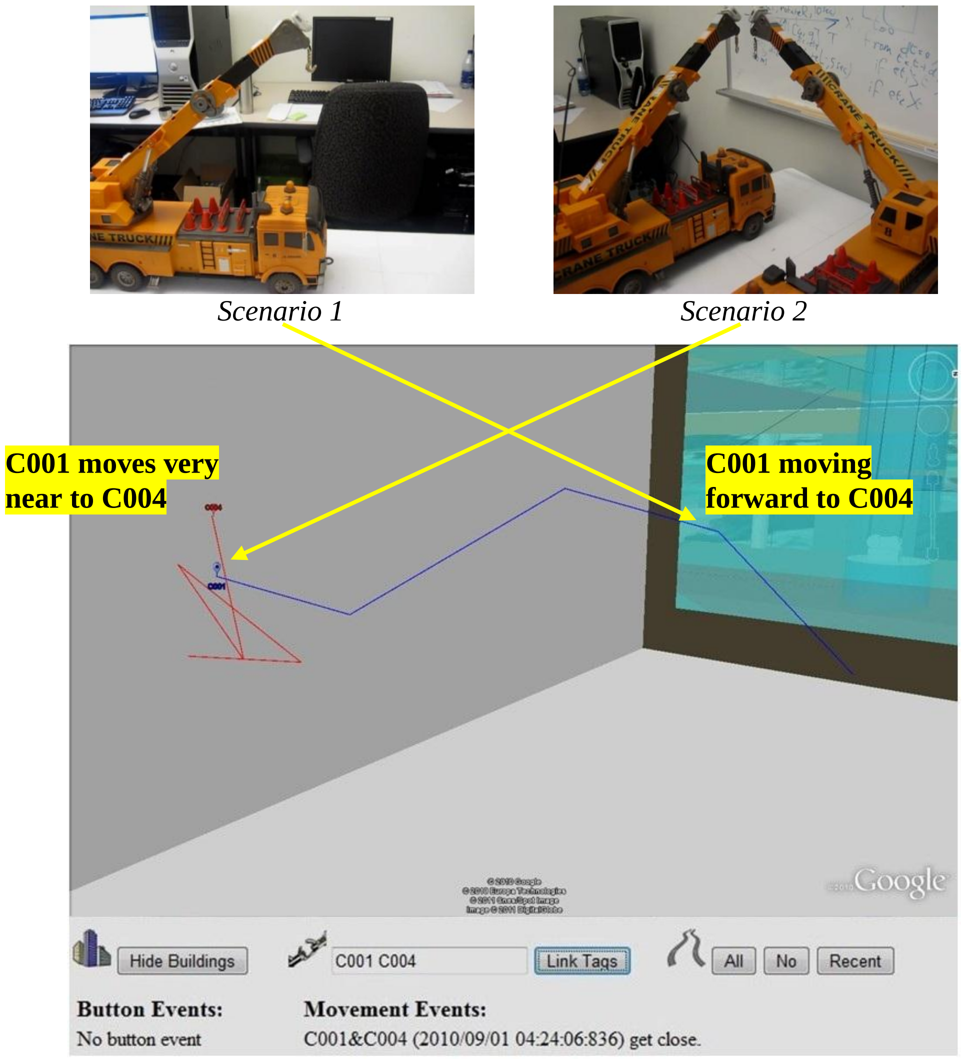
Movement Events:
C001&C004 (2010/09/01 03:50:33:038) lie down.
C001&C004 (2010/09/01 03:50:22:308) stand up.

Figure 6-2 Visualization of Simulated Test of Fire Rescue

6.2.3 Crane Operation

In the simulated test for crane operation, two crane toys were working near each other, and two tags were chosen for tracking: (1) C001 attached to the boom tip of the moving crane; and (2) C004 attached to the boom tip of the inactive crane. As illustrated in

Figure 6-3 (C001 in blue and C004 in red), the two scenarios are: (1) a crane moving forward to another crane; and (2) the moving crane gets close to the inactive crane and their booms almost collide. The system can record all these scenarios.



Button Events:
No button event

Movement Events:
C001&C004 (2010/09/01 04:24:06:836) get close.

Figure 6-3 Visualization of Simulated Test of Crane Operation

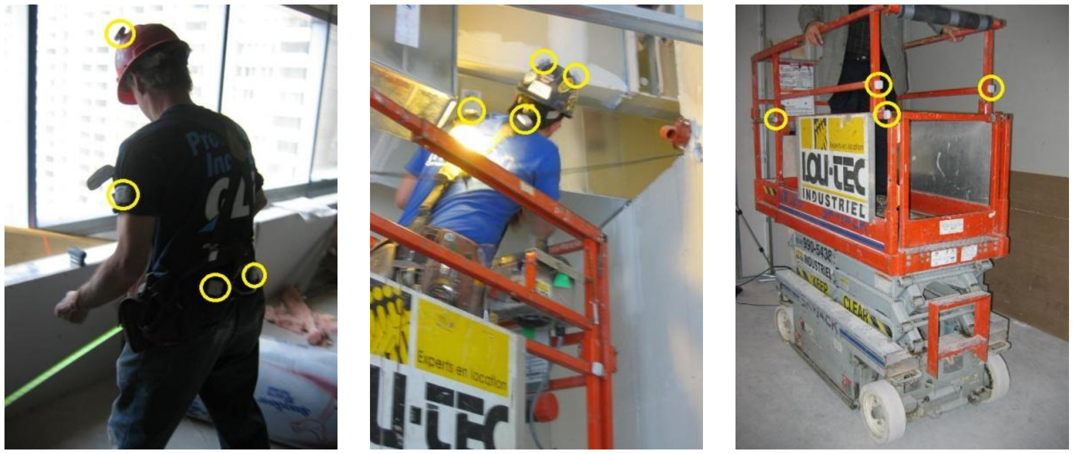
6.3 Practical Tests

The two practical tests were undertaken by our research group to evaluate the advantages of UWB RTLS in improving productivity and safety. The ***UWB RTLS Web Service*** plays a supporting role in the practical tests.

6.3.1 Tracking Workers in a Construction Project

On April 29th and 30th 2009, an indoor test was done on the 7th floor of the John Molson School of Business (JMSB) building in the Concordia SGW Campus. The test focuses on investigating the performance of UWB RTLS for detecting, locating and tracking workers and equipment on a construction site. The test details and results can be found in (Zhang, 2010) and (Rodriguez, 2010).

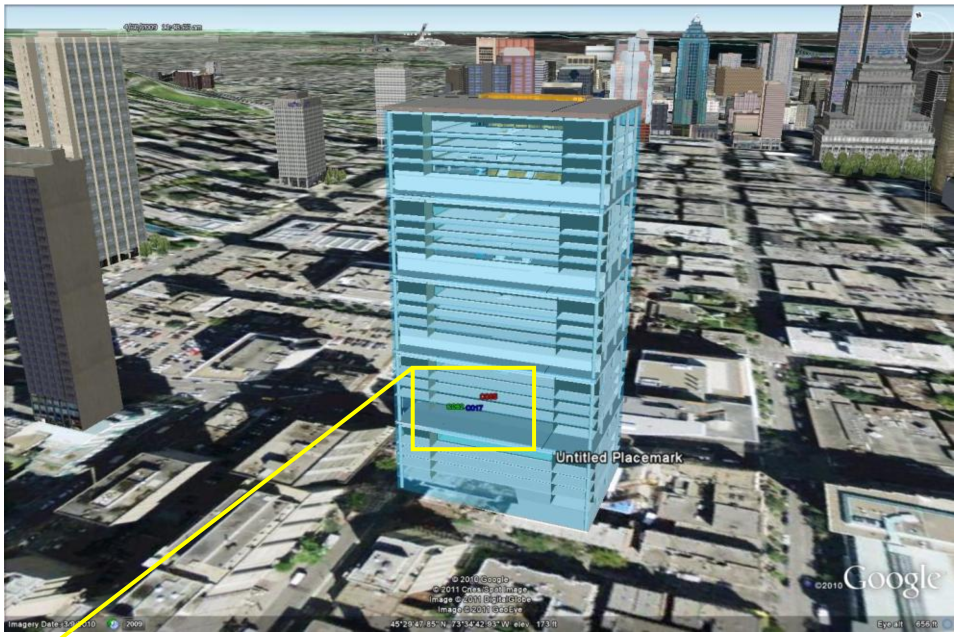
In this test, several tags were attached to workers (hardhats and belts) and equipment (different parts of a scissor lift). The details of tagged positions are illustrated in Figure 6-4 and can be found in Appendix G. Real-time location data were collected during workers and equipment working on the installation of heating, ventilating and air conditioning (HVAC) ducts.



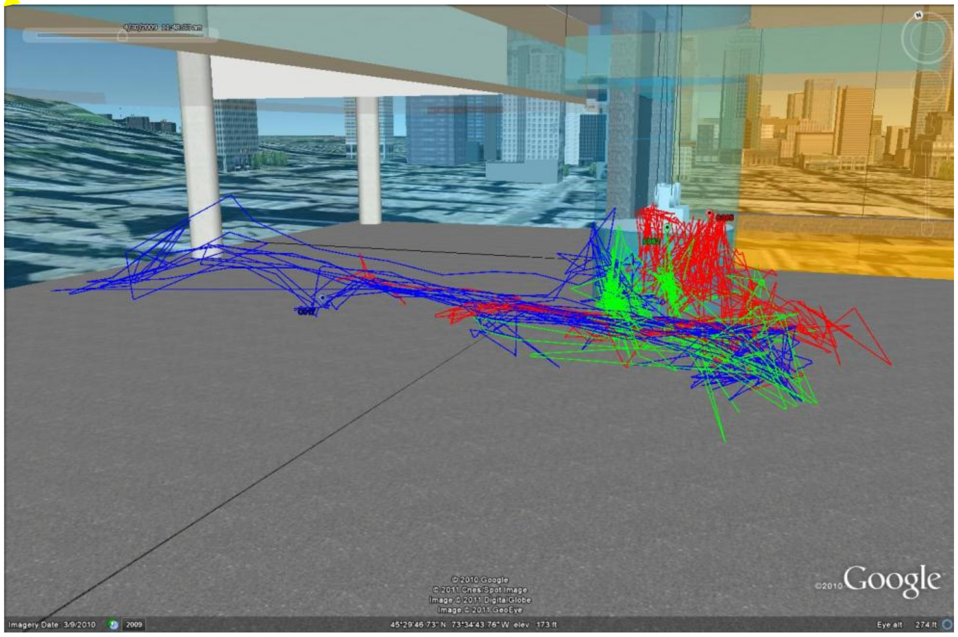
(a) Worker-1 *(b) Worker-2* *(c) Scissor Lift*

Figure 6-4 Tag Positions in Test of Tracking Workers in a Construction Project

A construction site in JMSB Building was used as the monitoring location, and three tags were chosen for tracking in this test: (1) C017 attached to worker-1 hardhat; (2) C085 attached to worker-2 hardhat; and (3) C082 attached to the scissor lift. The monitoring results are illustrated in Figure 6-5 (C017 in blue, C085 in red, and C082 in green).



(a) Overview Building Model



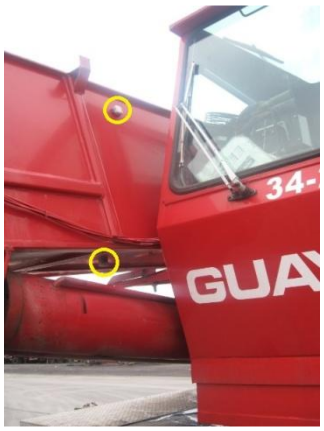
(b) Detailed Tag Movements

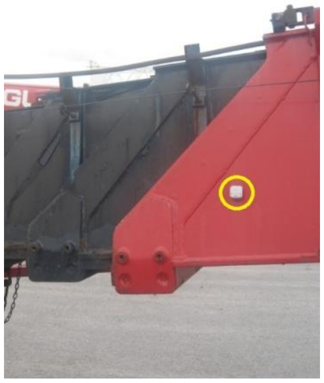
Figure 6-5 Awareness Visualization of Test of Tracking Workers in a Construction Project

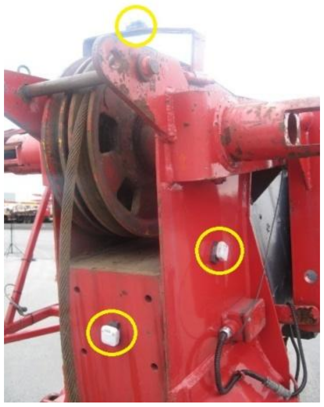
6.3.2 Tracking the Motion of a Crane

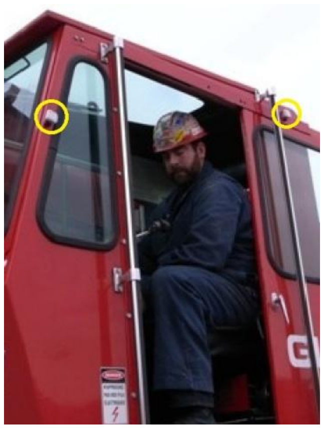
On December 4th 2009, an outdoor test was done in the yard of the Guay Company with a TMS300 crane. The test focuses on the crane to capture the crane movements and actions. The test details and results can be found in (Zhang, 2010) and (Rodriguez, 2010).

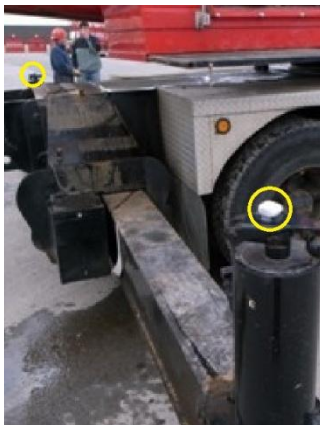
In this test, several tags were attached to different parts of the crane (e.g. boom, hook, and outriggers), lifted object, and worker hardhats. The details of tagged positions are illustrated in Figure 6-6 and can be found in Appendix H. Real-time location data were collected during the crane lifting an object back and forth and workers walking around the working range of the crane.

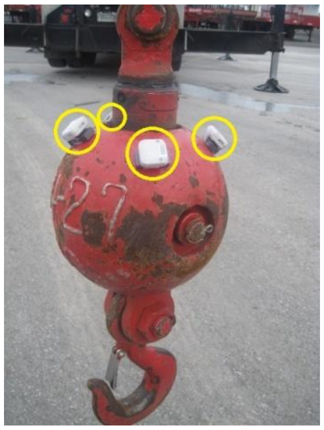

(a) First Section Boom Base

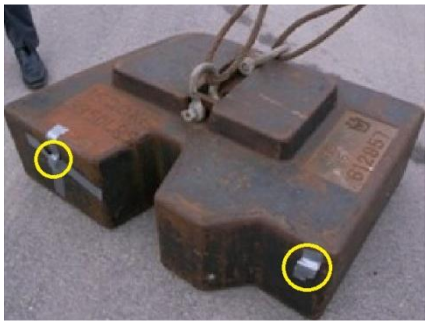

(b) *First Section Boom Tip*


(c) *Boom Tip*


(d) Operator Cab


(e) Outrigger


(f) Hook


(g) Lifted Object

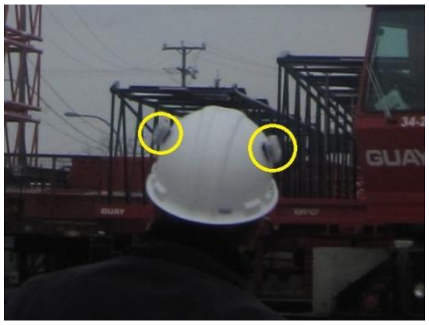

(h) Hardhat

Figure 6-6 Tag Positions in Test of Tracking the Motion of a Crane

The yard of the Guay Company was used as the monitoring location, and three group of tags were chosen from three parts of the boom to be tracked: (1) C004, C165, C006 and C007 attached to first section boom base; (2) C008, C009, C010 and C012 attached to first section boom tip; and (3) C013, C014, C015 and C016 attached to boom tip. The monitoring results are illustrated in Figure 6-7 (right: C004 in blue, C165 in red, C006 in green, and C007 in yellow; middle: C008 in blue, C009 in red, C010 in green, and C012 in yellow; left: C013 in blue, C014 in red, C015 in green, and C016 in yellow).

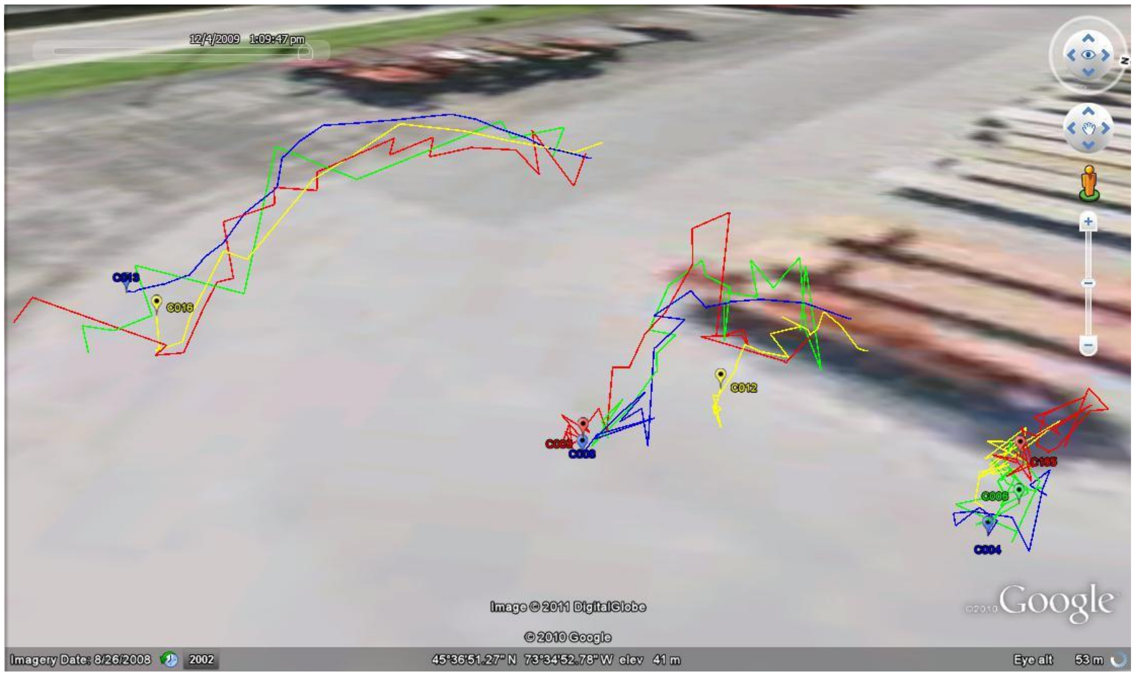

12/4/2009 1:09:47 pm
C013
C016
C012
C009
C008
C165
C006
C004
Image © 2011 DigitalGlobe
© 2010 Google
Google
Imagery Date: 8/26/2008 2002
45°36'51.27" N 73°34'52.78" W elev 41 m
Eye alt 53 m

Figure 6-7 Awareness Visualization of Test of Tracking the Motion of a Crane

6.4 Summary and Conclusions

After implementing the three programming modules in Chapter 5, the ***UWB RTLS Web Service*** was tested in three simulated tests based on the three aforementioned typical UWB RTLS applications and applied in two practical tests by using the ***Tag Positioning Web Application***. The results of these tests satisfied the requirements of the ***UWB RTLS Web Service***, and it is possible to use the Web service developed in this research in practical UWB RTLS applications.

CHAPTER 7 CONCLUSIONS & FUTURE WORK

The main purpose of this chapter is to summarize the present work, highlight the contributions of this research, and suggest recommendations for future research.

7.1 Summary of Research

This research investigates the feasibility of extending the scope of UWB RTLS applications further into industrial and daily life areas.

In the definition of requirements, after defining common requirements of UWB RTLS applications by exploring the specific requirements of the three typical UWB RTLS applications, developing Web services for UWB RTLS is considered as the best approach to enhance its capabilities and satisfy these common requirements. In order to design the Web service for UWB RTLS in Chapter 4, the requirements of the ***UWB RTLS Web Service*** are determined based on the common requirements, and the five fundamental use cases of the ***UWB RTLS Web Service*** are analyzed.

In the Web service design for UWB RTLS, the ***UWB RTLS Web Service*** is designed to cover the solutions of technology integration and collaboration, remote interaction, awareness visualization, and event detection in order to satisfy the common requirements of UWB RTLS applications. The design details of the ***UWB RTLS Web Service***, such as its namespace decision and interface design, are also presented.

In the implementation, in order to implement the Web service design, the integrated components and the programming platform are reviewed, and three programming modules are presented in detail.

In the test, after implementing the three programming modules in Chapter 5, the ***UWB RTLS Web Service*** was tested in three simulated tests based on the three aforementioned typical UWB RTLS applications and applied in two practical tests by using the ***Tag Positioning Web Application***. The results of these tests satisfied the requirements of the ***UWB RTLS Web Service***, and it is possible to use the Web service developed in this research in practical UWB RTLS applications.

7.2 Research Contributions and Conclusions

The major contributions of this research are summarized in the following:

(1) The definition of the common requirements of UWB RTLS applications was successfully used to determine the requirements of the Web service developed in this research.

(2) The development of the Web service — ***UWB RTLS Web Service*** enhanced the capabilities of UWB RTLS and satisfied the common requirements by implementing technology integration and collaboration, remote interaction, awareness visualization, and event detection.

To satisfy the requirements of technology integration and collaboration, the Web service is designed to serve as middleware which integrates UWB RTLS, WMS applications, databases, and time-management Web applications together in a system and collaborates their services. The Web service collects real-time location data from UWB RTLS, generates visualization data in a specific file format of geographic data used in a WMS application, saves real-time location data and event messages into a database, and saves event messages into a time-management Web application.

To satisfy the requirements of remote interaction, the Web service is designed to provide services invoked by other applications. The Web service is on a remote server system hosting the requested services to customized applications developed by users. Those applications can access the Web service though the WWW and execute remote interaction using the services provided by the Web service to achieve instant remote monitoring, monitoring result sharing, and remote UWB RTLS control.

To satisfy the requirements of awareness visualization, the Web service is designed to convert real-time location data into visualization data in a specific file format of geographic data used in a WMS application. Awareness visualization includes the feature of 4D digital display for UWB RTLS indoor monitoring on general Web browsers. The design of awareness visualization explores variations on how to best to represent a range of features to provide efficiency in further monitoring and analysis.

To satisfy the requirements of event detection, the Web service is designed to detect events and generate event messages for monitoring purposes and further analysis. Event messages are either presented in Web pages or saved into databases and time-management Web applications.

(3) The development of applications using the services provided by the Web service — ***Tag Positioning Web Application*** and ***BIM Visualization Application*** — demonstrated the usefulness of the Web service.

7.3 Future Work

In practical applications, a large amount of real-time location data is collected by the ***UWB RTLS Web Service***. Unfortunately, most of them are inaccurate, redundant and irrelevant, so that real-time location data analysis plays an important role to solve this problem. In future extensions, the Web service should be improved by using artificial intelligence in real-time location data analysis to filter out these data. Real-time location data analysis can be used by inspecting, cleaning, transforming, and modeling data to perform accuracy analysis and redundancy elimination. Accuracy analysis is to analyze the accuracy of real-time location data and find out which rows of the data have low accuracy, such as when a tag is unreasonably jumping out of its trace. Redundancy elimination is to eliminate the rows of real-time location data which are redundant, such as when a tag is not moving.

REFERENCES

3D Modeling (2010). <http://en.wikipedia.org/wiki/3D_modeling> [accessed March 2010].

Abdul-Latif, O., Shepherd, P., and Pennock, S. (2007). TDOA/AOA data fusion for enhancing positioning in an ultra wideband system. *IEEE International Conference on Signal Processing and Communications (ICSPC), Dubai, United Arab Emirates, 2007*.

Autodesk (2010). <http://usa.autodesk.com/> [accessed January 2010].

Carbonari, A., Naticchia, B., Giretti, A., and Grassi, M. D. (2009). A Proactive System for Real-Time Safety Management in Construction Sites. *The 26th International Symposium on Automation and robotics in Construction, Austin, Texas, U.S., 2009*.

Cerruti, A., Kintner, P. M., Gary, D. E., Mannucci, A. J., Meyer, R. F., Doherty, P. H., and Coster, A. J. (October 19th 2008). Effect of intense December 2006 solar radio bursts on GPS receivers. *Space Weather*, Vol. 6, S10D07, p. 10.

Cho, Y. K., Youn, J. H., and Martinez, D. (2010). Error Modeling for an Untethered Ultra-Wideband System for Construction Indoor Asset Tracking. *Automation in Construction*, Vol. 19, No. 1, pp. 43-54.

City of Sheridan (2009). City of Sheridan GIS. <http://www.city-sheridan-wy.com/info/pwd-pd-gis/index.php> [accessed April 2010].

Clarinox (2009). <http://www.clarinox.com/> [accessed November 2009].

Cockburn, A. (2001). *Writing Effective Use Cases*, pp. 15-16. Canada: Addison-Wesley.

Colwell, S. (April 2008). UWB Location Tech on A Roll: Ultra-Wideband Location Applications Envisioned or Already Created Support Emergency Services, Asset Tracking, and Manufacturing Inventory Management. GPS World. Available at: <http://findarticles.com/p/articles/mi_m0BPW/is_4_19/ai_n25431499/> [accessed in December 2009].

Crane Accident (2009). Crane Accident Statistics. <http://www.craneaccidents.com/stats.htm> [accessed December 2009].

CSST (2009). Commission de la sant é et de la s écurit é du travail du Qu ébec. <http://www.csst.qc.ca/portail/fr> [accessed December 2009].

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhl, N., and Weerawarana, S. (2002). Unraveling the Web Services Web an Introduction to SOAP WSDL and UDDI. IEEE Internet Computing. Available at: <http://www.cc.gatech.edu/classes/AY2004/cs6210_fall/papers/00991449.pdf> [accessed October 2010].

Domdouzis, K., Kumar, B. and Anumba, C. (2007). Radio-frequency identification (RFID) applications: A brief introduction. *Advanced Engineering Informatics*, Vol. 21, No. 4, pp. 350-355.

ESRI (2010). What Is GIS? <http://www.esri.com/what-is-gis/index.html> [accessed March 2010].

Ferris, C. and Farrell, J. (June 2003). What are Web Services? *IBM Communications of the ACM*, Vol. 46, No. 6, p. 31.

Fire Government (2009). Fire Fighter Location. <http://www.fire.gov/locator/index.htm> [accessed April 2010].

GEAbios (2010). <http://www.geabios.com/> [accessed February 2010].

Ghavami, M., Michael, L. B., and Kohno, R. (2004). *Ultra-wideband signals and systems in communication engineering*. Chichester: John Wiley & Sons.

Glass, G. (November 1st 2000). The Web Services (R)evolution: Part 1 - Applying Web Services to Applications. IBM. Available at: <http://www.ibm.com/developerworks/Webservices/library/ws-peer1.html> [accessed September 2009].

Google Code (2010). <http://code.google.com/intl/en/more/> [accessed February 2010].

Google Earth (2010). <http://earth.google.com/> [accessed February 2010].

Gordon, V.R. and Holness (June 2008). Building Information Modeling Gaining Momentum. *ASHRAE Journal*, pp. 28-40. Available at: <http://findarticles.com/p/articles/mi_m5PRB/is_6_50/ai_n32064740/> [accessed October 2009].

GPS (2010). The Global Positioning System Serving the World. <http://www.gps.gov/> [accessed March 2010].

Guan, H., Jin, B., Wei, J., and Xu, W. (2005). A Framework for Application Server Based Web Services Management. *12th Asia-Pacific Software Engineering Conference (APSEC'05), Taipei, Taiwan, December 15th - 17th 2005*.

Haas, H. and Brown, A. (February 2004). Web Services Glossary. W3C Working Group Note. Available at: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211> [accessed September 2010].

Heinrich, C. E. (2005). *RFID and Beyond: Growing Your Business through Real World Awareness*. Indianapolis, Indiana: Wiley Publishing.

Hitachi (2010). Indoor Messaging System. <http://www.hitachi.com/rd/sdl/people/gps_module/01.html> [accessed March 2010].

Hong, S. and Feuerlicht, G. (2008). *Web Service Interface Design for e-Business Application: A Minimalistic Design Approach*, p. 143. Department of Information Technology, University of Technology, Sydney, 2008 IEEE.

IBM (2005). Information home: IBM WebSphere Application Server. <http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp> [accessed November 2009].

ISP (2010). Indoor Positioning System. <http://en.wikipedia.org/wiki/Indoor_Positioning_System> [accessed March 2010].

Janelli, H. V. (2005). Web Services Applications for RFID. Available at: <http://proceedings.ndia.org/693A/Janelli.pdf> [accessed February 2010].

Kerer, C., Dustdar, S., Jazayeri, M., Gomes, D., Szego, A., and Burgos Caja, J. A. (2004). Presence-Aware Infrastructure using Web services and RFID technologies. Distributed Systems Group, Information Systems Institute, Vienna University of Technology. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.7955&rep=rep1&type=pdf> [accessed February 2010].

Khemlani, L. (2007). 2007 Third Annual BIM Awards, Part 1. Available at: <http://www.aecbytes.com/buildingthefuture/2007/BIM_Awards_Part1.html> [accessed June 2009].

Lahiri, S. (August 31st 2005). *RFID Sourcebook*. Westford, Massachusetts: IBM Press.

Landt, J. (2005). The history of RFID. *IEEE Potential*, Vol. 24, No. 4, pp. 8-11.

Lawyers and Settlements (2010). Fire Accidents. <http://www.lawyersandsettlements.com/case/fire_accident.html> [accessed May 2010].

Lee, G., Sacks, R., and Eastman, C. M. (2006). Specifying parametric building object behavior (BOB) for a building information modeling system. *Automation in Construction*, Vol. 15, No. 6, pp. 758-776.

Legner, C. and Vogel, T. (2007). Design Principles for B2B Serviecs – An evaluation of two alternative service designs. *IEEE SCC, 2007*, pp. 372-379. Institute of Information Management, University of St. Gallen. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.2509&rep=rep1&type=pdf> [accessed July 2010].

Malik, A. (2009). *RTLS for Dummies*. Hoboken, NJ: Wiley Publishing.

Microsoft (2009). .NET Tutorial for Beginners. <http://download.microsoft.com/download/8/e/7/8e725d96-7ec3-498b-9fa7-86779aed101f/dotNET%20Tutorial%20for%20Beginners.pdf> [accessed September 2009].

Motamedi, A. (April 2009). Framework for Lifecycle Management of Facilities Components Using RFID Technology. *ITcon* Vol. 14 (2009), *Motamedi & Hammad*, pp. 238-262. Available at: <http://www.itcon.org/data/works/att/2009_18.content.01307.pdf> [accessed October 2009].

Muñoz, D., Bouchereau, F., Vargas, C., and Enriquez-Caldera, R. (2009). *Position Location Techniques and Applications*. Burlington, MA: Academic Press.

Open Geospatial Consortium (2010). Web Map Service. <http://www.opengeospatial.org/standards/wms> [accessed October 2009].

Oracle (2009). J2EE Tutorial. <http://download.oracle.com/docs/cd/E17410_01/javaee/6/tutorial/doc/> [accessed September 2009].

Papazoglou, M. P. (2008). *Web Services: Principles and Technology*. Edinburgh Gate, England: Pearson Education Limited.

Poirier, C. and McCollum, D. (2006). *RFID Strategy Implementation and ROI: A practical Roadmap to Success*. Fort Lauderdale, Florida: J. Ross Publishing.

Reuters (2009). New satellite navigation system may save fire-fighters. <http://uk.reuters.com/article/idUKL1263470020071212> [accessed April 2010].

Rodriguez, S. (2010). Experimental Study on Location Tracking of Construction Resources Using UWB for Better Productivity and Safety. Master Thesis Paper. Concordia Institute for Information systems Engineering, Department of Engineering and Computer Science, Concordia University, Montreal.

Scharl, A. and Tochtermann, K. (2007). *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society*, p. 225. Advanced Information and Knowledge Processing Series 2007. London: Springer.

Shepard, S. (2005). *RFID: Radio Frequency Identification*. USA: The McGraw-Hall Companies.

Tang, W. and Selwood, J. (2003). *Connecting our World GIS Web Services*, p. 2. Redlands, California: ESRI.

Teizer, J., Venugopal, M., and Walia, A. (2008). Ultrawideband for Automated Real-Time Three-dimensional Location Sensing for Workforce, Equipment, and Material Positioning and Tracking. *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2081, pp. 56-64.

Ubisense (2009). <http://www.ubisense.net/en/> [accessed November 2009].

Updated Crane Report (2008). Crane-Related Deaths in Construction and Recommendations for Their Prevention. <http://www.cpwr.com/research-cranereport.html> [accessed December 2009].

Vawter, C. and Roman, E. (2001). J2EE vs. Microsoft.NET: A comparison of building XML-based Web services. Available at: <http://www.theserverside.com/news/1365389/J2EE-vs-MicrosoftNET-A-comparison-of-building-XML-based-Web-services> [accessed September 2009].

Ward, A., Jones, A., and Hopper, A. (October 1997). A New Location Technique for the Active Office. Available at: <http://Web.sau.edu/LillisKevinM/wirelessbib/WardJonesHopper.pdf> [accessed August 2009].

Web API (2010). <http://en.wikipedia.org/wiki/Web_API> [accessed February 2010].

Work Safe BC (September 2008). Work Safe Bulletin. Available at: <http://www.worksafebc.com/news_room/news_releases/assets/nr_07_01_12/crane_accidents_mobile.pdf> [accessed December 2009].

Zhang, C. (2010). Improving Crane Safety by Agent-Based Dynamic Motion Planning Using UWB Real-Time Location System. PHD Thesis Paper. Department of Building, Civil and Environmental Engineering, Concordia University, Montreal.

Zhang, C., Hammad, A., and AlBahnassi, H. (2009). Collaborative Multi-Agent Systems for Construction Equipment Based on Real-Time Field Data Capturing. *ITcon*, Vol. 14, *Special Issues on Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning*, pp. 204-228. Available at: <http://www.itcon.org/cgi-bin/works/Show?2009_16> [accessed November 2009].

APPENDIX A COMPARISON OF AOA, TDOA, AND RSSI

Table A-1 Comparison of AOA, TDOA, and RSSI (Zhang et al., 2009)

	AOA	**TDOA**	**RSSI**
Antenna Type	Directional Antenna (Monopole), multimode antenna.	Any based on requirement.	Monopole, bi-directional, multimode.
Algorithms	Triangulation, along with TDOA applied to array of antenna.	Triangulation / Trilateration. Difference in arrived signal time and transmitted signal time gives estimated distance.	Triangulation. Received signal strength (RSS) inversely proportional to square of distance.
Accuracy	Potentially high. Determined by the position of the angles, and the robustness of installation.	Potentially very high dependant on clock synchronization.	Variable as received signal strengths changes with changing energy of transmitted signals
Advantages	When used with TDOA, can be an accurate measurement of the position of asset.	Efficiently used by the GPS satellites, US Federal Communication Commission.	Simplest of all implementation techniques. Low cost.
Disadvantages	Costly depending on size of antenna array. Often used with TDOA which makes cost very high.	Very costly as makes use of expensive electronic quartz clocks to maintain synchronization.	Lower level of accuracy.

APPENDIX B SPECIFICATIONS OF UBISENSE SERIES 7000 SENSOR

Table B-1 Specifications of Ubisense Series 7000 Sensor (Ubisense, 2009)

Size and Weight	
Dimensions	20cm x 13cm x 6cm (8” x 5” x 2.5”)
Weight	650g (23 oz)
Operating Conditions	
Temperature	0 ℃ to 60 ℃ (32 ℉ to 140 ℉)
Humidity	0 to 95%, non-condensing
Enclosure	
IP30	
Location Performance	
Operating Range	Up to 160m (520ft) in open field conditions
Achievable Accuracy	Better than 30cm (12”) in 3D
Radio Frequencies	
Ultra-wideband	6GHz – 8GHz
Telemetry channel	2.4GHz
Certifications	
FCC Part 15 (FCC ID SEASENSOR20)	
EU CE	

Power Supply
Power-over-Ethernet IEEE 802.3af compatible 12V DC @ 10W (optional)
Mounting Options
Adjusting mounting bracket (supplied)
Ubisense Part Codes
UBISENSOR7000, UBISENSPS (optional 12V power supply)

APPENDIX C PROCEDURES OF UBISENSE SYSTEM INSTALLATION AND CALIBRATION

1. Needed Equipment

a) Sensors

b) Mounting equipment

c) Cables (4 regular network cables and 3 thick timing cable)

d) Computer

e) Software: Ubisense package, DHCP server, solver

f) Switch and power: PoE switch plus its power cables, power generator for outdoor scenario

g) Level

h) Cutter

2. Layout Design Steps

a) Conceptual connectivity design (daisy chain, star, extended start)

b) Decide on where to put the sensors in the yard

c) Decide on where to put one tag for calibrating the sensors

d) Decide on two, easy to measure, dots on the yard

e) Decide on how to run the cables and protect them

f) Decide on where to put the switch

g) Draw the connectivity map

h) Decide on the reference dot (0, 0, 0)

i) Fill out table (Calibration tag ID info, Sensors: X, Y, and Z, and MAC)

3. Site Preparation

Note: activities with letter “P” could be done in parallel

P1: Fix the sensors on the mounting device on the designated place

P1: Put the switch on its designated place and attach to the power

P1: Run the network cables from switch to sensors and fix the cables

P1: Run the timing cables based on the connectivity map (most often between the sensors)

4. Measurements

P2-0: Measure the “area”: W, L, H

P2-1: Measure X, Y, and Z of the sensors (one by one or using the solver)

a) Decide on two dots and measure the x, y of them, (preferably we set the (0, 0) on the corner of two walls in the area)
b) Measure the distance from the dots to the sensors
c) Enter them in the solver
d) Get the XYZ of the sensors
e) P2-2: Measure the reference dot for calibration (x y z), (could be the same dots in the last step)

5. Basic Configuration

a) Note: sensors are now turned off because they are not attached to the switch
b) All cables should be connected properly (check if they are loos/ timing cables are connected to upper left)

c) Restart the PC (switch is powered, sensors are not connected, PC is connected to switch)

d) Attach the computer to the switch

e) Start DHCP server

f) Open platform control

g) Make sure the services are running (no prefix, not in standalone mode)

h) Open location engine

i) Open log tab

j) Connect the sensor cables

k) Looking at logs to see if we receive logs

l) Move sensors to cell

m) Define master (checkboxes, ...)

n) Look at the LEDs

6. Software Configuration

a) Note: So far the sensor MAC addresses should be under “available sensors”

b) Open “site manager”

c) Area tab

d) Open note pad

e) Type the coordinates of the area the file (P2-0)

f) Save in dat file

g) Create walls in the area tab > load walls > load dat

h) Go to the cell tab and load the area

i) Extend the cell

j) Open location engine

k) Load area and cell

l) Drag the sensors to the area

m) Select the master and check: “master”, “timing source” and “disable sleep”

n) Check the RF power of the cell (must be 250)

o) Check the LEDs (should be solid green)

p) Enter XYZ of the sensors (using P2-1)

q) Put the tag on the calibration dot (using P2-2)

r) Do the dual calibration

APPENDIX D SPECIFICATIONS OF COMPACT AND SLIM TAGS

Table D-1 Specifications of Compact and Slim Tag (Ubisense, 2009)

	Compact Tags	**Slim Tags**
Size & Weight	Height 40 mm (1.5 inches) Width 40 mm (1.5 inches) Depth 15 mm (0.6 inches) Weight 30 g (1 Ounce)	Height 83 mm (3.26 inches) Width 42 mm (1.65 inches) Depth 10 mm (0.39 inches) Weight 35 g (1.2 Ounce)
Operating Temperature	Standard -20 C to 60 C (-4 F to 140 F) Extended -30 C to 70 C (-22 F to 158 F)	Standard -20 C to 60 C (-4 F to 140 F) Extended -30 C to 70 C (-22 F to 158 F)
Mounting Options	Multiple including screw mounts, cable ties, self adhesive and mounting plate	Multiple including mounting plate and attachment to personnel badges
Operating Frequencies	Ultrawideband: 6 GHz – 8 GHz Telemetry channel: 2.4 GHz	Ultrawideband: 6 GHz – 8 GHz Telemetry channel: 2.4 GHz
Beacon Rate	Dynamically variable (from 1 per minute to 10 per second)	Dynamically variable (from 1 per minute to 10 per second)

APPENDIX E SAMPLE OF TXT FILE

```
C000 2010/11/05 21:02:23:515 1.94 6.77 0.00 False 0 0 1
C001 2010/11/05 21:02:23:538 0.27 3.81 0.00 False 0 0 1
C004 2010/11/05 21:02:23:564 0.27 6.46 0.00 False 0 0 1
C005 2010/11/05 21:02:23:589 2.44 3.09 0.00 False 0 0 1
C000 2010/11/05 21:02:23:689 2.17 6.55 0.00 False 0 0 2
C001 2010/11/05 21:02:23:715 0.58 3.81 0.00 False 0 0 2
C004 2010/11/05 21:02:23:740 0.43 6.19 0.00 False 0 0 2
C005 2010/11/05 21:02:23:765 2.55 3.39 0.00 False 0 0 2
C000 2010/11/05 21:02:23:865 2.22 6.23 0.00 False 0 0 3
C001 2010/11/05 21:02:23:890 0.93 3.82 0.00 False 0 0 3
C004 2010/11/05 21:02:23:915 0.62 5.89 0.00 False 0 0 3
C005 2010/11/05 21:02:23:940 2.68 3.71 0.00 False 0 0 3
C000 2010/11/05 21:02:24:039 2.21 5.88 0.00 False 0 0 4
C001 2010/11/05 21:02:24:065 1.28 3.82 0.00 False 0 0 4
C004 2010/11/05 21:02:24:090 0.81 5.59 0.00 False 0 0 4
C005 2010/11/05 21:02:24:115 2.81 4.04 0.00 False 0 0 4
C000 2010/11/05 21:02:24:215 2.20 5.53 0.00 False 0 0 5
C001 2010/11/05 21:02:24:240 1.63 3.82 0.00 False 0 0 5
C004 2010/11/05 21:02:24:265 0.99 5.30 0.00 False 0 0 5
C005 2010/11/05 21:02:24:290 2.93 4.36 0.00 False 0 0 5
C000 2010/11/05 21:02:24:390 2.19 5.18 0.00 False 0 0 6
C001 2010/11/05 21:02:24:415 1.98 3.82 0.00 False 0 0 6
C004 2010/11/05 21:02:24:440 0.97 4.96 0.00 False 0 0 6
C005 2010/11/05 21:02:24:465 3.06 4.69 0.00 False 0 0 6
C000 2010/11/05 21:02:24:565 2.18 4.83 0.00 False 0 0 7
C001 2010/11/05 21:02:24:589 2.33 3.83 0.00 False 0 0 7
C004 2010/11/05 21:02:24:615 0.92 4.61 0.00 False 0 0 7
C005 2010/11/05 21:02:24:640 3.19 5.02 0.00 False 0 0 7
C000 2010/11/05 21:02:24:740 2.17 4.48 0.00 False 0 0 8
C001 2010/11/05 21:02:24:765 2.68 3.83 0.00 False 0 0 8
C004 2010/11/05 21:02:24:790 0.86 4.27 0.00 False 0 0 8
C005 2010/11/05 21:02:24:814 3.31 5.34 0.00 False 0 0 8
C000 2010/11/05 21:02:24:914 2.15 4.13 0.00 False 0 0 9
C001 2010/11/05 21:02:24:939 3.03 3.83 0.00 False 0 0 9
C004 2010/11/05 21:02:24:964 0.81 3.92 0.00 False 0 0 9
C005 2010/11/05 21:02:24:990 3.43 5.62 0.00 False 0 0 9
C000 2010/11/05 21:02:25:089 2.14 3.78 0.00 False 0 0 10
C001 2010/11/05 21:02:25:114 3.38 3.83 0.00 False 0 0 10
C004 2010/11/05 21:02:25:140 0.75 3.58 0.00 False 0 0 10
C005 2010/11/05 21:02:25:164 3.38 5.27 0.00 False 0 0 10
…
```

APPENDIX F   SAMPLE OF KML FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
 <Document>
  <name>Trace of tags</name>
  <description>Show traces of tag in Google Earth by using the data</description>
  <Style id="bluePoint">
   <IconStyle>
    <color>ffffffff</color>
    <colorMode>normal</colorMode>
    <scale>0.5</scale>
    <Icon>
     <href>http://maps.google.com/mapfiles/kml/paddle/blu-circle.png</href>
    </Icon>
    <hotSpot x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
   </IconStyle>
   <LabelStyle>
    <color>ffff0000</color>
    <colorMode>normal</colorMode>
    <scale>0.5</scale>
   </LabelStyle>
  </Style>
  ...
  <Placemark>
   <name>C000</name>
   <styleUrl>#bluePoint</styleUrl>
   <description>real-time location of a tag</description>
   <TimeSpan>
    <begin>2010-11-05T21:02:23Z</begin>
    <end>2010-11-05T21:02:24Z</end>
   </TimeSpan>
   <Point>
    <extrude>0</extrude>
    <tessellate>0</tessellate>
    <altitudeMode>relativeToGround</altitudeMode>
    <coordinates>114.161617319607,22.2792555124634,100</coordinates>
   </Point>
  </Placemark>
  …
  <Style id="blueLine">
   <LineStyle>
    <color>ffff0000</color>
    <width>1.5</width>
   </LineStyle>
   <PolyStyle>
    <color>ffffffff</color>
    </PolyStyle>
   </Style>
   ...
   <Placemark>
    <name>C000</name>
    <description>Path or link of a trace of tag</description>
    <styleUrl>#blueLine</styleUrl>
    <LineString>
     <extrude>0</extrude>
     <tessellate>0</tessellate>
     <altitudeMode>relativeToGround</altitudeMode>
     <coordinates>114.161617319607,22.2792555124634,100
114.161620178028,22.2792555761976,100 114.161622528286,22.2792538553749,100
114.161624687983,22.2792515609448,100 114.161626847679,22.2792492665146,100
114.161629007376,22.2792469720844,100 114.161631167072,22.2792446776542,100
114.161633326768,22.2792423832241,100 114.161635422944,22.2792400250597,100
114.161637582641,22.2792377306295,100 114.161639742337,22.2792354361994,100
114.161641584431,22.27923346044,100 114.161643426525,22.2792314846807,100
114.161645268619,22.2792295089214,100 114.161647110713,22.2792275331621,100
114.161648952807,22.2792255574028,100 114.161650794901,22.2792235816435,100
114.161652319393,22.279221924555,100 114.161653843884,22.2792202674665,100
114.161655431897,22.2792186741123,100 114.161656956388,22.2792170170238,100
114.16165848088,22.2792153599353,100</coordinates>
    </LineString>
   </Placemark>
   ...
  </Document>
</kml>
```

APPENDIX G TAGGED POSITIONS IN TEST OF TRACKING WORKERS IN A CONSTRUCTION PROJECT

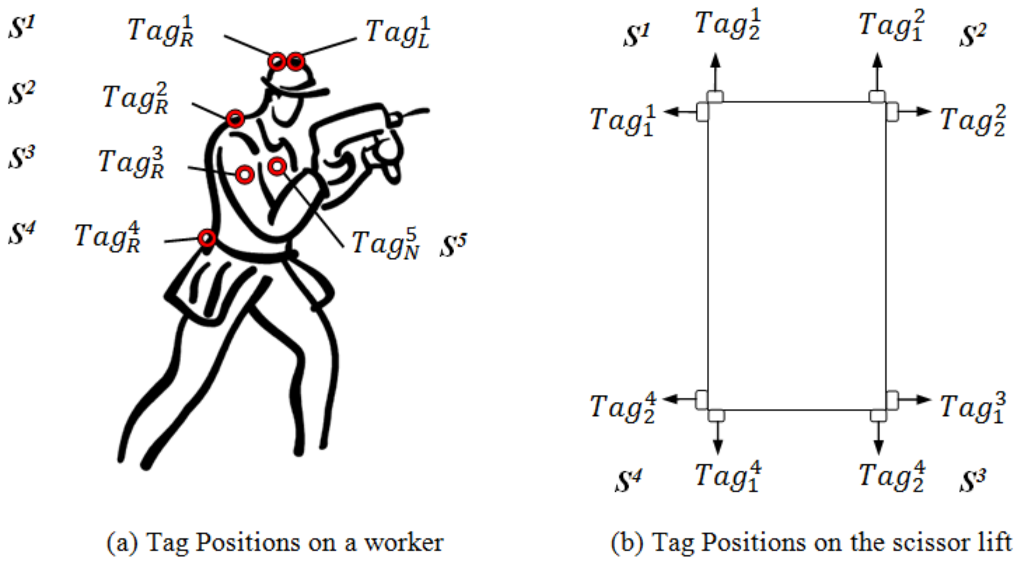

S^1 Tag_R^1 Tag_L^1
S^2 Tag_R^2
S^3 Tag_R^3
S^4 Tag_R^4
Tag_N^5 S^5
(a) Tag Positions on a worker
S^1 Tag_2^1 Tag_1^2 S^2
Tag_1^1 Tag_2^2
Tag_2^4 Tag_1^3
S^4 Tag_1^4 Tag_2^4 S^3
(b) Tag Positions on the scissor lift

Figure G-1 Tag Positions in Test of Tracking Workers in a Construction Project (adapted from Rodriguez, 2010)

Table G-1 Tag Positions in Test of Tracking Workers in a Construction Project

Description	Position		Tag Name	Tag ID
Worker-1 (6 tags)	S^1	hardhat – right	$W_1 - Tag_R^1$	C162
		hardhat – left	$W_1 - Tag_L^1$	C017
	S^3	arm – right	$W_1 - Tag_R^3$	C075
		arm – left	$W_1 - Tag_L^3$	C007
	S^4	belt – right	$W_1 - Tag_R^4$	C009

		belt – left	$W_1 - Tag_L^4$	C089
Worker-2 (4 tags)	S^1	hardhat – right	$W_2 - Tag_R^1$	C085
		hardhat – left	$W_2 - Tag_L^1$	C077
	S^2	shoulder – right	$W_2 - Tag_R^2$	C247
		shoulder – left	$W_2 - Tag_L^2$	C098
Scissor lift (8 tags)	S^1	corner 1-1	$E_1 - Tag_1^1$	C082
		corner 1-2	$E_1 - Tag_2^1$	C070
	S^2	corner 2-1	$E_1 - Tag_1^2$	C029
		corner 2-2	$E_1 - Tag_2^2$	C078
	S^3	corner 3-1	$E_1 - Tag_1^3$	C015
		corner 3-2	$E_1 - Tag_2^3$	C014
	S^4	corner 4-1	$E_1 - Tag_1^4$	C087
		corner 4-2	$E_1 - Tag_2^4$	C010

APPENDIX H TAGGED POSITIONS IN TEST OF TRACKING THE MOTION OF A CRANE

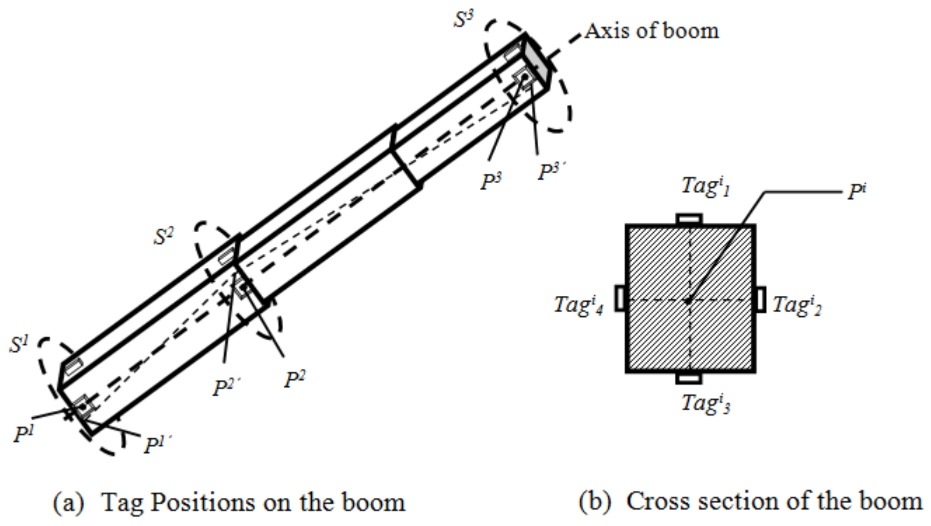


Figure H-1 Tag Positions in Test of Tracking the Motion of a Crane (adapted from Zhang, 2010)

Table H-1 Tag Positions in Test of Tracking the Motion of a Crane

Description	Position		Tag Name	Tag ID
First Section Boom Base (4 tags)	S_1	top	t_{11}	C004
		right	t_{12}	C165
		bottom	t_{13}	C006
		left	t_{14}	C007
First Section Boom Tip (4 tags)	S_2	top	t_{21}	C008
		right	t_{22}	C009
		bottom	t_{23}	C010
		left	t_{24}	C012

Boom Tip (4 tags)	S_3	top	t_{31}	C013
		right	t_{32}	C014
		bottom	t_{33}	C015
		left	t_{34}	C016
Outriggers (2 tags)	S_4	left	t_{41}	C079
		right	t_{42}	C164
Operator Cab (2 tags)	S_6	front	t_{61}	C080
		rear	t_{62}	C081
Hook (4 tags)	S_7		t_{71}	C098
			t_{72}	C099
			t_{73}	C073
			t_{74}	C075
Person 1 — professor (2 tags)	S_8	hardhat – right	t_{81}	C088
		hardhat – left	t_{82}	C089
Person 2 — student (2 tags)	S_9	hardhat – right	t_{91}	C091
		hardhat – left	t_{92}	C092
Lift object (2 tags)				C162
				C247