Prophet: A Context Aware Framework for Construction Site Applications

By

Vipul Singhvi

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree

Master of Science

Thesis Committee:

Dr. Michael Terk, Assistant Professor, Thesis Advisor

Dr. Satish Nagarajaiah, Associate Professor

Dr. Ahmad J. Durrani, Professor

Department of Civil and Environmental Engineering, Rice University Houston, Texas
August 2003
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
Abstract

Prophet: A Context Aware Framework for Construction Site Applications

By

Vipul Singhvi

Construction sites are information intensive environments. The timely and satisfactory completion of any project depends on the accessibility of project information to the construction project engineers. Up to now, accessing this information at the point of work, where and when it will be most effective, has been difficult or impossible. This work describes Prophet, a context aware, framework for the development of information system that integrates context-aware applications to provide construction workers with on-site contextual data and services. Prophet is designed to take advantage of off-the-shelf portable computing devices and uses wireless network for communication between the users and servers that host applications and data. Prophet is designed to simplify the creation of systems that use the knowledge of user's context to help in managing the complexity of the construction data. This is accomplished by proactively tracking current resource requirements and proactively obtaining access to context-relevant information and services. This thesis describes the capabilities of the Prophet framework and provides a description of a sample implementation used to demonstrate the benefits of this approach.
# Table of Contents

**Title** .................................................................................................................. I  
**Abstract** ............................................................................................................... II  
**Table of Contents** .............................................................................................. III  
**List of Figures** ................................................................................................. V  
**Chapter 1** ........................................................................................................... 1  
**Introduction** ....................................................................................................... 1  
**Chapter 2** ........................................................................................................... 7  
**Understanding Context and Context Awareness** ........................................... 7  
  2.1 Model for Contextual Information Processing ........................................ 7  
    2.1.1 Defining Context .................................................................................. 9  
    2.1.2 Establishing Context .......................................................................... 10  
    2.1.3 Managing Context ............................................................................. 11  
    2.1.4 Designing Context Aware Applications ........................................... 11  
  2.2 Definition of Context .................................................................................... 12  
  2.3 Establishing Context .................................................................................... 18  
    2.3.1 Global Positioning System (GPS) ....................................................... 20  
    2.3.2 Active Badge .................................................................................... 20  
    2.3.3 Active Bat ......................................................................................... 21  
    2.3.4 Cricket .............................................................................................. 23  
    2.3.5 Radar ............................................................................................... 24  
    2.3.6 Smart Floor ....................................................................................... 25  
    2.3.7 Electromagnetic Sensing ................................................................... 25  
**Chapter 3** ........................................................................................................... 27  
**Software Architecture for Context Aware Systems** ................................... 27  
  3.1 Requirements for Prophet Architecture .................................................. 28  
  3.2 Related Software Architecture .................................................................. 31  
    3.2.1 Schilit's System Architecture ............................................................ 31  
    3.2.2 Stick-e Notes Architecture ............................................................... 32  
    3.2.3 Cyber Desk ...................................................................................... 33  
    3.2.4 Context Toolkit ............................................................................... 34  
    3.2.5 Blackboard Architecture .................................................................. 35  
    3.2.6 Service Infrastructure Based Architecture ...................................... 36  
    3.2.7 Cooltown ......................................................................................... 37  
    3.2.8 Context and Location Aware Information Service (CALAIS) ........... 38  
    3.2.9 Context Information Service (CIS) ................................................. 38  
    3.2.10 Situated Computing Service ............................................................ 39  
**Chapter 4** ........................................................................................................... 41  
**Prophet: Software Architecture** ................................................................. 41  
  4.1 Definition of Context ............................................................................... 42  
  4.2 Establishing Raw Context ......................................................................... 44  
  4.3 Prophet Software Architecture .................................................................. 46  
    4.3.1 Service Module .............................................................................. 48  
    4.3.2 Registry Module .............................................................................. 50
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.3 Sensor Module</td>
<td>50</td>
</tr>
<tr>
<td>4.3.4 Context Module</td>
<td>51</td>
</tr>
<tr>
<td>4.3.5 User Interface Module</td>
<td>52</td>
</tr>
<tr>
<td>4.3.6 Cache Module</td>
<td>52</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>54</td>
</tr>
<tr>
<td>Prophet: Software Architecture Implementation</td>
<td>54</td>
</tr>
<tr>
<td>5.1 Sensor Module</td>
<td>56</td>
</tr>
<tr>
<td>5.2 Registry Module</td>
<td>57</td>
</tr>
<tr>
<td>5.3 Context Module</td>
<td>58</td>
</tr>
<tr>
<td>5.4 Cache Manager Module</td>
<td>59</td>
</tr>
<tr>
<td>5.5 Service Module</td>
<td>60</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>62</td>
</tr>
<tr>
<td>Conclusion and Future Work</td>
<td>62</td>
</tr>
<tr>
<td>Future Work</td>
<td>63</td>
</tr>
<tr>
<td>Reference</td>
<td>65</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 Model of Information Processing ....................................................... 8
Figure 2 Active Badges .................................................................................. 21
Figure 3 Active Bat ....................................................................................... 23
Figure 4 Prophet Software Architecture ....................................................... 47
Figure 5 iPAQ 3850, GPS and Wireless Card ............................................... 54
Figure 6 Schematic of Web Service ............................................................... 60
Figure 7 User Interface .................................................................................. 61
Chapter 1

Introduction

Construction projects are information intensive work environments. In general, construction projects require large information repositories that are used to store knowledge, data, drawings and documents generated by various people and organizations that take part in the construction project. Even a small commercial construction project consists of a several hundred architectural and structural drawings that detail various aspects of the structure and the construction process. These drawings are used by project contractors to generate additional contracts shop drawings based on the structural and architectural drawings. This makes the construction project a highly information intensive process that involves a range of sources of communication.

The information complexity of construction projects and sites are further complicated because of the dynamic and ad-hoc nature of the construction sites. As a result, the information related to a construction project changes can drastically through different stages of progress of the construction project. The information involved is complex and is constantly updated and revised by the architectures and engineers. This requires
engineers and construction inspectors to regularly make onsite visits to monitor and ensure that the construction progresses according to the most recent drawings and contract documents. Engineers are often confronted with unforeseen practical problems at the construction site, which require them to access information and software applications not readily available at the job site. This lack of access to information and tools often leads to construction delays and budget overruns.

The drawings and documents are normally available in paper or electronic format, both formats present issues when carrying this information on-site. In paper format construction drawings are generally available in A1 or A0 sheet sizes. The sheer size and their low manageability make it impossible for the engineers and inspectors to carry more than a handful of drawings and documents onsite. As a result, engineers and inspectors end up carrying less information then they might require for onsite visits.

Portable computing offers, engineers and inspectors, an improved method for onsite access to information by giving them the ability to retrieve, utilize, update and relay project data directly from the field. When structural drawing and contract documents are available in electronic format they can be stored in the portable devices and retrieved onsite when and where required. Until recently costs were prohibitive to take portable computers to site but with the decreasing cost of portable computing devices makes them appealing for construction applications, their limitations in storage capacity, processing power and user interface presents a challenge in accessing, synthesizing and modifying the large amounts of data associated with modern construction projects. Use of portable
computing also raises new challenges. Engineers and inspectors find it difficult to use the portable devices because they require a significant user input and attention. During their onsite visits it is difficult for the engineers and inspectors to focus both on the situation/task at hand and on using the portable devices. This often leads to dissatisfaction of users with portable computing as an information delivery mechanism.

The software architecture and the application discussed in the thesis are designed to help the construction project engineers and inspectors to use portable computing devices by intelligently assessing their information requirements and proactively retrieving that information and presenting it in a convenient way. The system extends the hardware support available in the form of portable computing devices by providing a software layer to make best use of low storage and processing capacity of the portable devices.

The thesis describes the Prophet software architecture and its implementation. Prophet is a context aware information system designed to address the problems of non-availability of information onsite and lack of processing and storage capacity of portable devices. The system senses the requirements of the user by analyzing the situation in which the user is involved. The situation, consisting of users, tools etc., are modeled in the form of entities (people and objects) and events interacting with each other based on their individual attributes and interdependencies among each other. Context in the information system refers to the attributes of the entities and events and their interdependencies among each other. The information system analyzes the context of each entity and object involved in the situation to determine possible needs of the user of the system. The information
system abstracts the real world situation by understanding context of various entities and objects involved in the situation. The closer the definition of context is to real life, the better the support provided by the context aware information system.

Construction sites are very complex environments. Every entity and event has a complex set of attributes that affect the situation in which they are involved. The challenge for the context aware information system is in the complexity of defining, capturing, representing and processing contextual data at a construction site.

In recent years there has been a lot of development in context aware application and systems. These applications support and enhance the abilities of its users in executing tasks. These tasks range from navigating through an unfamiliar space to providing reminders for activities or just keeping track of people in the building. These applications and systems, though capable of supporting context awareness in a fixed-static environment, lack the ability to manage in an ad-hoc and dynamic environment. As a result, present context aware applications are unable to model the dynamic situation and ad-hoc environment encountered at a construction site.

The Prophet context aware information system is implemented by utilizing portable devices, carried by onsite workers. The portable devices connect to the database and information stored in servers (computers) at the main offices. The communication between the portable device and the servers can be implemented by providing wireless or local area network at the construction site. We have provided a WiFi (802.11b) network
for communication as it fulfills the requirements of low cost and low infrastructure at a construction site. The servers in the main office have all the required information, documents and drawings that could be required at the construction site. The system provides the portable devices the capability to sense the context and operate on it by using the services provided in the system. The services operate on context and provide the user with information relevant to the context sensed by their portable devices. These services are either available locally on the portable device or are available remotely through the wireless network. The communication between the portable devices and the main server is ephemeral as there are a lot of interferences that are encountered at a construction site. We have designed the Prophet system to require a minimum amount of onsite support to provide for a graceful degradation of network services in case the communication between the portable devices and the main server is lost.

The work presented here has three main goals:

- To provide an operational understanding of context at construction sites from which we can derive concepts useful for context aware application development for construction sites.
- To introduce a software architecture to assist in the design of context aware applications for construction sites.
- To provide an implementation of the software architecture defined in the thesis.

In the next chapter we present an overview of current understanding and uses of context. We also propose a definition for context and identify different types of context based on
ways they are collected. In chapter 3 we present requirements for the architecture that support the acquisition, representation, delivery and reaction to context information that can be automatically sensed and used as implicit input to affect application behavior. We then review the present software architectures that are available for development of context aware applications. We highlight the need for a new framework for development of context aware applications for construction sites due to the dynamic nature of the construction sites. In chapter 4 we present the Prophet Software architecture, for the design of context aware applications for construction sites. In chapter 5 we present a prototype of a few applications developed based on the Prophet architecture. Chapter 6 presents the conclusions and proposes several directions for future work.
Chapter 2

Understanding Context and Context Awareness

2.1 Model for contextual information processing

Humans have an innate ability to establish, understand and use context to help them to interact with the environment. Context refers to information which humans require to evaluate the situation and make a decision for their action in a given situation. As a simple illustrative example consider a situation when a person is late for a meeting in an office. In this case their response would be to start walking faster towards the meeting room. Here the context of the situation is the combination of the meeting’s time and the person’s location at that time. The person understands that since he or she has not reached the location of the meeting room at the specified time, he or she better reach there as soon as possible. People handle similar and more complex situations encountered in their day-to-day life by processing the contextual information in their mind.

Contextual human information processing model is referred as the ecological approach to model information processing [Wickens 1992]. Ecological approach emphasizes on the integrated flow of information through the human being and the interaction with the
environment. It focuses on modeling the perceptual characteristics of the environment to which the user is "tuned" and responds in order to meet the goals of a particular task. Action and perception are linked, since to act is to change what is perceived and to perceive is to change the basis of the action. As a consequence of its properties, the ecological approach is most directly relevant to describing human behavior in interaction with the natural environment (e.g. walking or driving).

Wickens (1992) has suggested a model for ecological information processing in humans, as it is shown in Figure 1.

![Model of Information Processing](image)

**Figure 1 Model of information processing**

The above model consists of various processing stages which are depicted as black boxes.

The challenge to develop a context aware information system, to assist humans in their interaction with the environment, is to mirror the way human process information as it is modeled in Figure 1. Each human is unique in the way they are involved in environment, the way they perceive the environment and the way their actions change the environment.
The challenges of a context aware information system are to model the environment in an objective way and yet subjectively cater to the need of each human using the information system. The main issues identified in the research for development of a context-aware information system are:

- Defining context
- Establishing context
- Managing context
- Designing context aware applications

In the following paragraphs we briefly highlight the main challenges in each of these areas.

2.1.1 Defining context

The definition of context is of most importance as it forms the basis of information processing in the context-aware information system. Context acts as a primary key by which the system would relate the humans with the knowledge and resources available to them for interacting with the environment. As shown in figure 1, interaction between humans and the environment is very contextual, dynamic and iterative. To successfully model the interaction we need a definition of context which is flexible and extensive to accommodate the dynamics of human action in the environment. The definition of context has to be operational so that it can be abstracted in form of a standard data structure in the context-aware information system. The definition of context has to be specific for each human using the information system and at the same time it has to be general enough to accommodate everybody using the system. Later in the chapter we will
evaluate various definitions of context that have been used in context aware research and provide the definition developed for Prophet.

2.1.2 Establishing context

Establishing context is the next important step in developing a context-aware information system. The definition of context would help in identifying various elements of the environment that are a part of context and to establish those elements of context we need to have a mechanism to establish the values from the environment. One easy approach would be to let humans, who are using the system, provide contextual information. This approach though easy would however be inefficient as it poses an added responsibility on the humans to keep track of this information and continuously provide it to the information system. A more automated approach is needed to keep track of the context in the environment.

Sensor networks provide a promising solution to continuously monitor environment and provide values to establish the contextual information. However, sensor networks are limited in their ability to sense information. Their performance is limited also due by the lack of support infrastructure and communication requirements. To integrate sensor networks, capable of establishing certain elements of context, in the information system we have substantiate the partial contextual information to generate complete context. In chapter 3 we talk about various sensor networks we have evaluated for establishing context at a construction site.
2.1.3 Managing context

Context forms the basis of interaction between the humans, the environment and the context aware information system. It is important to have a flexible and extensible data format to represent context in the system. As discussed earlier context is a dynamic concept so need to be constantly updated as the user of the context aware information system interact with the environment. Partial contextual information, which can be established using sensor networks, has to be substantiated the information by providing relevant information though other sources, for example, human input or automated services to substitute human intervention. Contextual information forms the basis of interaction between various parts of the context-aware information system it is important to have a standard way to communicate contextual information among various parts of the system. Data structures representing context should be same across the system to provide the ease to integrate information from various sources.

2.1.4 Designing context aware applications

Context awareness refers to dynamically using context to provide relevant information and services to support and enhance the abilities of its users for executing tasks in a particular operating environment. Designing context aware application for a context aware information system is a challenge, as present application used by humans are not designed to use and process contextual information. Context aware adapters need to be written to make the traditional application context aware. Also to integrate different application to interact with each other based on the system requirements requires a standard protocol need to be adopted. The communication protocol should be flexible to
accommodate the varied kind of applications used in the context aware information system.

2.2 Definition of Context

As discussed in previous section defining context is of central importance in designing a context aware information system. The requirements for context definition are: flexible and extensible to accommodate the dynamic interaction between humans and the environment, operational to help develop a standard abstract data structure that can be used to describe the context based on the way the elements are established and their importance in the context aware information system and specific enough to provide user-specific information and still be general to model the environment. In this section we evaluate definitions, given by various researchers, for their suitability for Prophet.

There have been a number of definitions for context given over the years. These definitions range from providing general references of environment to more specific and operational definition. Definitions that provide broad general references to environment, define context as the whole environment of user or the application. For example, Franklin and Flaschbart [Franklin et al., 1998] defined context as the situation of the user. Ward et al. [Ward et al., 1997] defined context as the state of the application’s surroundings or the environment and Rodden et al. [Rodden et al., 1998] defined it as the application’s setting or the environment. Hull et al. [Hull et al., 1997] included the entire environment by defining context to be aspects of the current situation. These definitions help in understanding the concept of context as a model of real world but they do not aid in development of context aware applications. The broad focus of the definition makes it
difficult to uniquely and discretely identify contextual information (information that affects the situation at hand) from other incidental information that is not required for the design of the context aware systems and applications.

As a result, many researchers have given a more specific definition for context. They define context as elements of the user's environment that the system knows about (Brown [Brown, 1996]). Dey [Dey, 1998] defines context as information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. These definitions are more specific than the previous set of definitions but still not operational. The definitions leave the interpretation of what constitutes relevant context to developers. This might lead to contradictions in development of data structures to represent context during the design of context aware systems. The definitions, though more specific than the previous set, cannot be prescriptively used in the design of context aware information system. We seek a more operational approach to define context in our information system at a construction site. The definition of context should model the environment by identifying information relevant in the situations the system is likely to encounter. The system should be able to construct context, using the definition of context, in a way that makes it easy to create, modify and use it.

The definitions by Schilit, Adams, and Want [Schilit et al., 1994], is closest to the operational and specific definition we seek. Schilit [Schilit et al., 1994] claimed that the
important aspects of context are: where the user is, whom the user is with, and what resources are nearby. They define context to be the constantly changing execution environment. The execution environment consists of three layers:

- Computing environment identified as available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing.
- User environment identified as location, collection of nearby people, and social situation.
- Physical environment identified lighting and noise level.

This definition though specific and operational is not flexible and extensible. To model the dynamic interactions of users operating in an environment we need to have a definition that can be constantly updated and is flexible to suit the changes in the environment.

After evaluating various definitions it is evident that context is a very subjective and pervasive concept. It is hard to establish a definition that can be prescriptively used across all context aware information systems and applications. Present definitions of context given in the literature are often specific to the context-aware application and environment for which they have been defined. Due to this one to one correspondence between the definition of context and the operating environment it is difficult to use a definition from one system or application into another. It thus behooves to have specific definition of context that can be prescriptively used in the development of the application at a construction site. The definition for context should standardize the subjective concept
of context in the domain of the operating environment of the application, which, in our case is the construction site environment.

We define context as elements of the environment at the construction site that are useful in modeling the user's current situation. These elements are determined by the nature of the tasks the user expects to undertake. For example, some important elements of the user's operating environment are location, time and project name. The contextual information required to model situations at construction site can either be sensed from the environment or be can be established by accessing the construction information repository. For the ease of handling context elements they are categorized based on the way they are established. We have divided context into two categories:

- Raw context
- Derived context

Based on the way the elements of context are established we categorize them either as raw context or as derived context. Raw context refers to the information that is sensed directly from the environment using either external or internal sensors. In this case, external sensors refer to those sensors that are placed in the environment and communicate to the computing system. For example, temperature sensors that are placed in the environment communicate temperature information to the computing system. Internal sensors refer to sensors embedded in the portable devices or those, which can be attached to the portable devices.
The various elements that can be sensed from the environment and include:

- Identity of the user.
- Spatial information (Latitude, Longitude, Altitude)
- Temporal information
- Surroundings (People, objects, tools etc.)

The list above is not exhaustive; the elements of the raw context are dependent on the availability of the sensors embedded in the environment and on the portable device. The raw context forms the basis of creating a model of the operating environment of the user. However, to provide contextual information to the user using just the raw context information is not efficient for the following reasons:

- Information in the project databases is not indexed according to the raw context elements, which makes it difficult to search for retrieve information using raw context elements as the search parameter.
- Human-environment interaction very complex and cannot be modeled by the information that can presently sensed using the sensors. For example, in a certain group a person could be a leader but at the same place in a different group he could be team member. In each situation the information required by the person would be different which cannot be sensed by using the raw context elements. A more refined model of the environment is required

The second category of context addresses to the above problems and makes the context of definition flexible, extensible and specific. Derived context elements, as the name
suggests, are the context elements that can be derived from the raw contextual information. Derived context elements are established by processing the raw context information, which can be done either on the portable computer or remotely processing raw context on some other computer. In essence derived context provides a unique bridge between the project database and the raw context information. The bridge is unique in the sense that now each user has a set of raw and derived context which uniquely describe his environment and using that the information system can provide information specific and tailored to his needs. The novelty of the concept of derived context can be realized in a scenario where construction engineering is having a meeting with an electrician and a plumber at the same time. The raw context elements like location and time would be the same but by processing them along with their own identity the contextual information system provide them with relevant information.

We can model all the real world situations at a construction site using the elements of the raw context. The main objective of the context aware information system is to provide context relevant information rather than modeling the real world situation. Using the raw context definition for searching context relevant information is difficult since information relating to construction process is not indexed according to the elements of raw context. This led us to introduce a new type of context element defined as processed part of context, that is, derived context. Derived context refers to the processed elements of context that provide an easy link between the raw context and the indexes of the information repository for the constructions.
The next step in designing the context aware information system is to establish raw context. Some elements of raw context are easier to establish than others. For example the identity of a user can be established through a login mechanism, or every user could be provided individual equipment with a unique identification code. Similarly, the temporal information is easily obtained from the operating system installed on the computers. The identity of surrounding people in the situation can be established through a range of communication technologies such as Bluetooth [Bluetooth, Web Resource] and Rendezvous [Rendezvous, Web Resource] to identify surrounding people in the group.

Location information is the most challenging and difficult element of the context to establish, especially in a dynamic and ad-hoc environment like a construction site, but is also one of the most useful elements for establishing context. In the next chapter we discuss various sensor technologies that are available to sense the location of a computing device and this can be used to establish the location element of context.

2.3 Establishing Context

The increased availability of commercial, off-the-shelf sensing technologies is making it more viable to sense location information in a variety of environments. These sensing technologies enable context aware environments to interpret and begin to understand the spatial cues of its occupants. Since the focus of this research is on developing of software architecture for context aware information systems, we have used available localization system for establishing location.
As discussed above location is the most important and difficult contextual element to establish is location. There are three basic techniques used for localization. Localization is the process for determining location in a given environment. They are triangulation, scene analysis and proximity. Location systems employ them individually or in combination. The triangulation location sensing technique uses the geometric properties of triangles to compute object locations. The scene analysis location sensing technique uses features of a scene observed from vantage point to draw conclusions about the location of the observer or of objects in the scene. The proximity location sensing technique entails determining when an object is near a known location. The object’s presence is sensed using a physical phenomenon with limited range.

Due to dynamic and ad-hoc nature of construction sites there are constraints on the choice of the localization system that can be easily adapted to a construction site. The following are the important features that are required in the localization system at a construction site.

- Low cost of installation
- Low maintenance
- Adaptability to the ad-hoc and dynamic environment
- Low infrastructure

The following paragraphs discuss existing location-based system developed.
2.3.1 Global Positioning System (GPS)

Global Positioning System (GPS) is a wide area positioning system. The infrastructure consists of 24 satellites and 5 ground support stations spread across the globe. In GPS, each satellite transmits a unique code, a copy of which is created in real time in the user-set receiver by the internal electronics. The receiver then gradually time shifts its internal clock till it corresponds to the received code: an event called lock-on. Once locked-on to the satellite, the receiver can determine the exact timing of the received signal in reference to its internal clock. If that clock were perfectly synchronized with the satellite’s atomic clocks, the distance to each satellite could be determined by subtracting a known transmission time from calculated receive time. The main drawback of the system is that GPS signal are not available in any environment, where the line of contact between satellite and the receiver is blocked and the receiver takes quite a while for establishing its first lock-on.

2.3.2 Active Badge

The Active Badge [Want et al., 1992] location system was developed in early 90’s at the Olivetti Research Lab. The localization system consists of a cellular proximity system that uses diffuse infrared technology. System consists of badges that transmit IR-signals, sensors and servers. The badge emits a globally unique identifier every 10 seconds or on demand. The active badge is shown in Figure 1 below. A central server collects this data from fixed infrared sensors around the building, aggregates it, and provides an application-programming interface (API) for using the data.
As with any diffuse infrared system, Active Badges have difficulty in locations with fluorescent lighting or direct sunlight. These problems occur because of the spurious infrared emissions these light sources generate. In addition, diffuse infrared has an effective range of several meters, which limits cell sizes to small or medium-sized rooms. In larger rooms, the system must use multiple infrared beacons. As a result, the initial cost of set up is high and requires constant monitoring.

2.3.3 Active Bat

In more recent work, AT&T researchers have developed the Active Bat location system, which uses an ultrasound time-of-flight lateration technique to provide more accurate physical positioning than Active Badges [Ward et al., 1997]. Users and objects carry Active Bat tags shown in Figure 2. In response to a request the controller sends via short-range radio, a Bat emits an ultrasonic pulse to a grid of ceiling-mounted receivers. At the same time the controller sends the radio frequency request packet, it also sends a synchronized reset signal to the ceiling sensors using a wired serial network. Each ceiling
sensor measures the time interval from reset to ultrasonic pulse arrival and computes its distance from the Bat. The local controller then forwards the distance measurements to a central controller, which performs the lateration computation.

![Active Bat](image)

**Figure 3: Active Bat**

Statistical pruning eliminates erroneous sensor measurements caused by a ceiling sensor hearing a reflected ultrasound pulse instead of one that traveled along the direct path from the Bat to the sensor. The system, as reported in 1999, can locate Bats to within 9cm of their true position for 95 percent of the measurements, and work to improve the accuracy even further is in progress. It can also compute orientation information given predefined knowledge about the placement of Bats on the rigid form of an object and allowing for the ease with which ultrasound is obstructed. Each Bat has a global unique identification (GUID) for addressing and recognition. Using ultrasound time of flight this way requires a large fixed-sensor infrastructure throughout the ceiling and is rather sensitive to the precise placement of these sensors. Thus, scalability, ease of deployment, and cost are disadvantages of this approach.
2.3.4 Cricket

Complementing the Active Bat system, the Cricket Location Support System uses ultrasound emitters to create the infrastructure and embeds receivers in the object being located [Priyantha et al., 2000]. This approach forces the portable devices to perform all their own triangulation computations. Cricket uses the radio frequency signal not only for synchronization of the time measurement, but also to delineate the time region during which the receiver should consider the sounds it receives. The system can identify any ultrasound it hears after the end of the radio frequency packet as a reaction and ignore it. A randomized algorithm allows multiple uncoordinated beacons to coexist in the same space. Each beacon also transmits a string of data that describes the semantics of the areas it delineates using the short-range radio.

Like the Active Bat system, Cricket uses ultrasonic time-of-flight data and a radio frequency control signal, but this system does not require a grid of ceiling sensors with fixed locations because its mobile receivers perform the timing and computation functions. Cricket, in its currently implemented form, is much less precise than Active Bat in that it can accurately delineate 4x4 square-foot regions within a room, while Active Bat is accurate to 9cm. However, the fundamental limit of range-estimation accuracy used in Cricket should be no different than Active Bat, and future implementations may compete with each other on accuracy.

Cricket implements both the lateration and proximity techniques. Receiving multiple beacons lets receivers triangulate their position. Receiving only one beacon still provides useful proximity information when combined with the semantic string the beacon
transmits on the radio. Cricket's advantages include privacy and decentralized scalability, while its disadvantages include a lack of centralized management or monitoring and the computational burden (and consequently power burden) that timing and processing both the ultrasound pulses and RF data place on the mobile receivers.

2.3.5 RADAR

A Microsoft Research group has developed RADAR, a building-wide tracking system based on the IEEE 802.11 WaveLAN wireless networking technology [Bahl et al., 2000]. RADAR measures, at the base station, the signal strength and signal-to-noise ratio of signals that wireless devices send, and uses this data to compute the 2D position within a building. Microsoft has developed two RADAR implementations: one using scene analysis and the other using lateration.

The RADAR approach offers two advantages: It requires only a few base stations, and it uses the same infrastructure that provides the building's general purpose wireless networking. Likewise, RADAR suffers two disadvantages. First, the object it is tracking must support a wireless LAN, which may be impractical on small or power-constrained devices. Second, generalizing RADAR to multi floored buildings or three dimensions presents a nontrivial problem. RADAR's scene-analysis implementation can place objects to within about 3 meters of their actual position with 50 percent probability, while the signal strength lateration implementation has 4.3-meter accuracy at the same probability level. Although the scene-analysis version provides greater accuracy, significant changes in the environment, such as moving metal file cabinets or large groups of people
congregating in rooms or hallways, may necessitate reconstructing the predefined signal-
strength database or creating an entirely new database.

2.3.6 Smart Floor

In Georgia Tech's Smart Floor proximity location system [Orr et al., 2000], embedded
pressure sensors capture footfalls, and the system uses the data for position tracking and
pedestrian recognition. This unobtrusive direct physical contact system does not require
people to carry a device or wear a tag. However, the system has the disadvantages of poor
scalability and high incremental cost because the floor of each building in which Smart
Floor is deployed must be physically altered to install the pressure sensor grids.

2.3.7 Electromagnetic Sensing

Finally electromagnetic sensing also offers a position-tracking method [Raab et al.,
1999]. The large body of research and products that support virtual reality and motion
capture for computer animation often offers modern incarnations of this technology.
These tracking systems generate axial DC magnetic-field pulses from a transmitting
antenna in a fixed location. The system computes the position and orientation of the
receiving antennas by measuring the response in three orthogonal axes to the transmitted
field pulse, combined with the constant effect of the earth's magnetic field. Disadvantages
include steep implementation costs and the need to tether the tracked object to a control
unit. Further, the sensors must remain within 1 to 3 meters of the transmitter, and
accuracy degrades with the presence of metallic objects in the environment.
In this chapter we discussed the concept of context and context awareness. We have looked at various definitions of context to develop a specific and operational definition of context for the unique construction site environment. For establishing location information at a construction site we have looked at various localization systems. We have identified the important requirements for a localization system at the construction site. In the chapter 4 we would identify the localization system of our choice based in these requirements that must be satisfied. In the next chapter we would discuss the requirements on the design of the software architecture for the context aware application for the construction site.
Chapter 3

Software Architecture for Context Aware Systems

Software architecture provides guiding principles for design and development of any software application. The software architecture of an application is the structure, which comprises software components, the externally visible properties of those components, and the relationships among them. The externally visible properties of the components refer to those assumptions other components can make of a component, such as its provided services, performance characteristics and fault handling. Software architecture provides a high-level abstraction that helps in viewing the application without going into the implementation details of the components. It identifies the needs of and from the system and provides a road map for the developers to implement functionalities required by the system. Various software components that are used in development of software architecture are: libraries, frameworks, toolkits or infrastructures. A library is a generalized set of related algorithms for performing certain tasks like manipulating strings and performing complex mathematical calculations. Libraries focus exclusively on code reuse. Frameworks concentrate more on design reuse by providing a basic structure for certain class of application. Toolkits build on framework by offering a large
number of reusable components for common functionality. An infrastructure is a well-established set of technologies that act as a foundation for other systems or applications. Software components generally consist of either of these components individually or in combination with each other.

The present lack of guidelines for handling context and contextual information poses a challenge for the development of context aware applications and information systems. The study of present context aware applications shows a lack of generality in their software architecture. The lack of generality is inherent from the way each application uniquely defines and processes context and contextual information. Uniqueness of the construction environment due to the dynamic and ad hoc nature makes it important to develop a software architecture that takes into consideration the unique features of the construction site. To develop information system for construction sites it is therefore important to design a architecture for the information system. It will help in viewing the various requirements of the system. In the next section we identify particular requirements of the software architecture for developing context aware information system at a construction site.

3.1 Requirements for Prophet Architecture

There are certain basic requirements that should be addressed while designing software architecture for context aware applications or systems. The software architecture should provide components for:
• Establishing context for the entities and events. It should provide a component to interface with external or internal sensors to establish the contextual information that can be sensed from the environment. External sensors refer to sensors that can be attached to the portable devices to monitor information like location and temperature. Internal sensors are sensors already in the portable devices like clocks to monitor time.

• Managing of context. Managing of context refers to the ability to convert contextual information in different formats to a standard format and to deduce new contextual information by analyzing already established contextual information. The component manages the compiling of final context after taking inputs from all raw context sources and processed information in form of derived context.

• Cataloguing of various services based on the relevant context. The module should have the directory of services that are available in the system to aid the user. The services should be listed in a way that it is easy to search for relevant services based on established context of the user.

• Listing of relevant services as processed by the system based on the user’s context. The module should be able to list all the services that have been identified by the system based on the user’s context. The module should also provide for displaying all the information that is relevant to the user in a suitable manner.

Prophet is designed for construction sites so the system should perform well under the constraints imposed by construction site environment. Besides fulfilling the basic requirements of a context aware application the nature of the construction work places
further restrictions on the software architecture. As we have discussed in the previous chapter the ad-hoc nature of the construction site poses a limitation on the type of sensor network used to establish context. The architecture for the information system should provide for the structure where the portable device is capable of establishing context without relying heavily on external sensor network to provide the context information. The architecture should provide support in form of context services, which can operate on least input of raw context to provide refined contextual information regarding the user.

Construction involves interaction between different trades, each trade supports different protocol for sharing information. The information system should be able to accommodate the need for handling diverse formats of data. The architecture should also address the issue of low processing power and low storage of portable devices. The system should be able to assist the portable computers through network storage and processing as and when required. This requirement of providing network storage and processing leads to the issue of tradeoff, between having a smart portable device with large storage/processing power and limited network interaction or thin portable device with low storage/processing power, heavily dependent on network support. In either case the architecture should provide structure for graceful degradation of service quality in case of loss of network support.

In developing architecture for our context aware information system we looked at different architectures proposed by researchers. In the next section we will look at important features of these software architectures.
3.2 Related Software Architecture

There have been several context-aware applications built in recent years. Most of these software architectures seem to be based on the general architectural framework for context aware application as discussed in the previous section. They differ from each other in the way each component is implemented, individually or in conjunction with other software components, to suit their unique operating environment.

3.2.1 Schilit’s System Architecture

In his Ph.D. thesis, Schilit [Schilit, 1995] presented a software architecture that supported context-aware computing. This architecture was the first to support context-aware computing.

The software architecture consisted of three components:

- **Device agents**: refers to components that store contextual information about devices, like printers, involved in the situation.
- **User agents**: refers to components that store contextual information about user and other people involved in the situation.
- **Active Maps**: refers to components that keep track of the spatial information of both the devices and the users involved in the situation.

The system represents the real world in two forms: users and devices. Devices are tools that can be used to get the task at hand done, for example, printers and phones. The
context information, like location, for the devices is stored in active maps. Active maps support queries over location and context for finding devices.

The architecture does not support or provide guidelines for the acquisition of context from external or internal sensors. Instead, device and user agents are built on an individual basis tailored to the set of sensors that each used. Unlike the requirement of the Prophet architecture, Schilit's architecture does not provide any support through context service that process raw contextual data to provide derived context information. In addition, this architecture does not provide guidelines for managing and handling of context, leaving developers to implement these features.

3.2.2 Stick-e Notes Architecture

The Stick-e Notes system is a general framework for supporting context-aware applications [Brown, 1996]. The Stick-e Notes application implements a system where the user can attach notes to different context elements and later when context occurs the note is executed. The context could be time, location, user's identity etc. It could also be a combination of such context elements. The note could consist of text, html pages, sound files, programs to be executed or different user interfaces.

The software architecture consists of three components:

- Triggering component: refers to the catalogue component that searches for the list of available Stick-e Notes based on the established context.
• Execution component: it is the component to list and execute services and programs stored in the Stick-e Notes.

• Sensor component: it is the component for interfacing with the sensors and for periodically feeding information to the triggering component.

The architecture provides a general mechanism for indicating what context an application designer wants to use and provides simple semantics for writing rules, or Stick-e Notes, that specify what actions to take when a particular combination of context is realized. A group of notes or rules are collected together to form a Stick-e document. A context-aware application consists of the document and the Stick-e note architecture.

3.2.3 CyberDesk

The CyberDesk project [Dey, 1998] is aimed at providing a software architecture that dynamically integrates software modules in a system, for example in a desktop computer. This integration is driven by a user's context, where context includes the user's physical, social, emotional, and mental (focus-of-attention) environments. The architecture is built to automatically integrate Web-based services based on virtual context, or context derived from the electronic world. The virtual context is the personal information the user is interacting with on-screen including e-mail addresses, mailing addresses, dates and names. The virtual context is used to infer what activity the user was involved in, in order to display a list of relevant Web-based services. Although it is limited in the types of context it can handle, CyberDesk contains many of the mechanisms that are necessary for a general context-aware architecture. Applications specify what context types they are interested in, and are notified when those context types are available. The modular or
component based architecture supports automatic interpretation—that is, automatically interpreting individual and multiple pieces of context to produce an entirely new set of derived context. The architecture also supports the abstraction of context information and aggregation/combination of context information.

3.2.4 Context Toolkit

Context Toolkit is a software architecture developed by Dey et al. [Dey et al., 2001] to aid the development of context aware applications. The architecture uses an object-oriented approach and has three types of components:

- Context widgets
- Servers
- Interpreters

A context widget is a software component that provides applications with access to context information from their operating environment. They insulate applications from context acquisitions concerns. By hiding the complexity of the sensors used from the applications and by abstracting context information to suit the expected needs of applications. Context widgets are designed as reusable and customizable building blocks of context sensing and are defined their attributes and callbacks. Attributes of a context widget are pieces of context that it makes available to other components via polling or subscribing. Callbacks represent the types of events that the widget can use to notify subscribing components. Other components can query the widget’s attributes and callbacks. The widget also allows other components to retrieve historical context
information. Context servers are used to collect the entire context about a particular entity, such as a person. The context server is responsible for subscribing to every widget of interest and act as a proxy to application. It can be seen as a compound widget. Context interpreters are responsible for the interpretation of context information. They can transform between different representation formats or combining different information and create new information out of that.

3.2.5 Blackboard Architecture

Blackboard architecture provides the basic framework for developing context aware applications. The blackboard-based architecture adopts a data-centric rather than process centric point for the software design of context aware systems [Winograd, 2001]. It means that, rather than sending requests to distributed components and getting callbacks from them, a process posts message to a common shared message board, and can subscribe to receive messages matching a specified pattern that have been posted. In blackboard architecture, all communications go through a centralized server.

The overall architecture is a blackboard with two levels of data:

- Event heap
- Context memory.

The Event Heap, which uses T Space [Wyckoff et al., 1998], provides fast distribution of simple events tuples. A tuple is a well-formed piece of information. Any process can post tuples, which include fields for their source and timestamp, along with explicit data
associated with the event type. Any process can subscribe to a pattern of field values and receive callbacks when an appropriate tuple is posted. The second-level blackboard is a database that allows any process to store and retrieve relevant contextual data. This is used for data, which is relevant across applications and sessions such as physical objects and their locations in space. Queries are sent to Context Memory as strings or by posting an event to the Event Heap with the query string as one of the fields. In addition to responses that return data, the Memory has a template mechanism that allows users to post new data in Event Heap.

3.2.6 Service Infrastructure Based Architecture

An infrastructure is a well-established, pervasive, reliable and publicly accessible set of technologies that act as a foundation for other systems [Hong et al., 2001]. The service infrastructure refers to middleware technologies that can be accessed through the network by clients and the server. Any kind of device or application can use these services by adhering to predefined data formats and network protocols. The service-based architecture is based on the client-service architecture. In a service-based architecture, a client needs to find the location of a service and then set up a connection with it.

Acquisition and processing of context is done by the infrastructure through a set of services that can be accessed by any device and any application. An application that wants to use a particular service can either have a direct address or can run a discovery process using the description of a desired service.
A key feature of a service-based architecture is the independence of services. There is no manager to keep global track of services and their connections. Each service contains appropriate code to create connections and to marshal outgoing and incoming messages. This adds complexity to each component but this makes them more independent. The benefit of a service infrastructure is that this it can be used independently of hardware platform, operating system and programming language and the infrastructure can support a greater range of device and applications. This makes the infrastructure easier to evolve as new sensors, devices, operating systems and programming languages appear.

3.2.7 CoolTown

CoolTown is a web-based infrastructure that supports context-aware applications by representing people, places and things, with a Web page [Caswell et al., 2000]. Each Web page dynamically updates itself as it gathers new information about the object that it represents. As a result, CoolTown is primarily aimed at supporting applications that display context and services to end-users. For example, as a user moves throughout a physical environment, they will see a list of available services for this environment. They can request additional information or can execute one of the services. The CoolTown infrastructure provides abstraction components (URLs for sensed information and Web pages for entities) and a discovery mechanism to make the building of these types of applications easier. However, it is not designed to support interpretation of low-level sensed information. Although it focuses on displaying context, it is not intended to support other context-aware features of automatically executing a service based on context or tagging captured information with context. The infrastructure consists for the
application consists of: Web presence infrastructure, infrastructure for things (ChaiServer and eSquirt) and infrastructure of places (PlaceManager).

3.2.8 Context And Location Aware Information Service (CALAIS)

CALAIS is another architecture that is designed to support context-aware applications [Nelson, 1998]. This architecture is designed to solve two problems in context aware applications: the ad hoc nature of sensor use and the lack of a fine-grained location information management system. An abstraction was developed to hide the details of sensors from context-aware applications. However, similar to Schilit’s architecture [Schilit, 1995], there was very little support to aid developers in adding new sensors to the architecture and the architecture did not support storage of context or interpretation of context, leaving application developers to provide their own mechanisms on an individual basis. CALAIS supports the use of distributed context sensing and provided query and notification mechanisms. An interesting feature in this work was the use of composite events, being able to subscribe to a combination of events. For example, an application could request to be notified when Event B occurred after Event A with no intervening events. This is a powerful mechanism that makes the acquisition and analysis of context easier for application developers.

3.2.9 Context Information Service (CIS)

The CIS is another proposed architecture for supporting context-aware applications [Pascoe, 1998]. It has yet to be implemented at any level but contains some interesting features. It supports the interpretation of context and the selection of a sensor to provide
context information based on a quality of service guarantee. The CIS maintains an object-oriented model of the world where each real-world object is represented by an object that has a set of predefined states. Objects can be linked to each other through relations such as "close to." For example, the set of nearby printers would be specified by a "close to" relation with a user, a given range, and "printers" as the candidate object. The set would be dynamically updated as the user moves through an environment.

3.2.10 Situated Computing Service

The Situated Computing Service is a proposed architecture that is similar to CyberDesk for supporting context-aware applications [Hull et al., 1997]. It insulates applications from sensors used to acquire context. A Situated Computing Service is a single server that is responsible for both context acquisition and abstraction. It provides both querying and notification mechanisms for accessing relevant information.

In this chapter we discussed the requirements for designing software architecture for context aware applications and systems. Besides the general requirements for context aware applications and system, we identified the requirements imposed by the construction site environment on the design of the software architecture for context aware information system for construction sites. We looked at various context aware applications and systems for developing the software architecture for Prophet. Though the general requirements of Prophet Architecture are similar to other context aware applications, the Prophet architecture presents a different design for the software architecture to suit the requirements at the construction site. All of the software
architectures discussed above have been designed for application that operate in building or places that have fixed environment unlike the construction site, which is a dynamic and ad hoc work environment. As a result none of the architectures have the features required in the architecture of a context aware application for construction sites. We have used some ideas from the architectures, presented above, in our own design. In the next chapter we discuss about the Prophet software architecture developed for the construction site environment.
Chapter 4

Prophet: Software Architecture

Construction sites are information intensive environments. On-site decisions require large amount of design and archival data and knowledge to make rapid on-site decisions, and the lack of access to these resources at the job site often causes these decisions to be deferred. Traditionally, accessing this information at the point of work, where and when it will be most effective, has been difficult or impossible. In addition, the complex nature of the construction process and the dynamic nature of the construction sites can lead construction personnel to overlook important issues that may require quick response. The increasing power and decreasing cost of portable computing devices makes them appealing for construction applications but their limitations in storage capacity and user interface presents a challenge in accessing, synthesizing and modifying the large amounts of data associated with modern construction projects. To address these issues, we have built Prophet, a contextual information system designed to deliver up-to-date project information at the construction site. This system helps the users to manage the complexity of the construction data by proactively tracking current resource requirements and proactively obtaining access to context-relevant information and services.
This chapter describes the software architecture for Prophet, a context-aware system or architecture utilizing portable computers for on-site delivery of knowledge and data. Prophet’s architecture has been designed to intelligently link on-site construction personnel to the entire project information repository. The challenge for such a system lies in the complexity of defining, capturing, representing and processing contextual data. The next section describes various steps in development of the system starting with handling of context to the final software architecture of the system.

4.1 Definition of Context

Context is an intuitive and a subjective concept, difficult to articulate in the form of a definition. As discussed in Chapter 2, there have been various definitions given for context in various context aware applications. The definitions are used to model the operating environment of their users. Due to the uniqueness of the construction site environment we have developed a specific and operational approach to handle context for the development of Prophet. The approach consists of identifying elements of the environment that affect the current situation of the user and then classifying those elements based on the way they are established in the information system.

Context is defined as elements of the environment of the user at the construction site that are useful to model user's current situation, to help him in executing his tasks. Some important elements of the user's operating environment are location, time and project name. The contextual information required to model situations at construction site can either be sensed from the environment or be can be established by accessing the
construction information repository. For the ease of handling context elements they are
categorized based on the way they are established. As discussed in Chapter 2, we have
divided the contextual elements into two categories.

- Raw Context
- Derived Context

Raw context refers to those elements of context that can be easily sensed from the
environment with the help of the sensors attached to or in the portable devices. In Prophet
we categorized the following context elements as raw context elements.

- Identity of the user.
- Spatial information (Latitude, Longitude, Altitude)
- Temporal information
- Surroundings (People, objects, tools etc.)

Derived context refers to contextual information that can be derived from processing the
raw contextual information along with the project data. Some of the derived context
elements include project name, building name and weather conditions. Most of the data
in the construction project information repository are indexed based on the project name,
building name and the stage of the construction project. The derived context provides a
better link between the established context and the indexes of the information repository.
The processing of the raw contextual elements to generate corresponding derived context
elements is done by various services through interaction with the project information. The software architecture enables these services to run on either the portable devices or to run on a remote server that is accessed through the communication network available on-site. The next step after defining context is to establish context. In the next section we discuss about various ways to establish context.

4.2 Establishing Raw Context

Establishing raw context is the first step in context-aware computing. Some elements of context are easier to establish than others. For example, the identity of a user can be established through a login mechanism, or every user could be provided individual equipment with a unique identification code. The temporal information is easily obtained from the operating system installed on the portable computers. Finally, the handheld computer can use a range of communication technologies like Bluetooth and Rendezvous to identify other members of the community by identifying and recognizing neighboring equipments in a given space.

Location information is the most challenging and difficult element of the context to establish, especially in a dynamic and ad-hoc environment like a construction site, but is one of the most useful elements for establishing context. The increased availability of commercial, off-the-shelf sensing technologies is making it more viable to sense location information in a variety of environments. These sensing technologies enable context aware environments to interpret and begin to understand the spatial cues of its occupants. As the focus of the research work is on development of software architecture for context
aware information system, we have used available localization system for establishing location.

As discussed earlier in Chapter 3 the most important and difficult contextual element to establish is location. There are three basic techniques used for localization. They are triangulation, scene analysis and proximity. Location systems employ them individually or in combination. The triangulation location sensing technique uses the geometric properties of triangles to compute object locations. The scene analysis location sensing technique uses features of a scene observed from vantage point to draw conclusions about the location of the observer or of objects in the scene. The proximity location sensing technique entails determining when an object is near a known location. The object’s presence is sensed using a physical phenomenon with limited range.

Due to dynamic and ad-hoc nature of construction sites there are limitations on the choice of the localization system. The following are the important features that are required in the localization system at a construction site.

- Low cost of installation
- Low maintenance
- Adaptability to the ad-hoc and dynamic environment
- Low infrastructure

While the Global Positioning System (GPS) has been the localization technology of choice for outdoor environments, indoor localization is a much more difficult problem. Indoor localization is an important issue for building construction because a large portion
of construction activities occurs in areas that are unable to receive GPS signals. In our research, we are using the RADAR system [Bahl et al., 2000] developed at Microsoft to provide indoor localization. RADAR is an indoor tracking system and it is built using an IEEE 802.11 standard-based RF wireless local area network (WLAN) deployed at the construction site. The fundamental idea used for localization in RADAR is that in an RF network (wireless LAN) the energy level or the signal strength of a packet is a function of the user’s location. Based on the signal strength database or a RF propagation model the system is able to determine the user’s location. The main advantage of the system is that it requires few base stations and it uses the same wireless network infrastructure that could be used at the construction site for communication. The following section describes the software architecture of the system.

4.3 Prophet Software Architecture

Most of the contextual information system architectures, like context toolkit [Dey et al., 2001], infrastructure-centered architecture [Hong & Landay, 2001] and blackboard architecture [Winograd, 2001] are designed to support applications that are part of fixed environments, where the only dynamic component is the position of the user. These environments are well defined, have a fixed set of resources and a well-defined work process. Construction applications, on the other hand, have limited and changing infrastructure, must deal with changes to the physical layout of the site and must be able to adapt to each new trade and discipline as they become involved in the project. To enable context-aware applications to function in this environment, we have designed the Prophet software architecture to address the requirements of the construction site. This
architecture is designed to be flexible, robust and extensible in order to cope gracefully with the dynamic nature of the construction process. The hardware infrastructure used in Prophet consists of a wireless network, a collection of server machines and "client" applications running on the portable device. The outdoor localization is handled using a GPS receiver attached to the portable device and indoor localization is implemented through a modified RADAR approach. The schematic diagram of the architecture is shown in the following Figure 3.

Both the server and the client host software modules that interact with each other to ensure delivery of contextual project information from the servers to the portable devices. The various software modules interact with each other through a Standard Descriptive Language (SDL) or the communication protocol provided in the architecture. SDL is provided in Prophet to ease interaction among modules in the clients or on the server side. SDL is a structured and standard way sharing information between modules. Over the network the modules interact through the provided communication protocol.

Figure 4: Prophet Software Architecture
The following paragraphs describe how each module functions in the whole system.

4.3.1 Services Module

The Prophet information architecture uses local and network services to provide contextual information to construction personnel. The services are divided into three categories based on the functionality they provide:

- Context services
- Data services
- Processing services

A context service is defined as a service whose results modify the current context for an application. An example of a context service would be one that takes the current location of the user and determines the floor of a building that the user is currently on. The SDL description of a context service includes service name, location of the service, the elements of the context that are needed to execute the service and the elements of the context that are modified by the result of this service. The context services can add a new element to the context, remove and modify elements of the context. The service that identifies the building floor the user is on would add context element floor-id when the user enters the building, modifies it as the user moves from floor to floor and removes it once the user has left the building.

A data service is defined as a service that can search and retrieve a specific set of documents from the server. An example of a data service would be a service that uploads
design drawings of the construction site based on the user’s position on that site. The SDL for a data service includes the name of the service, location of the service, the elements of the context that are needed to execute the service and a set of intervals for the element of the context for which the resulting document is valid. The output of each service would be a set of documents and a range defining what changes to context will render these documents non-relevant. For example, each CAD drawing retrieved based on a position has a range that defines that, as long as position remains within the specified limits, this service does not need to re-run.

A processing service is defined as a service that does some computation on behalf of the user. These are intended to be services that allow the user to perform quick, back-of-the-envelope calculations, though the architecture does not place any restrictions of the type of calculations produced by the services. The SDL for a processing service includes the name of the service, the location of the service, the context in which the service is applicable and a set of input and output fields that define the values needed and generated by the service.

The description of all the services available in the system is stored in the registry module and is represented using SDL. Each service uses the SDL to define its requirement for inputs, its location and the output generated as the result of its processing. The next section describes the structure of the registry that stores and processes the SDL descriptions of services.
4.3.2 Registry Module

The registry module implements a catalogue service listing all services available to the users of the Prophet architecture. The capabilities and requirements of each service are described using the service description language (SDL). To reduce the communication requirement and to help deal with the interruptions in the wireless connectivity, copies of registry are located both on the handheld device and on the main server. The registry lists all services available to the user regardless of their location. The registry allows the services to be added and removed at any time and provides a mechanism for context-based search of data, context and processing services.

4.3.3 Sensor Module

The sensor module provides the mechanism for obtaining the raw context. The sensor module includes a set of interfaces that are used to query the sensors on or connected to the portable device. The sensor module provides an interface that allows it to populate the context module with the information that defines the raw context. Some of the elements of raw context that could be obtained from widely available sensors include: time, position, orientation and availability and quality of the Internet connection. It is important to note that the notion of sensors, as considered by the sensor module, includes not only sensors able to monitor the environment external to the handheld device but also the sensors that are able to monitor the state of the handheld device itself. The raw context elements that could be obtained by monitoring the portable device include the identity of the user, the battery level of the device, the current focus of the user and the existence or lack of additional hardware such as digital cameras. The sensor module interface
provides a mechanism for querying the sensors for raw context. The interface provides a
mechanism for controlling both what sensors will be polled and the rate at which they
will be queried.

4.3.4 Context Module

The context module is responsible for building the derived context. After it receives the
raw context from the sensor module its objective is to refine the context using the
available context services. Based on the network availability, the context module scans
the registry, either locally or remotely on server, to find all available context services.
Context services are local or network services, which are able to further refine the current
state of the context. For example a context service would take in the values of latitude
and longitude to provide the project name associated with that location. A detailed
description of these services is given in service module section. The context module then
iterates invoking available context services until it detects that the iteration did not
produce any changes to the context. This process is iterative because of the dependencies
between context services. For example, a service that finds the floor-id of the user in a
building is dependent on the service that generates building id based on the location of
the user. As a result, every iteration is expected to add information to the current context.

To reduce the processing load on the handheld device and the server, the output of each
context service is qualified with a set of fields that define the range during which this
output will remain valid. The output of each service and the qualifying range are stored in
form of a tuple in a context cache, which is located locally on the handheld device.
4.3.5 User Interface Module

The user interface module is responsible for listing and invoking of all the relevant services that are available to the user based on its context. The user interface module uses the current context to scan the registry and to identify all services relevant to the current context. As a result, the user can be presented with the list of services that are relevant to the current context. The user interface can also identify all services available to the user and highlight the ones that are relevant to the current context. To further reduce the computing burden, the Prophet system uses a data cache to store recently executed requests. As a result, before contacting the data or processing service, this module checks the local cache manager to determine if the results of this request have been cached there. If managed properly, the cache module can produce significant reduction in the processing and communication requirements of web services.

4.3.6 Cache Module

The cache module on the client is divided into two parts: cache and the cache manager. The cache is a repository for files retrieved from data services and a database storing the results of processing services. The cache manager indexes the information about the data in the cache. As a result, before contacting the data or processing service, context and user interface modules check the local cache manager to determine if the results of their request have been cached there. If managed properly, the cache module can produce significant reduction in the processing and communication requirements of web services.
In the next chapter we will talk about the implementation of the Prophet Software architecture, done as a part of this research project.
Chapter 5

Prophet: Software Architecture Implementation

We have implemented the components of the Prophet architecture and built a client application that uses this architecture to pro-actively deliver project information. The implementation uses an iPAQ 3850 handheld computers with a GPS receiver and a wireless Ethernet card as the client hardware platform [Figure 4]. The client software is built using a combination of Embedded Visual Studio and Visual C # and uses the Microsoft .NET architecture to support remote service interaction.

Figure 5: iPAQ 3850, GPS and Wireless Card
The client software uses AutoCAD OnSite View to view CAD drawings, Pocket Word to view Word Documents and custom viewers are provided for the remaining types of data received from the services. The choice of using an iPAQ was based on the availability of software development kits and APIs, but all implemented code will run on any handheld device that run the Pocket PC operating system.

The server side implementation of Prophet uses Microsoft Internet Information Service (IIS) as the server platform. The services are implemented using Microsoft .NET platform and Microsoft Soap Toolkit 3. The project repository, at this time, contains a set of CAD drawings, a set of Microsoft Word documents and a construction schedule modeled using Primavera Project Planner. The current set of services allows the user to query and modify the project repository.

The client application built for the implementation of Prophet Architecture consists of the following software modules as discussed in the previous chapter:

- Sensor Module
- Registry Module
- Context Module
- Cache Manager Module
- Service Module

In the following section we will discuss the implementation of each of the module in our application. The modules exchange information in form of XML (eXtensible Markup Language) tuples. A tuple is defined as a structured list of values and an XML-tuple is a
list of data fields expressed in XML. The list can have any number of fields. A data field or field is simply XML-tagged data. XML provides several benefits as an encoding mechanism for sharing information and functionality between disparate systems. Because XML is essentially text based, it can integrate easily with existing application. In addition, the syntax can be easily read, understood and shared by developers. The communication between the clients, the network services and the main server is implemented using Simple Object Access Protocol (SOAP). SOAP is the new industry standard for communication protocol.

5.1 Sensor Module

The sensor module interacts with various sensors inside or attached to the handheld device. In our implementation we have two sensors attached onto the handheld device. These sensors are:

- GPS receiver
- Wireless Ethernet card

Both of these sensors are used for sensing location information of the user. The sensor module polls the GPS to provide the location information of the handheld device. GPS provides the information according to National Marine Electronic Organization (NMEA) standard and the sensor module parses the stream of data to get values of latitude, longitude and altitude. It formats the information in SDL as shown below:

- <context>
  
  <latitude>29.45</latitude>

  <longitude>90.29</longitude>
This information is shared with rest of the modules, which polls the sensor module for contextual information. In absence of the availability of GPS the sensor module polls the wireless 802.11 sensor to find the signal strength values, if available, and tries to calculate user's position using RADAR method. The RADAR method has a few practical problems. The signal strength reading received by the card is stored in a proprietary way in the handheld computers, often it is difficult to get the card company give this information voluntarily. It decreases the choice of wireless sensor cards that can be used in the system.

5.2 Registry Module

The registry module is implemented as a catalogue service listing all web services available to the users of the Prophet architecture. The registry lists all services available to the user regardless of their location. The registry allows the services to be added and removed at any time and provides a mechanism for context-based search of data, context and processing services. The capabilities and requirements of each service are described using the service description language (SDL). The following is an example of a typical registry entry for a data service:

- `<ServiceList>`
  - `<service>`
<id>service1</id>
<type>data generation</type>
<availability>network</availability>
<location>http://128.42.42.42/apps/apps.wsdl</location>
<-<context>
    <latitude>latitude</latitude>
    <longitude>longitude</longitude>
    <time>time</time>
</context>
<-<output>maps</output>
</service>
-  </ServiceList>

5.3 Context Module

The context module is responsible for building the derived context. After it receives the raw context from the sensor module its objective is to refine the context using the available context services. This module is implemented as tuple space [Ahuja et.al. 1986] with each tuple represented using XML. This approach to the concept of tuple spaces is referred to as XML-spaces. XML-tuples can be written to, and read from, XML-spaces. Writing is done simply by sending an XML-tuple to the XML-space. Reading is done by presenting the XML-space with a "template", which is an XML-tuple, whose tags and values are to be matched against XML-tuples already in the XML-space. If the template matches a stored tuple, a copy of that tuple is returned. The normal read operation returns
null if the template does not match anything that is stored in the space. Another read operation, called a scan, returns copies of all the tuples that match a template, instead of returning just the first one to match, as the normal reads do. In addition, XML-spaces can be used for communications buffers. For example, a client application can register with an XML-space, giving it a template to be matched, and an operation, such as write. If any other client application writes an XML-tuple that matches the template, the XML-space notifies the registered application, sending it a copy of the new XML-tuple. This approach has several significant advantages: the simple nature of the representation ensures the ease of implementation, and, since the senders and the receivers do not interact, XML-spaces are able to gracefully deal with client failure and evolution of the client set [Carrier et al. 1994].

Because of the requirement of the flexibility in implementation of our proposed architecture, XML-spaces provide significant advantages in representing and manipulating context. Equally important for use of this approach in handheld devices is the ease of implementation that ensures that the task of context matching will not overburden the processing and storage capabilities of the handheld computer. As a result, the context module is implemented as an XML-space and uses the XML-space operations to manage the context tuples generated by the context services running within Prophet.

5.4 Cache Manager Module

The cache manager is implemented as a tuple space with each tuple defining the input fields and the service used to generate each piece of data in the cache. Just as in the case of the context module, each tuple has a qualifying range defining the range for which this
data remains valid. For example the AutoCAD file for a plan of a building would be retrieved by a single position but would be stored in the cache as long as the user’s position does not extend outside the four corners of the drawing. Cache manager is also implemented as a XML tuple space with each tuple in the tuple space serving as a XML representation for data stored in the cache and the period during which it is valid.

5.5 Service Module

The Prophet architecture uses local and network services to provide information to construction personnel. The network services are implemented in form of web services. A Web Service [Figure 5] is a programmable application accessible using standard

![Diagram of Web Service]

Figure 6: Schematics of Web Service

Internet protocols, in this case SOAP. Web Services combine the aspects of component-based development and the HTTP protocol. Like software components architecture, Web Services model the underlying functionality of the services as a black box that can be assessed and reused without worrying about how the underlying service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-
specific protocols, such as Distributed COM (DCOM), Remote Method Invocation (RMI), or Internet Inter-ORB Protocol (IIOP). Instead, Web Services are accessed through HTTP and use XML to represent data exchanged between services. The capabilities and inputs of web services are described using Web Service Description Language (WSDL). WSDL is an XML format for describing network services as a set of endpoints operating on SOAP messages containing either document-oriented or procedure-oriented information. WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, we have used WSDL in conjunction with SOAP. The above figure 5 gives a schematic for web service architecture.

The contextual information generated and collected by the Prophet is presented to the user through the user interface module. The user interface module consists of available commercial software to display the information.

Figure 7: User Interface
Chapter 6

Conclusion and Future Work

This thesis describes the Prophet Software architecture for context-aware information systems. The main contribution of this architecture is the use of context to proactively identify and retrieve information and services that would benefit the user as they move around a construction site. The architecture is designed on a client-server framework. Client is designed keeping in consideration the low processing and storage capacity of the portable devices. The architecture is flexible, robust and extensible in order to cope gracefully with the dynamic nature of the construction process.

The Prophet architecture was designed to deal with dynamic changes to the set of services, the context of the user, the communication infrastructure and changes to the data from the sensors that are available to the user. The architecture allows services to be distributed across network and allows graceful degradation in situations where the network connectivity is temporarily unavailable. Finally, the architecture has been designed to run off low-cost, off-the-shelf hardware and uses commonly available technology that reduces the cost of building and deploying this architecture. The system design also reduces the level of computer sophistication required from the users of this architecture.
Future Work

The Prophet architecture provides a framework for building an information system that aids construction workers in managing their resources intelligently and efficiently. While the initial implementation shows significant benefits of this approach, we have also identified a number of issues for future research.

One of the major issues is the ability to adapt to variations in the bandwidth of the available communication. At a construction site it is difficult to maintain a strong network connection due to interferences from the environment. In condition of weak network connection, communicating information from the server to the client can take indefinitely long period and even weaken the access of other user in the system. The solution for this problem is to design the network to provide strong connection at all location of construction sites. Even with a better design of network, communication is hindered by interferences at the construction sites. A more practical solution is to change the data files being send to the user according to the connection strength between the user and the system. The strength of the connection between the user and the system is measured by the available bandwidth of communication between them. At high bandwidth the user will get access to full data files but at low bandwidth the system will edit the data files and send only the important feature of that file. The ability to filter data files based on the bandwidth and retrieving and communicating only a sub-set of information available in those files would make this system more responsive.
There are also a number of issues that need to be addressed in improving the ability of the localization to handle the specifics of the construction site. We are currently looking at how knowledge about the construction site, and its evolution, and domain-specific characteristics can be applied to RADAR to improve its accuracy and reduce the effort involved in setting up its network infrastructure. Another interesting method for localization would be to attach a digital camera on the portable devices and do a frame-by-frame analysis of images taken by the camera. Finally, there are a number of approaches that can be implemented to support the building and distribution of context-aware services such as sending location based services.
References


• Web Resource: http://www.bluetooth.com/