

https://dspace.org/

# Designing a usable mobile application for field data collection

**Moe, Kyaw Hlwan**

**2006-03-13**
http://hdl.handle.net/10539/249

Downloaded from a DSpace repository. https://dspace.org/

# DESIGNING A USABLE MOBILE APPLICATION FOR FIELD DATA COLLECTION

**Kyaw Hlwan Moe**

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfillment of the requirements of the degree of Master of Science in Engineering.

Johannesburg 2004

# DECLARATION

This project is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg.

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been previously submitted for any degree or examination in any other university.

_____

(Signature of candidate)

_____ day of _____ 200 ___

# ABSTRACT

The advent of mobile technology, Geospatial Information Systems (GIS) and convergence of voice and data over wireless networks have led to an explosion of a wide range of mobile applications. These applications include mobile internet browsers, handheld GPS navigation systems, Location Based Services (LBS), mobile workforce management systems, and so on. While much of the underlying technology is already available, there are challenges with respect to the usability of mobile applications.

This project investigates the usability of a mobile application for field data collection in a utility industry. The purpose of the investigation is to gain a better understanding of the usability requirements for a mobile field data collection application but more importantly, how to meet these requirements using appropriate usability engineering techniques. A usage-centered design approach is used to design the user interface for the field data collection application. During this model-driven design process, the usability requirements are analyzed in terms of the user requirements, field data collection tasks and the operational context of fieldwork. An Underground Utility Closure (UUC) data sourcing work employed at a telecommunications utility is used as a case study for the field data collection work. The user interface is implemented as a functional prototype on a pocket computer and evaluated for usability in a field setting. It is envisaged that the usability requirements and design guidelines presented in this project will enable software engineers to meet the design challenges of usable mobile applications for field data collection and mobile computing in general.

*In memory of my grandfather*

*U Ba Khin*

# ACKNOWLEDGEMENTS

Prof. Barry Dwolatzky, for his supervision, advice and support throughout the project.

Prof. Rex van Olst, for organizing field trips and meetings with relevant people from Telkom and Intergraph South Africa.

Eskom, for funding this research.

The operations manager and data sourcing personnel from the Telkom offices at Horizon, Roodepoort for their cooperation with the field trips to data sourcing sites. In particular, Mr. Dean van Inn and Mr. Wimpie Putter for giving permission to study field data sourcing work at Telkom.

# TABLE OF CONTENTS

Page Number/
Document Number

Declaration ……..i

Abstract ……. ii

Dedication ……..iii

Acknowledgements ……..iv

Table of Contents ……..v

Foreword ……..vi

Introduction ……..1

MSc Paper – K. Moe ……..3

Conclusions ……..14

Developments and Recommendations…….....18

References and Bibliography ……..21

**Appendix A**

Literature Survey ……..Doc. No. 1

Software Requirements ….....Doc. No. 2

User Interface Design ……..Doc. No. 3

Software Design ……..Doc. No. 4

Usability Evaluation ……..Doc. No. 5

**Appendix B**

UUC data sourcing forms…….....Doc. No. 6

Usability test questionnaires & data logging forms.....Doc. No. 7

iPaq H3900 Specification Sheet…….....Doc. No. 8

# FOREWORD

The format of this Masters Dissertation differs from a standard dissertation format mainly because it consists of a short *body* and numerous appendices. The *body* consists of an introduction, a technical paper, a conclusion, a recommendations section and references. In essence, the substance of this dissertation is in the body of the report. The technical paper provides an overview of the project and highlights the most important findings of the research. Conclusions, recommendations and future work regarding this project are proposed in the conclusions and recommendation section. The appendices support the body of the dissertation by providing a detailed report of the research.

The appendices are divided into two groups, namely Appendix A and Appendix B. The former contains documents pertaining to the literature survey (Doc. No. 1), software requirements (Doc. No. 2), user interface design (Doc. No. 3), software design (Doc. No. 4) and usability evaluation (Doc. No. 5). The latter contains other relevant documents such as the data sourcing forms (Doc. No. 6), usability questionnaires (Doc. No. 7) and technical specifications for the target device (Doc. No. 8). Noticeably, each document has a unique numeric identifier (e.g. Doc. No. 1) that the report uses as a reference to a particular document.

# INTRODUCTION

The advent of mobile technology (e.g. handheld computer, mobile phones, and portable GPS receivers etc.), Geospatial Information Systems (GIS) and convergence of voice and data over wireless networks (GSM, GPRS, 3G, Wireless LAN, and Bluetooth etc.) have led to an explosion of a wide range of mobile applications. These applications include mobile internet browsers, Location Based Services (LBS), mobile multimedia, "real-time" field data collection for resource management, and so on. While much of the underlying technology is already available, there are challenges with respect to the usability of mobile applications. Mobile computing is fundamentally different from their desktop counterparts. The software designers must meet these challenges and capitalize on unique characteristics of mobile devices such as a small screen, limited input mechanisms, finite power supply and network dependency etc. Not only are there differences in the technology, but also the environment and situation in which mobile applications are used.

The project described in this dissertation involves an investigation into the design of usable mobile field data collection application. The purpose of the investigation is twofold. Firstly, to understand usability requirements of a mobile application for field data collection and secondly, to design software that meets those usability requirements. The investigation begins with the literature survey on a number of topics pertaining to a mobile computing, particularly the usability of mobile applications. This is followed by the design and testing of a user interface for the field data collection application. An Underground Utility Closure (UUC) data sourcing work employed at a telecommunications utility is used as a case study to gather software requirements. The user interface design for the field data collection application is implemented as a working prototype on a mobile device. With the help of data sourcing personnel the prototype is evaluated for usability in a field setting.

The requirements gathering and analysis process is described. The requirements elicitation techniques include interviews with the intended users, observation of their

work and the analysis of data sourcing forms. A model-driven usage-centered design process is used to analyze the software requirements by placing more emphasis on the tasks and the operational context of the users' work rather than the users themselves. A complete software design for the field data collection application is presented. It consists of two separate layers supporting the user interface and the application proper, respectively. The former is concerned primarily with the presentation of the user interface whereas the latter is responsible for the functionality of the field application. The user interface is implemented as a functional prototype on a Pocket PC operated Personal Digital Assistant (PDA) and the selection of the target platform is also discussed. An object-oriented software design process called ICONIX is used to design and illustrate the application architecture which is really a high-level overview of functionality and data structures of the field application. The usability evaluation of the prototype with the test results are then discussed.

# Designing a Usable Mobile Application for Field Data Collection

Kyaw H. Moe
School of Electrical and Information Engineering
University of the Witwatersrand

*Abstract*—**This paper investigates the design of a usable mobile application for field data collection work employed in a utility industry. The purpose of the investigation is to gain a better understanding of usability requirements for a mobile field data collection application but more importantly, how to meet these requirements from a usability engineering standpoint. A model- driven usage-centered design approach is adopted to design the user interface. The field data collection work used in this project is based on a case study of Underground Utility Closure (UUC) data sourcing work carried out at a telecommunications utility. The usability requirements for the mobile field application are analyzed in terms of the user requirements of UUC fieldworkers, the data sourcing tasks and the operational context of fieldwork. The user interface is designed to meet the usability requirements and its design features are discussed. The user interface is implemented as a functional prototype on a pocket computer and evaluated for usability in the field. Overall, it is felt that the prototype is able to meet the usability requirements and the design challenges of usable mobile field application.**

*Keywords:* **Field Data Collection, Mobile computing, Usability, User Interface Design**

## 1. Introduction

THE past few years have seen significant rise in the use of mobile applications ranging from common SMS and WAP services to the Location Based Services (LBS), GPS navigation system, mobile Geospatial Information Systems (GIS) systems, mobile workforce management systems etc. Although much progress has been made in technological innovations, mobile applications typically suffer from poor usability. Mobile computing is fundamentally different from the much studied desktop counterpart. Not only are there inherent differences in the technology, but also in the nature of user interaction, usage patterns and environment in which people use mobile applications. Unfortunately, well-established usability design guidelines and techniques for desktop applications are not always suitable for mobile use. Hence, appropriate usability guidelines, new paradigms for interaction styles and data presentation techniques are needed to meet the design challenges of usable mobile applications.

In this context, this paper presents the design of a usable mobile application for field data collection in the utility industry. The primary objective of the project is to investigate the usability requirements of a mobile field application and to meet these requirements through use of appropriate usability design techniques. A field study of data sourcing work employed at a telecommunication utility is used to gather usability requirements. A model-driven usage-centered design approach is used for requirements analysis and the subsequent design of the user interface. The user interface is implemented as a working prototype on a PDA. User satisfaction of the prototype is determined using an empirical usability test in a field setting.

The background information on mobile computing particularly that pertaining to the usability of mobile applications is provided in section II. Section III describes the deployment of mobile technology for field data collection in the public utilities. Recent trends in usable software design process and methodologies are presented in section IV. The software requirements elicitation and analysis for the field data collection application are described in section V. The user interface design process and usability design features of the prototype are highlighted in section VI. Object-oriented software architecture of the field application is presented in Section VII. The usability evaluation of the prototype

and its findings are discussed in section VIII. And finally, more research work is recommended in section IX.

## 2. Mobile Computing

Mobile devices (mobile phone, palm-top computer and tablet computer etc.) have made remarkable advances in terms of technological innovation. For instance, they now come with high resolution full-colour screens, Bluetooth, GPRS and various add-on accessories such as GPS, digital cameras, infrared scanners and so on. In addition, convergence of voice and data networks and the next generation networks (e.g. 3G [1]) on the horizon will only increase the data throughput, bandwidth and coverage area of mobile networks. While much of the underlying technology is already available in handheld and mobile computing, there are challenges with respect to the usability of mobile applications.

Usability is defined as "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use*" [2]. Contrary to what some might think, usability is not just the appearance of the user interface (UI) but also relates to how the system interacts with the user. Usability constitutes five basic attributes [2]:

1) ***Learnability*** – How easy it is to learn the main system functionality and gain proficiency to complete the job.
2) ***Efficiency*** – is measured in terms of the number of tasks per unit of time that the user can perform using the system.
3) ***User retention over time*** – It is critical for intermittent users to be able to use the system without having to climb the learning curve again. This attribute reflects how well the user remembers how the system works after a period of non-usage.
4) ***Error rate*** – This attribute contributes negatively to usability. It addresses the number of errors the user makes while performing a task.
5) ***Satisfaction*** – This attribute measures the user's subjective impression of the system.

In general, usability of a mobile computer is poor compared to its desktop counterpart. As a prime example, the uptake by customers in many countries for new WAP-enabled mobile phones has been disappointing [3]. The most common reason of failure is that web site designers simply try to carry their web sites over to the mobile Internet [4]. In order to produce highly usable mobile applications, software designers need to understand and take advantage of unique characteristics of mobile computing in terms of the technology, user population, usage patterns but more importantly, the context and the environment in which mobile devices are used.

Mobile computing is a relatively new paradigm that is fundamentally different from the much studied desktop computing [5]. There are a number of unique features and characteristics that are intrinsic to mobile computing. The mobile devices, for example, are resource-poor relative to their desktop counterparts. They are limited in terms of disk space, memory, processor speed, battery life and screen size etc. They are also dependent on the mobile networks that are highly variable in performance and reliability. The differences, however, are not limited to the technology but also in the usage patterns, the context and environment in which mobile computers are used. Mobile environment is heterogeneous where the users are faced with different situations that change constantly. In general, mobile work can be described as [6]:

1) *Dynamic* – the interaction between the system and user is highly fluid.
2) *Contextual* – applications are highly related to the context of where they are in [4], [7].
3) *Limited Attention* – users can only pay limited attention to applications.

The fieldworkers, for example, often find themselves in awkward situations where they need to crawl, climb or walk while they do their work. They are involved in tasks external to operating the mobile computers that demand a high level of visual attention (e.g. to avoid danger or monitor progress) [8]. The user's hands are also used to manipulate physical objects as opposed to users in a traditional office setting where the hands are safely and ergonomically placed on the keyboard. Under such circumstances, traditional UI design techniques and guidelines do not apply well to mobile systems [8], [9].

As a result, new paradigms for data presentation and user interaction techniques are explored to leverage the usability of mobile devices. For example, some researchers have used informative feedback mechanisms [10] and contextual information to reduce demand of user attention (i.e. "a minimum-attention user interface") [8], [9]. Others have discovered the most appropriate methods for displaying dynamic text on mobile devices with

various screen sizes [11] and innovative ways to reduce the size of UI components on without compromising the usability [12].

## 3. Mobile Application for Field Use

To be proficient and competitive in a global market means companies must properly manage their infrastructure, workforce and other resources. Asset management enables these companies to operate in an optimum efficiency by allowing better management decisions based on accurate information about their assets. One such case is in the utility industry (e.g. water, electricity and telecommunications) where there is a need to capture and maintain comprehensive information about the network according to their location, technical specifications and logical configurations (see Fig. 1). Only then they can carry out their service and maintenance operations efficiently and productively.

In many developing countries, including South Africa, the public utilities do not yet have complete information about their networks, and they are in the process of data collection and verification. One of the major challenges is to find appropriate methods and technologies needed to collect data from the field and integrate "seamlessly" with existing network information. One such method is to use teams of mobile workers to do a complete field audit [13]. Though labour-intensive, it is one of the most inexpensive and effective way to gain accurate information about network assets as they exist in the field. The Landbase Layer (see Figure 1) represents a geographic location of the asset and therefore implemented using GIS database systems. Typically, the mobile data collection personnel are given paper maps and data sourcing documents to collect geographically referenced asset information. However, these paper-based workflow systems are often time consuming, error-prone and difficult to create a heterogeneous platform to share information.

A mobile solution, on the other hand, allows a much more efficient information exchange between the field and the corporate database [14]. A typical scenario would be where the latest asset information is downloaded (e.g. via a wireless network in "real-time") to a mobile terminal in the field which the data sourcing personnel verifies and updates the changes back to the database. The mobile technology, of course, is not entirely new to the utility industry where the tablet computers and GPS devices were widely used in various field activities for many years. The new generation mobile devices, however, are much smaller, more technologically advanced and significantly cheaper [15], [16]. As a result, mobile devices are now required to do more sophisticated, integrated functionality (e.g. mobile GIS [17]) and they are used in greater numbers by a larger percentage of mobile workforce.

## 4. Usability Design Process

The use of appropriate usability engineering techniques and methods is crucial to a successful design and subsequent development of usable software. This is no easy assignment. There are differing viewpoints on how to achieve software usability. For example, some designers place great emphasis on the users whereas others more on the work that the users are involved in. This section gives a brief overview of some of the most recognized usability design guidelines, design processes and methodologies employed in the development of a usable software.

### *A. Usability Design Guidelines*

Basic usability design guidelines often help software designers make reasonable decisions that lead to highly usable systems. The guidelines include both the usability "rules" and design principles. The rules define the general character of what constitutes well-designed, usable systems. These rules provide a broad, overall directions for UI design, pointing designer towards a generally superior solution. The design principles, on the other hand, provide a narrower guidance on more specific issues in software design. There are many usability design guidelines (may extend to hundreds of pages), and it is beyond the scope of this paper to describe all of

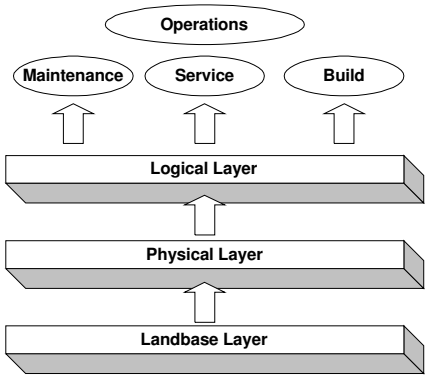



Fig. 1. Data model of network infrastructure in the utilities.

them. However, some of the most common usability rules and guidelines are listed below [18], [19]:

- Simple and natural dialogue
- Speak the user's language
- Minimize user memory load
- Strive for consistency
- Offer error prevention, and simple error handling
- Provide clearly marked exits
- Provide informative feedback
- Permit easy reversal of actions
- Support internal locus of control
- Enable frequent users to use shortcuts

Strictly speaking, these usability rules are simple rules of thumb that generally point the way to better designs but without any general guarantee of good results. These guidelines are very general and some of them may not be applicable for all software systems. They do not resolve

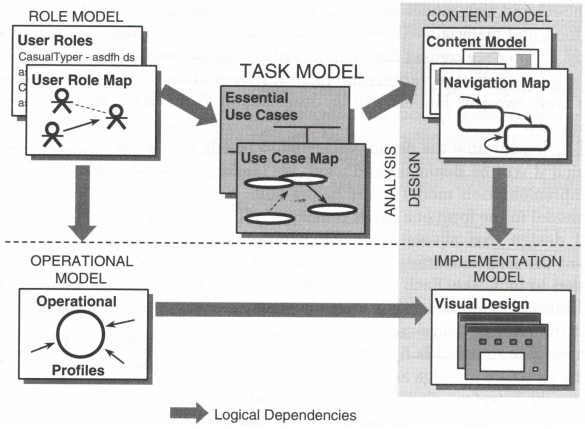



Fig. 2. A usage-centered design process [22].

design issues on their own and are no substitute for thoughtful analysis or inspired creativity on the part of the designer.

### B. User-Centered Design (UCD)

Basic computer programming and software development did not always have a concern for the end users or a focus on the usability of systems. Although software designed in a technically-oriented manner performs processing adequately, the code that is produced rarely meets human needs. In the past, a large percentage of computer users composed of programmers, technicians and engineers. The current users, however, are not dedicated to the technology, their background is more tied to workflow, and their use of computers may be discretionary. As a result, there is a gradual shift from focus on technology to a focus on people (users). This consciousness became a dominant force in software design and development in the form of a User-Centered Design (UCD) process. The UCD is essentially a user-centric software design methodology that focuses on users and their needs in an effort to design usable software. The UCD is defined as [20]:

"*an active involvement of users for a clear understanding of user and task requirements, iterative design and evaluation and a multi-disciplinary approach.*"

Unfortunately, the UCD practitioners do not always know how central the users are in the development of usable systems although the importance of user feedback is emphasized throughout the UCD process. A recent survey [20] has shown that only 13% of the UCD projects engaged in a full UCD approach in the sense of user involvement at all three stages of the development cycle. The methods used in the UCD process can include anything from informal usability testing, low-fidelity prototyping, heuristic evaluation and navigation design to scenario-based design. But there is a major discrepancy between the commonly cited measures and the actually applied methods [20]. For example, field studies were generally ranked high on practical importance but relatively infrequently used because they are costly, whereas heuristic evaluations are preferred because they are relatively easy and inexpensive.

### C. Usage-centered Design

A usage-centered design process also represents a shift in focus from the technology to the users. But unlike the UCD, the usage-centered design focuses primarily on the work or tasks that the users are attempting to accomplish rather than the users themselves. The usage-centered design views software systems as tools and "*to design dramatically more usable tools, it is not users who must be understood, but usage – how and for what ends software tools will be employed*" [21].

This model-driven design process uses the **essential models** [22] to capture the relationship of users and their tasks with the software system and subsequently determines the content and behaviour of user interface (see Figure 2). These abstract models are briefly described below [22]:

1) *Role model* – the relationship between users and the system.
2) *Task model* – the structure of tasks that the users will need to accomplish.
3) *Content model* – the tools and materials needed to be supplied by the user interface, organized into useful collections and the interconnections among these collections.
4) *Operational model* – the operational context in which the system will be deployed and used.
5) *Implementation model* – the visual design of the user interface and description of its operation.

## 5. Software Requirements

The software requirements for the field data collection application are identified and subsequently analyzed during the requirements engineering phase of the project. These requirements concern the usability, and functionality of software, as well as the target device on which the field application will run. The data sourcing work employed at Telkom, the country's largest telecommunications utility, is used as a case study to obtain software requirements for the field data collection application. This section briefly describes the data sourcing work employed at Telkom. In addition, the elicitation, analysis and specification of usability requirements for the field data collection application are explained.

### *A. UUC Data Sourcing Work*

Telkom keeps its vast, complex network of cables underground and these cables pass through the Underground Utility Closures (UUC). The UUCs are constructed as manholes that one typically finds on the city's streets and pavements. As part of an asset management, the utility keeps record of the network infrastructure by doing regular data sourcing activity. The data sourcing team is required to go inside the UUCs and obtain information about the network assets such as cables, ducts and joints etc. The data sourcing personnel are required to do a complete field data audit. They collect information about the UUC including its location, construction status, dimensions etc. They use paper street maps detailed with locations and IDs of UUC.

Inside the UUC, information collected includes the specifications of cables, joints and ducts. A UUC may contain hundreds of cables, joints and ducts depending on the size and location of the UUC. The interconnection (origin, destination, direction etc.) of these network elements are recorded using detailed drawings and specification tables. The data collection personnel works collaboratively in a team. Typically, each team consists of three members; one worker enters the data into the sourcing forms, the other two assist him by looking for the asset information. The workers communicate with each other verbally, and they "share" the asset information by reading out loud.

### *B. Requirements Elicitation*

The approach adopted in this project is to interact directly with the intended users of the field application in order to gather the most accurate requirements information. The UUC fieldworkers are the intended users, and they provided not only invaluable information about the data sourcing work but also about themselves. A multiple requirements elicitation technique is used to gather as much requirements information as possible. The techniques used include the user interviews, observation of their work and domain analysis. During the interviews, the fieldworkers were asked about the data sourcing work, and their background particularly the computer skills and familiarity with mobile technology. The data sourcing work is largely a form filling activity, and to understand what the work is about, it is appropriate to analyze the data sourcing forms. The information contained in these forms explained the nature of information sourced, the level of detail required, and the accuracy of data entered.

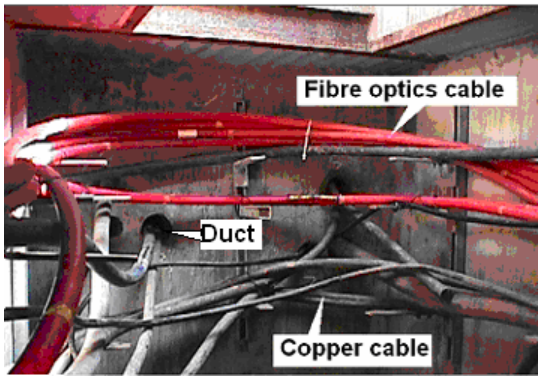



Fig. 3. The cable network inside the UUC.

Unfortunately, interviews and form analysis came short of explaining how the data sourcing work is actually done – in terms of interactions required, the context and environment etc. As a result, the data sourcing workers were observed while they do their work inside the UUC. From the data sourcing team, a group

leader was selected as a "field guide" and he was able to not only answer a broad range of issues relating to their work, but also highlighted the most important aspects of the fieldwork as well. A video recording of the site visits brought the field data back to the research lab and a more in-dept analysis of recorded material is done by identifying important details that were previously overlooked.

### C. Requirements Analysis

The software requirements are analyzed in terms of the needs of intended users, the nature of data sourcing tasks and the operational context of fieldwork. An in-depth understanding of the relationship of users with the system is established through the **user role model** [23]. It identifies a number of different roles of users to be supported by the field data collection application. In doing so, it captures the expectations, behaviour and responsibilities of the users in each role. Similarly, the **task model** [24] is used to understand in substantial detail what the users will be trying to accomplish in their work using the field application, and how they will need to go about it. Part of this model-driven task analysis process is the use of ***essential* use case** [24]. An essential use case focuses on the purpose or intentions of the user rather than on the concrete steps or mechanism by which that purpose or intention is carried out. This problem-oriented view of the tasks leaves open many possibilities for the subsequent design and implementation of the user interface.

The context in which data sourcing work takes place (i.e. operational context) is captured in the **operational model** [25]. The operational model constitutes characteristic of users and those important aspects of work that are most likely to affect the usability design. The operational context is important because it affects various design objectives, such as the speed of operation, accuracy, ease of learning, readability, and the like. It also has direct impact on highly specific design decisions and details, such as the appropriate use of sound, colour, arrangement of UI controls etc.

The constituents of the operational model for the field data collection application are outlined below:

1) *User Profile* – The UUC fieldworkers have the expertise and knowledge about the fieldwork but limited computer skill.
2) *Proficiency Profile* – The UUC fieldworkers are expected to use the field data collection application as well-informed users (as opposed to experts or novice users). It is expected that users would be given some basic training before they use the application.
3) *Interaction Profile* – Because the UUC fieldworker works collaboratively with his co-workers, he is always under pressure to keep up with the rest of the team without slowing them down. Under these circumstances, he learns to do thing quickly and opportunistically. The result is a high rate of user interactions amplified by the repetitive nature of data sourcing tasks. There is no predefined sequence of interaction by the user when he is entering the data into the sourcing templates. Consider a scenario where the user prepares to enter a cable specification number but his colleague is still busy finding it. Then another co-worker shouts out the specification number of another cable, forcing the user to abandon the former to search for a new cable in the cable network.
4) *Information Profile* – As would be expected, information flows predominantly from the user into the system. By and large, the data sourcing work is not a problem-solving activity hence the information that is entered is not calculated but rather obtained through verbal and visual means (i.e. through observations and asking co-workers). The nature of information sourced is rather straightforward but most of the information is presented graphically (drawings, tables, schematics etc.) and hence may appear to be somewhat complex (see example in Figure 6). The volume of information, of course, varies according to the number of cables and ducts inside the UUC. A typical UUC contains a few dozen cables, joints and ducts. If one considers approximately eight data attributes (specification number, construction status etc.) for each of these elements, it adds up to hundreds of separate data attributes collected from a single UUC.
5) *Environment Profile* – The data sourcing work takes place both inside and outside the UUC. Outside the UUC, the user would be standing on the street or pavement exposed to the environmental elements (rain, bright sunlight etc.). In addition, the user needs to contend with noise generated by motor vehicles and other road users. Inside the UUC, however, visibility is poor due to lack of natural light. There is also a moderate level of noise and echoes generated by verbal communications between fieldworkers as well as constant beeps and alarms from safety

equipments (e.g. gas detector).

### D. Usability Requirements

The usability requirements for the UUC data collection application are specified as follows:

1) The user interface should be highly intuitive so that the users with limited computer skills will be able to relate it to their work with minimum technical support (help manuals, training etc.).
2) A high rate of user interaction and the repetitive nature of data sourcing tasks call for a user interface that is optimized for efficiency.
3) A high rate and concentration of interaction often results in the user making errors, hence the user interface should be fault tolerant, or better still, minimize and prevent users from making errors in the first place.
4) The user often works collaboratively, drawing his attention away from the system. Hence, the user interface should demand minimum amount of user attention.
5) Because the data sourcing work is primarily about the data entry, the system should be oriented towards receiving and validating data with minimum effort from the user.
6) A fairly high volume and graphical nature of information that is sourced needs to be presented clearly and comprehensively on a relatively small screen of a mobile device.
7) Due to a lack of predefined sequence of action in a data entry work and generally unpredictable nature of fieldwork, the user should be given control of the system whenever possible (i.e. the need for flexibility).

### E. Functional Requirements

A high-level overview of functionality the field data collection application is specified after the analysis of users, their work and operational context. A detailed functionality of the application, however, is presented in the software design model. A brief overview of functionality of the field data collection application is described below:

1) Provide the user with all the UUC sourcing templates on a mobile device for field data collection.
2) Save the asset information in a format that is compatible or can be read from corporate GIS. For example, most commonly used files are XML and .txt files. And temporarily store the asset information on a mobile device until the data can be synchronized with a database on a desktop computer.
3) Present the user with a data sourcing map that contains details such as the asset identification, location, names of streets and suburb etc. The user must be able to navigate and search for assets, streets and suburb etc.

### F. Hardware Requirements

It is important that the target device has the capability to support functionality required by the field application. It is equally important that it is able to operate effectively in the environment and the context in which field application will be used. The goal, therefore, is to find the most suitable mobile device for the field data application by specifying the hardware requirements. The criteria for the mobile device for field data collection application are that it must:

1) Be lightweight and portable (preferable to be able to carry in one hand).
2) Be water-proof, dust-proof and shock-proof (or drop-proof).
3) Operate equally well in all light conditions from darkness to direct sunlight.
4) Have a display with high resolution to view data sourcing map with a high level of detail.
5) Have a wireless network capability as well as a serial connection to a desktop computer for data synchronization.
6) Have enough disk space to store the asset information collected over a period of a few weeks or months.
7) Support a rich set of text-entry tools or input devices for data entry.
8) Have a battery life that lasts for at least one working day (approximately 8 hrs).

## 6. User Interface Design

The user interface design is probably the most important aspect of a usable software development process. This is because the users interact with the system primarily through the user interface. Hence, most of the usability problems that the user encounters are directly related to the user interface.

A usage-centered design [21] approach is used to design the user interface for the field data collection application. An in-dept knowledge about users, their tasks and the operational context gained during the requirements analysis phase directs the user interface design process (as shown in Figure 2). A two-step approach is adopted to produce the user interface. The content of user interface is first determined and thereafter its layout, appearance and behavior are implemented. The user interface is implemented as a working prototype on a pocket computer. The design and implementation of the user interface are discussed in the sections below.

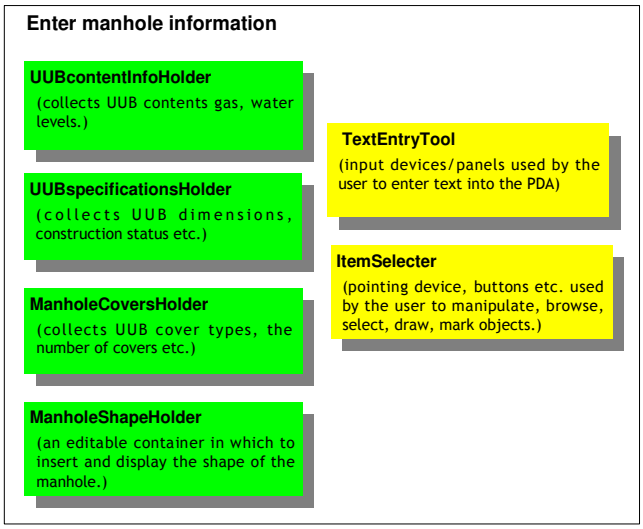



Fig. 4. The content model for Enter Manhole Information interaction context.

## A. Content Model

The contents of various interaction contexts in which field data collection work takes place is represented using the content model. The content of interaction context contains collections of abstract *tools* and *materials* needed to carry out some particular task or set of interrelated tasks. The *tools* supply the functions and active capabilities required to complete a task whereas *materials* are the data containers, displays or work areas upon which the *tools* of the user interface can operate (see example in Fig. 4). Each interaction context supports one or more essential use cases. The use cases that closely resemble each other are supported by a common interaction context.

The interrelationships between different interactions contexts are represented using a **navigation map** (see Fig. 5). This map determines the navigation patterns between various screens of the user interface later on in the implementation phase.

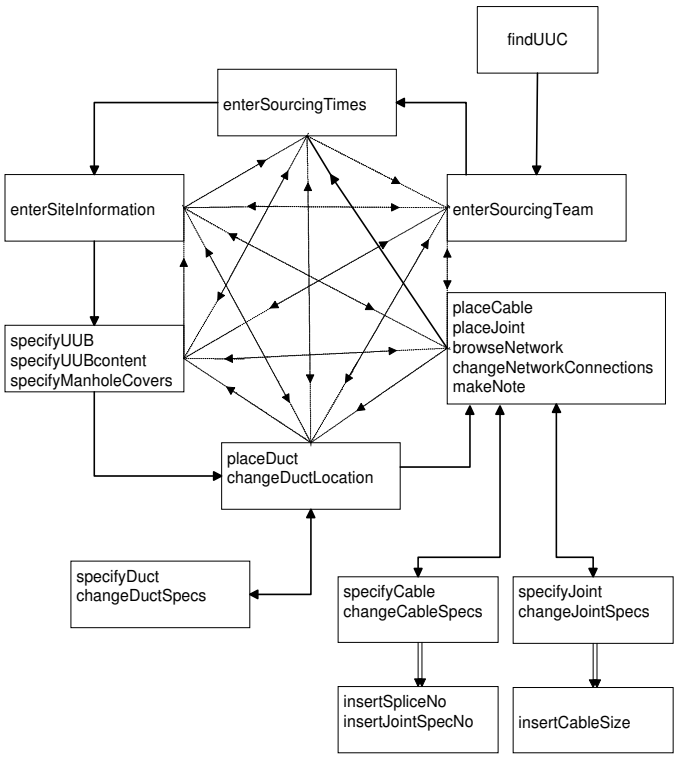



Fig. 5. The navigation map for interaction contexts.

## B. Implementation Model

The implementation model is a representation of how the user interface will actually look and function. In other words, the abstract components of the content model are now implemented into tangible user interface components (e.g. buttons, textbox, screen etc.). In order to close the gap between the abstraction and final implementation of the user interface, the following questions are addressed:

1) *Contexts* – What are the implemented interaction contexts?
2) *Components* – What are the user interface components within contexts?
3) *Composition* – What is the layout and organization of components in each interaction context?

The implementation model is realized using a high-

fidelity prototype – one that closely resembles what the final system would appear and behave. Unlike a paper prototype such as a sketch of user interface, a functional or high fidelity prototype requires a hardware platform for it to run on. For this purpose, three types of mobile devices, namely a mobile phone, a Personal Digital Assistant (PDA) and a tablet computer were evaluated for suitability according to the software, hardware and usability requirements of the field data collection application. It emerged that the PDA is the most suitable device of the three. It has a superior functional capability (e.g. better text-entry tools, bigger disk space, expansion packs for GPS, bar-code scanner, GPRS etc.) than a mobile phone, and significantly cheaper and more portable than a tablet computer. The iPaq™ H3900 from Compaq™ was selected as the target device which runs on a Pocket PC operating system. A conscious decision was taken to run the prototype on this particular platform although it is being criticized for duplicating its desktop counterpart [8], [9]. This project, however, took a pragmatic approach by selecting a platform with the most sophisticated functionality which makes it most likely to be used for a modern field data collection application.

### C. Usability Features

The main usability features of the user interface for the UUC data collection prototype are highlighted below:

1) *Efficiency* – The prototype is designed to increase the speed and accuracy of the data sourcing work. It does this by limiting the number of user actions to a minimum when entering data into the system. For example, the user selects from a predefined list or combo box rather than typing in the data using Soft Input Panel (SIP). The latter is more time-consuming than the former and generally, data entry tools such as the handwriting recognition software present a steep learning curve for the novice users. Automating certain data entries also helps improve the speed and accuracy of the data entry work. The user errors are prevented from taking place in the first place by disabling controls when they are not applicable. For entries that cannot be pre-populated or automated (i.e. UUC dimensions, gas readings etc.) the data is validated on entry. It is accompanied by a comprehensive error message if the user fails to enter data correctly. Resource-intensive items such as graphics and long specification lists are prepopulated so that the user waits only once when the program is loading rather than during the data sourcing tasks.
2) *Demand of user attention* – A number of design techniques are used to achieve a minimum-attention user interface. Firstly, the user is constantly informed about the status of the system by means of audio and visual feedbacks. The visual components, for example, are enhanced with unique sounds to provide informative feedback to better locate targets on the user interface. And secondly, user memory load is reduced by using graphical displays (as opposed to describing in text – "*a picture worth a thousand words*"), automation and predictive system response. The system, for example, automatically traces the cable path when the user clicks on the duct to which it is connected. If the user has to manually browse through the entire cable network to find a particular cable, it would require a high level of concentration and visual attention from the user. In an another example, the user is assisted in finding a sense of direction and orientation inside the UUC by providing the user with topographic view of the UUC that dynamically displays the routeface that he is working on and its orientation with respect to other routefaces (see Fig. 6).

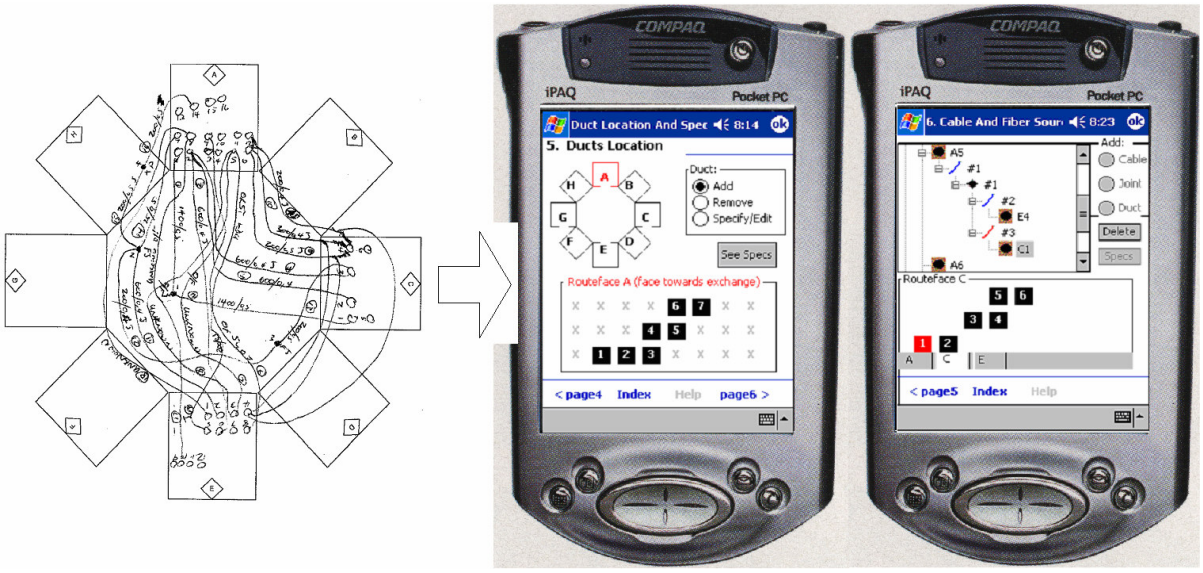



Fig. 6. The user interface for the topographic view of UUC cable network.

3) *Presentation* – The appearance of the user interface is designed to be simple, intuitive and aesthetically appealing. Each screen of the user interface has a well-defined structure which is consistent across all screens. The headings, content and navigation bar are placed at the top, center and bottom of the screen respectively. Differing colours and fonts are used to distinguish various components and more importantly, their functionality. It is assumed that the users are not necessarily knowledgeable with the use of standard Windows widgets and components that come with a Pocket PC platform. Hence, their functionality is made obvious to the user. The use of pop-up, nested menus and other hidden features are avoided. The user interface is further simplified by presenting only those functions that are absolutely necessary to complete the data sourcing tasks – the notion of What You See Is What You Need (WYSIWYN) is strictly applied. The biggest challenge is probably "converting" graphical drawings on the paper sourcing into a format that is suitable for use on a relatively small PDA screen. Although there are tools (e.g. Scalable Vector Graphics [26]) specifically designed to display graphics on mobile devices, use of innovative metaphors and new information presentation styles are often necessary to ensure comprehensibility and ease of user interaction. The tree architecture, for example, is more suitable to show interconnections of network nodes on a PDA screen rather than merely displaying the sketch of cable network on the mobile device (see Fig. 6).

4) *Human-Computer Interaction* – The nature of user interaction is primarily a *direct manipulation* where the user taps the touch screen of the PDA using a stylus. The SIP is a primary means of text entry for a PDA and it forms part of the direct interaction with the system. The indirect means of interaction includes an external keyboard, a navigation button and programmable shortcut buttons. The external keyboard is awkward to use because the user is often standing and there is no flat surface inside the UUC to place the keyboard. A direct-manipulation interaction style remains the most effective means of user interaction with the PDA. Hence, the prototype is designed to exploit a wide range of gestures and movements of the stylus on a touch screen. For example, dragging the page using a stylus is as effective as using a vertical scrollbar. The user interface is also designed to quickly respond (by visual and audio means) to user actions which happen relatively rapidly with the direct manipulation interaction style.

5) *Navigation* – The navigation on the field data collection prototype is based on the navigation map (see Fig. 5) of interaction contexts. By and large, navigation patterns are designed to support a workbench-style user interaction. This style of navigation provides the user with the freedom to move between different interaction contexts and adopt the workflow that is most appropriate for him. On the surface, a step-wise pattern of workflow

seems to provide a simpler user interface because at each step the user typically makes a single choice or enters one bit of information before going on to the next step. The consequence, however, is that there are too many tiny, time-consuming steps which are unsuitable for frequent and repetitive tasks such as specifying hundreds of network nodes. The workbench interaction, on the other hand, requires significantly fewer steps but the outcome is a compact, flexible and feature-laden user interface.

6) *Data*-sourcing map – an off-the-shelf GIS mapping software from Intergraph called IntelliWhere OnDemand™ is used to display the data-sourcing map (see Figure 7). The user interface prototype runs as a custom application over this software on a PDA. The data is downloaded to the client from the desktop GIS software namely GeoMedia™. After the data sourcing is complete, the data is uploaded back to the desktop computer and the changes synchronized with the database (MS Access, SQL etc) using ActiveSync™.

## 7. Software Design

An object-oriented software development process is used to document detailed functionality and data structures of the field data collection application. Although the primary objective is to illustrate the complexity and architecture of the field application, software design models can reveal more information about the functionality of the user interface than the implementation model. This information is necessary given the fact that usability depends to a large extent on how the user interface functions and interacts with the user. The software development process adopted is called the ICONIX process (see Fig. 7) which uses a subset of Unified Modeling Language (UML). It is a relatively lightweight object-oriented design process that "*sits somewhere between the very large Rational Unified Process (RUP) and the very small eXtreme programming (XP) approach*" [27]. It is particularly suitable for relatively small software projects because it does not have a lot of overhead that the often large and unwieldy UML brings.

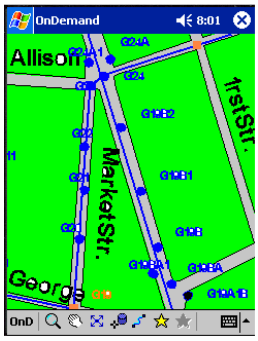



Fig. 7. Data-sourcing map implemented using IntelliWhere OnDemand™.

The ICONIX process begins with the user interface prototype (see Fig. 8). Hence, its design models are directly linked to the graphical components and features on the user interface. For example, the text for a given use case describes the system response to a user action performed on a particular GUI component. The ICONX process describes both the static components (i.e. data structures) and dynamic behaviour of the system designed (see Figure 7). These components of ICONIX process are briefly described below [28]:

1) **Use case model** – describes the runtime behavior of the system by answering question of what the users are trying to do with the system by capturing user actions and the associated system responses in great detail.
2) **Robustness diagram** – identifies objects that are needed to complete use cases in the use case model. It is similar to a UML collaboration diagram, in that it shows the objects that participate in the scenario and how these objects interact with each other.
3) **Domain model** – identifies the main conceptual objects that are going to participate in the system designed.
4) **Sequence diagram** – describes the system behavior in great detail in a sequential manner. In doing so, it encompasses the *basic course* and all *alternate courses of action* within each of the use case.
5) **Class diagram** – describes how the software program code is organized by specifying classes,

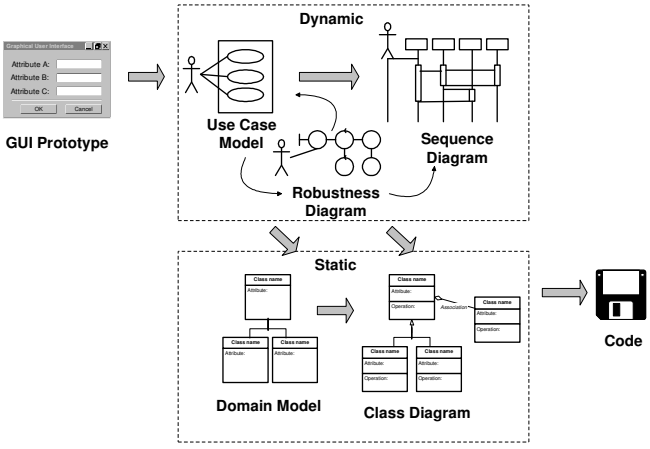



Fig. 8. ICONIX design process [26]

interfaces, collaborations and their relationships.

## 8. Usability Evaluation

An empirical usability test was carried out in the field at the actual data sourcing site to evaluate the usability of the prototype. A team of data sourcing personnel from Telkom again participated in the usability test. The test participants were trained for a few hours in order to orientate and familiarize with the prototype before the testing could begin. The participants were asked to complete a list of data sourcing tasks using both the prototype and a paper sourcing form. The duration and accuracy of these tasks were recorded using paper logging forms. During the test, they were actively probed for their understanding of the tasks. They were also asked to speak out loudly whatever they were thinking (intentions, frustrations and comments etc) while they are doing the tasks. The participants were again interviewed after the test and asked to complete a set of questionnaires addressing overall usability of the prototype as well as specific user interface design issues.

Both the performance and subjective measures of the field data collection prototype were collected during the usability test. The former is a measure of how well the participant can carry out the data sourcing tasks using the prototype relative to that of paper sourcing forms. The performance measures include:

1) **Completeness** – is the percentage of total assigned work completed within the allocated time.
2) **Correctness** – is the percentage of completed work that is correct.
3) **Effectiveness** – is the correctly completed work as a percentage of total work.
4) **Efficiency** – is the **Effectiveness** per unit time
5) **Productiveness** – is the percent of total subject time spent productively. Unproductive time includes time spent seeking help from the test conductor.

The performance measures are calculated as follows:

**Completeness** = $(T_f / T_p) \times 100$
= (811s/1137s) × 100
= 71.33 %
**Correctness** = ((correct data items)/ (total data items)) ×100
= (82/85) × 100
= 96.47 %
**Effectiveness =** (Correctness × Completeness) / 100
= (96.47 × 71.33) / 100
= 68.81 %
**Efficiency** = $(1/ T_p) \times$ Effectiveness
= (1/1137s) × 68.81
= 0.061 % per second
**Productivity** = $((T_p - T_h)/T_p) \times 100$
= ((1137s – 33s)/1137s) × 100
= 97.10 %

The $T_p$ and $T_f$ refers to an average amount of time taken to complete all the data sourcing tasks using the prototype and data sourcing form respectively. And $T_h$ refers an average amount of time taken to seek help from the test conductor (i.e. unproductive time).

The overall performance of the prototype is more than satisfactory. Bear in mind that this is the first time that the participants used the prototype (training aside) whereas they have been using the paper sourcing forms for the past few months. To make a fair comparison, one should really allow the participants to use the prototype for the same amount of time as the paper sourcing forms. Nevertheless, the performance data did help find a number of usability problems and explained the cause of these problems. Some of the most promising results came from the user's comments and suggestions. On average, the participants found the data sourcing tasks "easy" to do using the prototype and they were very keen to use the prototype in their work. However, the participants were unable to provide clear suggestions on possible improvements for the prototype.

Unfortunately, there is no accurate means of measuring software usability. This project has intentionally placed emphasis primarily on the user interface design techniques rather than relying on users or usability test as a whole to determine the usability of the prototype. This is because users often do not always know what they really want and are easily influenced. Usability testing is also very subjective and the results vary depending on a number of factors such as the number of participants, expertise of evaluator, test environment etc.

## 9. Future Research

Undoubtedly, mobile technology will continue to advance particularly in the areas of hardware, operating system, programming tools and development environments. The .NET platform [29], for example, provides a framework for data exchange, overcoming barrier to interoperability of applications across various smart devices (e.g. workstations, servers, handheld computers, smart phones etc.). XML has emerged as the standard format for data exchange between various components in Geospatial Information Systems (e.g. GML, LandXML etc.) [30]. Another XML variant, SVG [26] is also able to exchange graphical data across

resource constraints mobile devices. This technology would be particularly salient for mobile field applications (data collection and maintenance) where use of graphical data such as maps, detailed drawings and schematics are often necessary. Having mentioned some of the recent developments in mobile technology, more research work is still needed to fully explore how they will affect the usability of mobile field data collection applications.

## 10. Conclusion

Rapid advancements in mobile technology have become a fundamental challenge for designers to produce mobile applications with high level of usability. This project developed a working prototype to investigate the design of a usable mobile field application based on a field study of data sourcing work at a telecommunication utility. A usage-centered design approach is used to design the user interface. The usability requirements for the mobile field data collection application are presented. In addition, the most pertinent usability features and design guidelines are discussed. Having said that, they are based on a case study and hence may not be applicable to every usability design situation. Fieldworks generally have their own unique characteristics, and so is the platform on which the mobile application is implemented. There are also the user-centered and context-oriented design approaches that can be employed to produce usable software. All in all, case studies such as the one presented in this project are needed to explore the possibilities and design challenges of usable field applications and mobile computing as a whole.

### REFERENCES


[1] Mobile Streams "Yes 2 3G - White Paper". INTERNET. http://www.mobilewhitepapers.com/pdf/3g.pdf , Cited 19 January 2003.

[2] Ferre, X., N. Juristo, H. Windl, and L. Constantine "Usability Basics for Software Developers", *IEEE Software*, vol 18, no 1, February 2001, pp 22-29.

[3] Buchanan, G., *et al.* "Improving mobile internet usability", *Proceedings of the tenth international conference on World Wide Web*, ACM Press, New York, ISBN:1-58113-348-0, pp 673-680.

[4] Van Welie, M., and de Ridder, G., "Designing for Mobile Devices: A Context-Oriented Approach", *Mobile HCI 2001: Third International Workshop on Human Computer Interaction with Mobile Devices*, Lille, France, 2001.

[5] Satyanarayanan, M. "Fundamental Challenges in Mobile Computing", *Proceedings of Fifteenth annual ACM symposium on Principles of distributed computing*, May 1996, ISBN:0-89791-800-2, ACM Press, New York, pp 1-7.

[6] Perry, M., K. Ohara, A. Sellen, B. Brown and R. Harper "Dealing with Mobility: Understanding Access Anytime, Anywhere", ACM *Transactions on Human Computer Interaction*, vol. 8, no. 4, December 2001, pp 323-347.

[7] Dey, A K., and G D. Abowd "Towards a Better Understanding of Context and Context-Awareness", *CHI 2000 Workshop on the What, Where, When and How of Context-Awareness*. April 2000, Hague, Netherlands.

[8] Kristoffersen, S., and F. Ljungberg "Making Place to make IT work: empirical explorations of HCI for mobile CSCW", *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, ACM Press, New York, ISBN:1-58113-065-1, November 1999, pp 276-285.

[9] Pascoe, J., N. Ryan and D. Morse "Using While Moving: HCI Issues in Fieldwork Environments", *ACM Transactions on Human Computer Interaction*, vol. 7, no. 3, September 2000, pp. 417-437.

[10] Poupyrev, I. , S. Maruyama and J. Rekimoto "Ambient touch: designing tactile interfaces for handheld devices", *Proceedings of the 15th annual ACM symposium on User interface software and technology*, ACM Press, New York, ISBN:1-58113-488-6, October 2002, pp 51-60.

[11] Laarni, J. "Searching for optimal methods of presenting dynamic text on different types of screens", *Proceedings of the second Nordic conference on Human-computer interaction*, ACM Press, New York, ISBN:1-58113-616-1, October 2002, pp 219-222.

[12] Brewster, S. "Overcoming the lack of screen space in mobile computers", *Personal and Ubiquitous Computing*, vol. 6, issue 3, January 2002, pp 188-205.

[13] Kuhl, L. "Doing More with Less: Leveraging Your Spatial Asset Data", *Proceedings of the 26th annual GITA Conference* [CD-ROM], March 2003, San Antonio, USA.

[14] Indus International "Asset Management - Using Mobile Computing to gain Competitive Advantage". INTERNET. http://www.cmmscity.com/articles/MobileComputing_wp.pdf, Cited 27 January 2003.

[15] Fletcher, D. "Field Applications at DTE Using the Pocket PC", *Proceedings of the 26th annual GITA Conference* [CD-ROM], March 2003, San Antonio, USA.

[16] Interactive Business Systems, Inc. "Palm Pilot Data Collection for the Mobile Workforce". INTERNET. http://www.ibs.com/pdf/palm.pdf, Cited 27 January 2003.

[17] Randell, B. "Implementing a Mobile GIS to Enhance Efficiency of Field Operations", *Proceedings of the 26th annual GITA Conference* [CD-ROM], March 2003, San Antonio, USA.

[18] Nielsen, J. and R. Molich "Heuristic evaluation of user interfaces", In *Proceedings of CHI'90*, ACM Press, New York, April 1990, pp 249-256.

[19] Shneiderman, B. "Designing the User Interface: Strategies for Effective Human-Computer Interaction", 3rd Edition, Addison Wesley Longman, ISBN: 0-201-69497-2, 1998, pp 67.

[20] Vredenburg, K., J. Mao, P. Smith and T. Carey "A survey of user-centered design practice", *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, ACM Press, New York, ISBN:1-58113-453-3, April 2002, pp 471-478.

[21] Constantine, L. and A. Lockwood "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design", ACM Press, New York, ISBN: 0-201- 92478-1, 1999, pp 23.

[22] Constantine, L. and A. Lockwood "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design", ACM Press, New York, ISBN: 0-201- 92478-1, 1999, pp 30-32.

[23] Constantine, L. and A. Lockwood "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design", ACM Press, New York, ISBN: 0-201- 92478-1, 1999, pp 78-84.

[24] Constantine, L. and A. Lockwood "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design", ACM Press, New York, ISBN: 0-201- 92478-1, 1999, pp 99-119.

[25] Constantine, L. and A. Lockwood "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design", ACM Press, New York, ISBN: 0-201- 92478-1, 1999, pp 298-313.

[26] W3C "SVG - XML for the web". INTERNET. http://www.w3.org/Graphics/SVG/, Cited 20 July 2003.
[27] Rosenberg, D. and K. Scott "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, 2001, pp 1.
[28] Rosenberg, D. and K. Scott "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, 2001, pp 1-15.
[29] Microsoft "What is .NET". INTERNET. http://www.microsoft.com/net/basics/faq.asp, Cited 23 August 2003.
[30] Martin, I. "XML - Gateway to Interoperability", *Proceedings of the 26th annual GITA Conference* [CD-ROM], August 2003, San Antonio, USA.

# CONCLUSIONS

## 1 Achieving the project's objectives

Recent years have seen the use of mobile technology in a wide range of applications from Wireless Access Protocol (WAP) and GPS navigation systems to mobile GIS mapping solutions. Usability is a crucial requirement if these applications are to be used effectively by the users. The aim of this project is to study usability of mobile applications for field data collection in utilities. The project investigated usability requirements for a field data collection application and more importantly, how these requirements are met.

To achieve this objective, the project reviewed literature that is pertinent to the research. The topics include mobile computing, mobile GIS application and usability design process. A case study of the data sourcing work employed at a telecommunications utility is used to gather information about intended users, the environment and operational context of the field data collection work. This information is then analyzed before specifying the usability, functional and hardware requirements. A model-driven usage-centered design and ICONIX software design processes are used to design user interface and software architecture respectively. The user interface design is implemented using a working prototype on a PDA. The prototype is then evaluated for usability with the data sourcing personnel in the field. The test results and recommendations for future research are discussed.

The project is considered to have accomplished its objectives. The usability requirements for field data collection application are identified and the user interface prototype has demonstrated how these requirements are met using appropriate UI design techniques. The usability test is able to obtain very positive feedback from the intended users. However, it is felt that there is room for more research and expansion of the functionality of the prototype as new technologies emerge (see section on Developments and Recommendations below).

# 2 Conclusions regarding the literature survey

The literature survey informed the reader about a number of topics pertinent to the subject of this research. They include most recent literature on mobile technology, mobile field applications and usability design process. The section on mobile technology highlighted recent technological development in terms of hardware, wireless networks and mobile applications. But more importantly, it discussed unique characteristics that are intrinsic to mobile computing such as, hardware constraints, operating context and usability requirements. The argument here is that mobile computing is a relatively new paradigm that is fundamentally different from traditional desktop computing and as a result software designers need to take these factors into careful consideration if they are to produce usable mobile applications.

An extensive use of mobile technology for field data collection work is discussed in details. The survey began with discussion around field data collection in public utilities and some case studies in which mobile GIS technology is used to obtain data from the field. But the main focus was given to the way and environment in which fieldwork is carried out. Mobile workers operate in operating context and environment that is inherently heterogeneous where users are faced with different situations that change constantly. In general, mobile work can be described as dynamic, contextual and demands high level of user attention. This demands innovative HCI techniques and usability features from designers. In general, mobile applications require a much higher level of usability than their desktop counterparts.

The survey gave a summary of a number of usability design guidelines, techniques and methodologies currently used by designers. The usability guidelines include definitions of usability rules and design principles widely recognized by the software community. Two model-driven UI design processes called User-Centered Design and Usage-Centered Design are discussed. Both represent a move away from techno-centric way of designing software. The difference however is that the latter focuses on user to obtain software requirements and the latter on the operation context and usage. The survey ends with a section on various techniques used in

usability testing. They are essential for designers if they are to design and effectively evaluate software that meets a high level of usability requirements.

# 3 Conclusions regarding the software requirements

A proper understanding about the field data collection work is obtained from the requirements elicitation and analysis process. A direct interaction with actual fieldworkers at the data sourcing site was productive and it produced relatively accurate and reliable requirements information. These requirements, however, are not necessarily applicable to all types of data collection work. This is because the case study of field data collection work is based on a UUC data sourcing work which has characteristics that are unique to this particular fieldwork.

The use of usage-centered design process in requirements analysis proved to be effective in obtaining an in-dept understanding of intended users, their background and more importantly, their tasks and operational context of their work. It is felt that the software requirements that were obtained address the usability and functional aspects of the field application in a practical manner. Furthermore, the hardware requirements for the field application reflect realistic expectations in terms of functional capabilities and sophistication from a modern mobile device.

# 4 Conclusions regarding the user interface design

The user interface is designed using a two-steps approach. The content of user interface is addressed before determining how they will look or behave. The latter is derived from the requirements analysis of users, tasks and operational context. The result is that the user interface is able to give the users what they need rather than what is available.

The user interface design is implemented as a working prototype on a Pocket PC operated PDA which has shown to be the most suitable hardware platform for the prototype to run on. This operating system is criticized for merely duplicating desktop software on a mobile device. However, the reality is that no other operating system is able to give the kind of underlying functionality in terms of multimedia and wireless connectivity required by a modern field data collection application. Again, this is a pragmatic approach and may not be able to provide a truly innovative solution. Another concern is that the programming language used, namely the eMbedded Visual Basic (eVB), restricts the flexibility of the user interface design due to its limited availability of UI components and widgets. To overcome this problem, standard GUI components are used creatively so that they can still appear intuitive to novice users. By and large, the user interface design is able to meet the usability and functional requirements successfully despite some of the limitations mentioned above.

# 5 Conclusions regarding the software design

The software usability is not merely how the user interface looks. To a large extent, it concerns the behavior of software particularly how it interacts with the user. With this in mind, the functionality of field data collection application is presented in the software design. The software design process used, namely the ICONIX provides a framework to comprehend not only the behavior but also the structure and complexity of software as well. It is felt that the ICONIX design process is able to do this sufficiently without the complexity of a much larger object-oriented design process like Rational Unified Process (RUP).

Some of the contents of software design such as the class diagrams do not really affect the software usability. However, these models allow the software developers to extend the existing functionality and even to continue with the implementation of the prototype to fully functional software. This project, however, has not implemented the software design into a complete, fully functional software system since it is deemed unnecessary in terms of the objective of this research.

Nevertheless, object-oriented design will enable easier reuse and maintenance of software components especially when sourcing templates are modified regularly. Overall, the software design process used has adequately explained the behavior and functionality of field data collection application. However, it is felt that a superior design could be achieved if more time was spent on improving the design in an iterative manner.

# 6 Conclusions regarding the usability evaluation and test results

The usability of field data collection prototype was evaluated using an empirical field test with the UUC data sourcing personnel. Field test was carried out at the actual data sourcing site where the test participants were given the prototype to do their work. The time taken to do the data sourcing tasks using the prototype was recorded and then compared with that of paper sourcing forms. The participants were then interviewed and asked to complete questionnaires that probe their subjective views on usability of the prototype. The number of usability problems discovered by the usability test is considered to be relatively low but nevertheless corresponding recommendations were made to correct these problems.

Overall, the qualitative test results are very encouraging. The majority of the fieldworkers found the prototype easy to use and keen to use the application in place of paper sourcing template that they are currently using. However, the accuracy of quantitative data is questionable due to a number of shortcomings in usability testing. Firstly, the number of test participants and the frequency of usability testing are insufficient. It is felt that the usability test could have been executed far better by increasing the number of test participants and frequency of tests over a longer period of time. Secondly, without proper data logging equipments (video camera, data logging software etc.), a one-person team has little control over the test conditions in the field. This could be one of the reasons why there were very few usability problems uncovered by the usability test. Another concern is that the test conductor has a limited expertise in conducting a usability test. Given these

conditions many usability problems may have gone undetected. And finally, it is felt that test participants were easily influenced by the test conductor and did not have enough exposure to other mobile applications to make impartial evaluation of the prototype.

Ideally, the number of test participants should be increase to about 7 or 8 people and the tests be carried out over a period of 3 or 4 days. The main purpose of usability evaluation is to fine tune critical design features and it is crucial that redesign takes place over a number of iterations.

# Developments and Recommendations

## 1 Developments in Technology

Mobile technology is changing very rapidly. Even as we speak, there are new developments in operating systems, hardware, programming tools and so on. As a result, the functional capability and sophistication of mobile applications will continue to increase. This section provides a brief description on more recent developments in technology that are most likely to have an impact on the design and development of usable mobile field applications in the near future.

### 1.1 Extensible Markup Language (XML)

XML is a text-based, structured file format that presents a method for storing and exchanging data between disparate applications. It was traditionally used for structured data exchange on the Internet but it has grown into a nearly-ubiquitous format for data storage and information exchange. Not surprisingly, XML is now becoming a standard data exchange format for geographical information including both spatial and non-spatial properties of geographic features. Many of the XML related technologies such as the MultiSpeak, Geography Markup Language (GML) and LandXML are now used to resolve data interoperability problems in a wide range of Geospatial Information Systems (GIS) applications from land planning and surveys to field workforce automation in utilities (Martin I, 2003).

### 1.2 Scalable Vector Graphics (SVG)

SVG is a relatively new XML grammar for defining vector-based high quality 2D graphics for the Web and mobile applications. The SVG drawings are lightweight (much smaller and compressible than GIF, BMP and JPEG formats), dynamic (scalable, zoom in/out, searchable, animation etc.) and interactive (e.g. event handlers for mouse clicks). These unique features of SVG make it very suitable to

display and manipulate detailed graphics used in mobile field applications such as data sourcing maps, line diagrams and schematics. Recently, W3C (2003) has introduced a SVG standard, namely SVG 1.0 to address a mobile device as a target area for vector graphics display. Since each mobile device has different characteristics in terms of CPU speed, memory size, colour support etc, W3C defined two low-level profiles as subsets of SVG 1.0. The first profile, SVG Tiny (SVGT) is suitable for highly restricted mobile devices such as mobile phones, whereas the second profile, SVG Basic (SVGB) is targeted for higher level mobile devices such as PDAs.

### 1.3 Microsoft .NET

The .NET is the Microsoft's solution for Web services which is anticipated to significantly change how people interact with applications and devices via the Web. The .NET technology "*enables the creating and use of XML-based applications, processes and websites as services that share information and functionality with each other on any platform or smart device"* (Microsoft, 2003b). The .NET is a comprehensive family of products including development tools, servers and smart clients. Essentially, it uses software for smart devices to enable PCs, laptops, smart phones, handheld computers, game consoles and other smart devices to operate "seamlessly" in an integrated framework. The .NET programming model allows a multi-device support (a wide range of smart devices), a multi-language support (e.g. VC++, VB, eVC, eVB, and C# interoperability) to enable easier and faster development of .NET applications.

The field data collection prototype is built using a standalone eVB development tool which is now integrated into the .NET framework. It is envisaged that mobile devices will become "smarter" while the number of mobile applications will increase and more dependent on Internet and Web services (e.g. real-time data collection). For reasons of interoperability between applications on different platforms, better help support and faster product delivery, software developers will be drawn towards

building mobile applications in an integrated environment such as the one provided by .NET technology.

# 2 Project Recommendations

The user interface features and functionality of the field data collection prototype can certainly be expanded using some of the new technologies described above. It will be interesting to see how these new technologies impact the usability of the field application. SVG, for example, can be used to design aspects of user interface that require graphical presentations with a very high level of details, particularly the cable network and topographic view of the UUC. Even with this new graphical representation format, user interface design fundamentals are likely to remain unchanged. But from the perspective of better performance, software maintenance and more advance functionality such as manipulation of graphics, SVG is a superior solution compared to standard GUI components (e.g. picture boxes, shapes and image controls).

XML can also be used to improve interoperability of the field data collection prototype with other Geospatial Information Systems (GIS) but not necessarily the usability aspect of the field application. For example, XML can replace flat files (.ini or .txt files) currently used by the prototype to exchange data. It is not really a major modification since XML is still text-based, but it is structured in a proper format (i.e. XML schema) that is read by the GIS database.

# References


Abowd, G., et al. (1997) "Cyberguide: a mobile context-aware tour guide", *Wireless Networks*, vol. 3, issue 5, pp 421-433.

Barbara, S., H. Laamannen, S. Poslad, and A. Zipf (2002) "Location-based mobile tourist services - first user experiences". INTERNET. http://deepmap01.villa-bosch.de/zipf/papers/CRUMPET-ENTER03-final.pdf, Cited 12 March 2003

Bellamy, R. et al. (2001) “Designing an E-Grocery Application for a Palm Computer: Usability and Interface Issues”, *IEEE Personal Communications*, vol. 8, no. 4, pp 60-64

Brewster, S. (2002) "Overcoming the lack of screen space in mobile computers", *Personal and Ubiquitous Computing,* vol. 6, issue 3, pp 188-205.

Brun-Cottan, F. and P. Wall (1995) "Using Video to re-present the user", *Communications of the ACM,* vol. 38, issue 5, pp 61-71.

Buchanan, G., et al. (2001) "Improving mobile internet usability", *Proceedings of the tenth international conference on World Wide Web*, ACM Press, New York, ISBN:1-58113-348-0, pp 673-680.

Caceres, R. et al. (2002) “Mobile Technology at Vindigo”, *IEEE Wireless Communications,* vol 9, No 1, pp 50-53

Cheverst, K., N. Davies, K. Mitchell, and A. Friday (2000) "Experiences of developing and deploying a context-aware tourist guide: the GUIDE project", *Proceedings of the sixth annual international conference on Mobile computing and networking*, ACM Press, New York, ISBN:1-58113-197-6, pp 20-31.

Cheverst, K., et al. (2001) "Using Context as a Crystal Ball: Rewards and Pitfalls", *Personal and Ubiquitous Computing.* ISSN:1617-4909, Vol 5, issue 1, pp 8-11.

Cheverst, K., K. Mitchell, N. Davies (2002) "Exploring Context-aware Information Push", *Personal and Ubiquitous Computing*, vol. 6, issue 4, pp 276-281

Cisco Systems (2000) "GPRS White Paper". INTERNET. http://www.cisco.com/warp/public/cc/so/neso/gprs/gprs_wp.pdf, Cited 21 January 2003.

Constantine, L. (1996) " Usage-Centered Design for Embedded Systems: Essential Models." *Embedded Systems Conference '96 Proc*. San Francisco: Miller Freeman.

Constantine, L. and A. Lockwood (1999a) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 24.

Constantine, L. and A. Lockwood (1999b) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 21.

Constantine, L. and A. Lockwood (1999c) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 103.

Constantine, L. and A. Lockwood (1999d) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 109.

Constantine, L. and A. Lockwood (1999e) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 125-129.

Constantine, L. and A. Lockwood (1999f) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 390-460.

Constantine, L. and A. Lockwood (1999g) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,*ACM Press, New York, ISBN: 0-201-92478-1, pp 300.

Constantine, L. (2000) "What Do Users Want? - Engineering Usability into Software". INTERNET. http://www.forUse.com. Cited 31 July 2003.

Cooper, A. (1995) *About Face: The Essentials of User Interface Design,* IDG Books Worldwide, Inc. pp 389.

Davies, N., K. Cheverst, A. Friday, K. Mitchell (2002) "Future wireless applications for a networked city: services for visitors and residents", *IEEE Wireless Communications*, vol. 9, no. 1, pp. 8-16.

Dey, A K., and G D. Abowd (2000) "Towards a Better Understanding of Context and Context-Awareness", *CHI 2000 Workshop on the What, Where, When and How of Context-Awareness*. April 3, Hague, Netherlands.

Dumas, J., and J. Redish (1999a) *A Practical Guide to Usability Testing,*Intellect Ltd, USA, ISBN: 1-84150-020-8. pp 30-37.

Dumas, J., and J. Redish (1999b) *A Practical Guide to Usability Testing,*Intellect Ltd, USA, ISBN: 1-84150-020-8. pp 310.

Dumas, J., and J. Redish (1999c) *A Practical Guide to Usability Testing,*Intellect Ltd, USA, ISBN: 1-84150-020-8. pp 386.

Dumas, J., and J. Redish (1999d) *A Practical Guide to Usability Testing,*Intellect Ltd, USA, ISBN: 1-84150-020-8. pp 43.

Dwolatzky, B., and R. van Olst (2003) "The Use of Location-Based Services In Maintaining Electrical and Other Networks In Developing Countries", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

eSVG (2003) "eSVG - embedded scalable vector graphics". INTERNET. http://www.embedding.net/eSVG/. Cited 12 November 2003.

Ferre, X., N. Juristo, H. Windl, and L. Constantine (2001) "Usability Basics for Software Developers", *IEEE Software*, vol 18, no 1, pp 22-29.

Fletcher, D. (2003) "Field Applications at DTE Using the Pocket PC", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

GIS Lounge (2003), "Mobile and Field GIS". INTERNET. http://gislounge.com/ll/mobilegis.html. Cited 27 March 2003.

Grant, K., (2003) "Sifting through emerging mobile technologies to strike gold for your wireless workforce", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

Griffin, B. (2003) "Mommy Do I Have To? Or Overcoming Resistance To The New System", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

Hackman, G. S., and D. W. Biers (1992) "Team usability testing: Are two heads better than one?", *Proceedings of the Human Factors Society 36th Annual Meeting*, pp1205-1209.

Hinckley, K., et al. (1998) "Two-handed virtual manipulation", *ACM Transactions on Computer-Human Interaction*, vol. 5, no. 3, pp 206-302.

Hinckley, K., et al. (2000) "Sensing techniques for mobile interaction", *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM Press, New York, ISBN:1-58113-212-3, pp 91-100.

Holland, S., D. Morse and H. Gendenryd (2002) "AudioGPS: Spatial Audio Navigation with a Minimal Attention Interface", *Personal and Ubiquitous Computing,* vol. 6, issue 4, pp 253-259.

Hughes, J. et al. (1995) "The Role of Ethnography in Interactive Systems Design" *Interactions*, April 1995, pp 57-63.

Indus International (2002) "Asset Management - Using mobile Computing to gain Competitive Advantage". INTERNET. http://www.cmmscity.com/articles/MobileComputing_wp.pdf. Cited 27 January 2003.

Interactive Business Systems, Inc. (2001) "Palm Pilot Data Collection for the Mobile Workforce". INTERNET. http://www.ibs.com/pdf/palm.pdf, Cited 27 January 2003.

Jacobson, I., G. Booch and J. Rambaugh (1999a) "The Unified Software Development Process", Addison-Wesley, ISBN: 0-201-57169-2, pp 421.

Jeffries, R.J, J. R Miller, C. Wharton and K.M. Uyeda (1991) "User interface evaluation in the real world: A comparison of four techniques. *Proceedings of CHI'91,*ACM, New York, pp. 119-124.

Karat, C., R. Campbell and T. Fiegel (1992) "Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation", *In Proceedings of CHI'92*, ACM Press, New York.

Karat, J., et al. (1996) "User centered design: quality or quackery?" *Conference companion on Human factors in computing systems: common ground*, ACM Press, ISBN:0-89791-832-0, pp 161-162.

Kistler, R. (2003) "Data: The Critical Investment", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

Kristoffersen, S., and F. Ljungberg (1999),"Making Place to make IT work: empirical explorations of HCI for mobile CSCW", *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, ACM Press, New York, ISBN:1-58113-065-1, pp 276-285.

Kuhl, L. (2003) "Doing More with Less: Leveraging Your Spatial Asset Data", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

Laarni, J. (2002) "Searching for optimal methods of presenting dynamic text on different types of screens" *Proceedings of the second Nordic conference on Human-computer interaction, ACM Press*, New York, ISBN:1-58113-616-1, pp 219-222.

Laerhoven, K., and K. Aidoo (2001) "Teaching Context to Applications", *Personal and Ubiquitous Computing*, ISSN:1617-4909, Vol 5, Issue 1, pp 46-49.

Lester, K. J. (2001) "Shaping South Africa's future with GIS - 1999 General Election Experience", INTERNET. http://hsrc.ac.za/gis/papers/SAgis/index.html. Cited 1 December 2002.

Martin, I. (2003) "XML - Gateway to Interoperability", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

McKee, F. (2002) "The importance of GIS in modern society", *GIM International,* June 2002, pp 68-71.

Microsoft (2003a) "Embedded visual tools downloads and samples". INTERNET. http://msdn.microsoft.com/vstudio/device/embedded/download.aspx. Cited 27 January 2003.

Microsoft (2003b) "What is .NET". INTERNET. http://www.microsoft.com/net/basics/faq.asp. Cited 23 August 2003.

Millen, D. R. (2000) "Rapid ethnography: time deepening strategies for HCI field research", *Conference proceedings on Designing interactive systems : processes, practices, methods, and techniques: processes, practices, methods, and techniques*, ACM Press, New York, ISBN:1-58113-219-0, pp 280-286.

Mobile Streams (2001) "Yes 2 3G - White Paper". INTERNET. http://www.mobilewhitepapers.com/pdf/3g.pdf, Cited 19 January 2003.

MTN (2003) "MTNdataLive Q&A", INTERNET. http://www.mtn.co.za/assistance/qsa/gprs.asp?from=S, Cited 19 January 2003.

Murray, J., D. Schell and C. Willis (1997) "User centered design in action: developing an intelligent agent application", *Proceedings of the 15th annual international conference on Computer documentation*, ACM Press, New York, ISBN: 0-89791-861-4, pp 181-188.

Nielsen, J. and R. Molich (1990) "Heuristic evaluation of user interfaces", *In Proceedings of CHI'90*, ACM Press, New York, pp 249-256.

Norman, D. A., and S. W. Draper (1986) *User-Centered Design*. Hillsdale, N.J.: Lawrence Erlbaum.

Otterbox (2003) "Armour PDA cases". INTERNET. http://www.otterbox.com/category.cfm?Category=28. Cited 25 May 2003.

Pascoe, J., N. Ryan and D. Morse (2000) "Using While Moving: HCI Issues in Fieldwork Environments", *ACM Transactions on Human Computer Interaction*, vol. 7, no. 3, pp. 417-437

Perry, M., K. Ohara, A. Sellen, B. Brown and R. Harper (2001) "Dealing with Mobility: Understanding Access Anytime, Anywhere", *ACM Transactions on Human Computer Interaction*, vol. 8, no. 4, pp 323-347.

Peters, K. (2003) "Avoiding Data De-Evolution", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

PocketSVG (2003) "PocketSVG - overview". INTERNET. http://www.pocketsvg.com/. Cited 12 November 2003.

Poupyrev, I. , S. Maruyama and J. Rekimoto (2002) "Ambient touch: designing tactile interfaces for handheld devices", *Proceedings of the 15th annual ACM symposium on User interface software and technology*, ACM Press, New York, ISBN:1-58113-488-6, pp 51-60.

Preece, J. (1994), *Human-Computer Interaction*, first edition, Addison-Wesley, pp 14.

Randell, B. (2003) "Implementing a Mobile GIS to Enhance Efficiency of Field Operations", *Proceedings of the 26th annual GITA Conference* [CD-ROM], San Antonio, USA.

Rosenberg, D. and K. Scott (2001a) "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, pp 1-20.

Rosenberg, D. and K. Scott (2001b) "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, pp 35.

Rosenberg, D. and K. Scott (2001c) "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, pp 59-60.

Rosenberg, D. and K. Scott (2001d) "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, ISBN: 0-201-73039-1, pp 86.

Rowley, D. E. (1994) "Usability Testing in the Field: bringing laboratory to the users", *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, ACM Press, New York, ISBN:0-89791-650-6, pp 252-257.

Satyanarayanan, M. (1996) "Fundamental Challenges in Mobile Computing", *Proceedings of Fifteenth annual ACM symposium on Principles of distributed computing*, ISBN:0-89791-800-2, ACM Press, New York, pp 1-7.

Schwabe, C. (2001) "African Renaissance: Towards the Development of a Spatial Information System for Socio-Economic Development in Africa", INTERNET. http://hsrc.ac.za/gis/papers/AfricanRenaissance/index.html. Cited 1 December 2002.

Shneiderman, B. (1998a) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd Edition, Addison Wesley Longman, ISBN: 0-201-69497-2, pp 74-76.

Shneiderman, B. (1998b) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd Edition, Addison Wesley Longman, ISBN: 0-201-69497-2, pp 67.

Shneiderman, B. (1998c) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd Edition, Addison Wesley Longman, ISBN: 0-201-69497-2, pp 267.

Smith, S. L. and J. N. Mosier (1986) "Guidelines for Designing User Interface Software". Report MTR-10090. The MITRE Corp, Bedford, MA.

Spriestersbach, A., et al. (2001) "Integrating context information into enterprise applications for mobile workforce - a case study", *Proceedings of the first international workshop on mobile commerce,* ACM Press, New York, ISBN:1-58113-376-6, pp 55-59.

Symbol (2003) "MC900 Series ruggedized handheld computer". INTERNET. http://www.symbol.com/products/mobile_computers/kb_mc9000_roll.html, Cited 17 May 2003.

Thomas, P. and R. Macredie (2002) "Introduction to The New Usability", *ACM Transactions on Computer-Human Interaction*, vol. 9, no. 2, pp 69-73.

Virzi, R. (1992) "Refining the test phase of usability evaluation: How many subjects is enough?", *Humans Factors*, ACM Press, pp 457-468.

van Vliet, H. (2000a) *Software Engineering Principles and Practice*, 2nd Edition, John Wiley and Sons, John Wiley, ISBN: 0-471-97508-7. pp. 210.

Vodacom (2003a) "Network Coverage Map", INTERNET. http://www.vodacom.co.za/contracts/coverage_map.asp, Cited 19 January 2003.

Vodacom (2003b) "Vodacom MyLife services", INTERNET. http://www.vodacom.co.za/my_life/, Cited 19 January 2003.

Vredenburg, K., J. Mao, P. Smith and T. Carey (2002) "A survey of user-centered design practice", *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, ACM Press, New York, ISBN:1-58113-453-3, pp 471-478.

W3C (2003) "SVG - XML for the web". INTERNET. http://www.w3.org/Graphics/SVG/. Cited 20 July 2003.

Waterson, S., J. Landay and T. Matthews (2002) "In the lab and out in the wild: remote web usability testing for mobile devices", CHI '02 extended abstracts on Human factors in computer systems, ACM Press, ISBN:1-58113-454-1, pp 796-797.

# Bibliography

Jordan P *An Introduction to Usability* Taylor & Francis Publishing, Feb 1999, ISBN 0748407626

Weiss S *Handheld Usability* John Wiley & Sons, 1rst edition, July 2002, ISBN 047084469

Hanttula D *Pocket PC Handbook* John Wiley & Sons, Feb 2001, ISBN 0764535684

Nielsen J and Mack R *Usability Inspection Methods* John Wiley & Sons, 1rst edition, April 1994, ISBN 0471018775

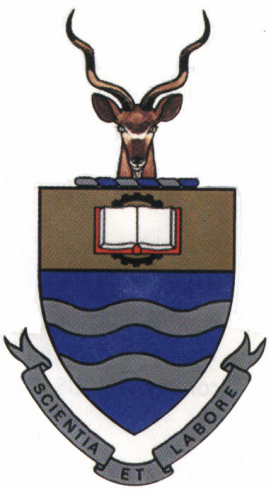
Balena F *Programming Microsoft Visual Basic.NET (Core Reference)* Microsoft Press, 1rst edition, April 2002, ISBN 0735613753

Halvorson M *Microsoft Visual Basic .NET Step by Step* Microsoft Press, Jan 2002, ISBN 0735613745

Harold E and Mearis W *XML in a Nutshell* O'Reilly & Associates, 2nd edition, June 2002, ISBN 0596002920

Walmsley P *Definitive XML Schema* Prentice Hall, 1rst edition, Dec 2001, ISBN 0130655678

# Appendix A




# Literature Survey

Doc. No. 1

# CONTENTS

CONTENTS ..... I

LIST OF FIGURES ..... III

1 SCOPE ..... 1

1.1 Introduction ..... 1

1.2 Purpose ..... 1

1.3 Audience ..... 1

2 MOBILE COMPUTING ..... 2

2.1 Overview ..... 2

2.2 Mobile Technology ..... 2
2.2.1 Mobile Devices ..... 2
2.2.2 Wireless Networks ..... 3
2.2.3 Mobile Application – some case studies ..... 5

2.3 Mobile System Characteristics ..... 8
2.3.1 Constraints ..... 8
2.3.2 Context-awareness ..... 9
2.3.3 Usability ..... 11

3 MOBILE APPLICATIONS FOR FIELD USE ..... 14

3.1 Field Data Collection ..... 14

3.2 Mobile GIS ..... 15

3.3 Mobile Work ..... 17

3.4 User Acceptance ..... 20

4 USABILITY DESIGN PROCESS ..... 21

4.1 Usability Design Guidelines ..... 21

4.2 User-Centered Design (UCD) ..... 27

4.3 Usage-Centered Design ..... 31

4.4 Usability Evaluation ..... 35

4.4.1 Heuristic Evaluation ..... 35
4.4.2 Empirical Usability Testing ..... 37

**5 CONCLUSION ..... 39**

# LIST OF FIGURES

*Figure 1: Asset Information Model in central data repository* _____ *14*
*Figure 2: An analysis of one task for an anesthesiologist. Further analysis would be necessary to understand subtasks like "take corrective action". (Source: Dumas J et al., 1999d)* _____ *29*
*Figure 3: A Usage-Centered Design Process (Source: Constantine L et al., 1999a)* _____ *35*
*Figure 4: Usability Test Laboratory (Source: Dumas J et al., 1999c)* _____ *37*

# 1 Scope

## 1.1 Introduction

This document presents a review of the literature pertinent to the project at hand, namely the development of a usable mobile application for field data collection. Included in this document are some background information on mobile computing (section 2), mobile application for field use (section 3) and usable software design process (section 4).

The section on mobile computing takes a broad look at current trends in mobile technology and their inherent characteristics. The section on field applications covers asset management, mobile Geospatial Information Systems (GIS), field data collection work and mobile workers. The various techniques in the software development including requirements gathering, user interface design and usability evaluation are discussed in the section on usability design process.

## 1.2 Purpose

The purpose of this document is to introduce the reader to background literature and other research works that are pertinent to this project. In doing so, it orientates the reader towards the problem at hand and sets the scene for subsequent investigation.

## 1.3 Audience

The intended readers of this document include researchers involved in the areas of software usability, mobile computing, as well as UI designers, developers, the external examiner and other interested parties.

# 2 Mobile Computing

## 2.1 Overview

The use of mobile devices (e.g. mobile phone, handheld computer etc.) and wireless networks has exploded in recent years. The availability of such a wide range of mobile devices and their increasing sophistication has become a challenge for designers of mobile applications and other interactive systems. This is because mobile computers have unique characteristics and fundamental differences from their desktop counterparts. Not only are there differences in the technology, but also in the way in which people *use* mobile applications. These important issues are discussed in the sections below.

## 2.2 Mobile Technology

### 2.2.1 Mobile Devices

At present, a wide variety of mobile devices with various form factors (full, half and quarter screen etc.) are available on the market. Their sophistication and capabilities in terms of screen resolution, size, weight, wireless communication (e.g. Bluetooth) and computational power have significantly increased in recent years. For example, mobile phones (1/8 screen) are used to run a wide range of applications from multimedia games to video conferencing. In addition, they come with add-on accessories such as GPS, IR scanners and digital cameras. Each type of mobile computer offers certain advantages over the other in terms of cost, portability, screen real estate and so on. In the end, the users, network accessibility and application requirements will determine the selection of the most suitable mobile device.

The trend in mobile computers is certainly moving towards smaller devices, particularly in the area of field data collection work (Interactive Business Systems,

2001). Recent studies have shown that the Personal Digital Assistance (PDA) is now more popular with field data collection personnel than traditionally used laptops (Fletcher D, 2003). The laptops are now considered too heavy and unwieldy to carry around in the field. The same sentiment is shared by tourists using mobile computers to navigate around cities in the developed countries (Barbara S *et al*., 2002). A tablet computer is also more portable than a traditional laptop but it becomes heavy when a rugged casing is used and often is significantly more expensive than a PDA. Some of the PDAs now come with GPRS, digital cameras and GPS which are particularly important for remote data collection. Despite these promising features, challenges over processor speed, memory limitations, and poor battery life still remain (see section 2.3 for details).

Three operating systems (OS) dominate current mobile computer market namely Palm OS, EPOC and Windows CE. The Palm Pilot PDA runs on the Palm OS, and maintains the majority of hand-held market share due to their simple and inexpensive design. The Window CE OS such as Pocket PC, on the other hand, offers more sophisticated graphics and multimedia capabilities. However, it is accused of merely replicating desktop software functionality on a small scale, and hence less preferred by some hand-held computer users. Unfortunately, a standard OS for mobile computers has not yet emerged and as a result many mobile applications across different OS are incompatible. This is one of the largest obstacles for creating stable mobile applications in the long run.

### 2.2.2 Wireless Networks

Mobile computers rely on wireless networks to share and access data between them and their static servers in “real time”. The two most common mobile networks are the wireless LAN (WLAN, also called 802.11a/b) and packet-based cellular network (Cisco Systems, 2000). The former is suitable for large content-based applications with high data throughput and fast Internet connectivity. They are typically used to provide Location Based Services (LBS) to visitors and residents alike in first world cities (Davies N *et al*., 2002). However, this network has relatively small coverage

area limited to where wireless access points (AP) are installed. The latter offers a wider coverage area since it operates on cellular network infrastructure but at much lower data throughput.

The next generation networks (NGN) such as 3G are expected to deliver larger bandwidth more reliably than the circuit switched networks available today (Mobile Streams, 2001). Unfortunately, 3G have being hyped extensively, and it has yet to be successfully implemented (with few exceptions, for example, NTT DoCoMo of Japan). Although the NGN has being slow in deployment, standards are emerging. It is likely that, in the future, all communications will be IP-based. In an IP-based paradigm, any gateway should be transparent, and thus implicitly supported by IP-compatible wireless applications. For instance, a mobile application with an application layer that is not dependent on the underlying wireless network is uniquely positioned to exploit advances in next-generation communications networks. Increased bandwidth from 3G will provide the infrastructure for much richer client experiences than is possible today.

The type of network employed depends not only on the geographic location or application requirements but also on the mode of connectivity required between the mobile client and the static server. Three modes of connectivity are as follows:

- *Always Connected* – In this mode, mobile users can be wired or wireless, using any mobile device that enables two-way, real-time communication with the system. Instant access to the system information is enabled.

- *Disconnected* – When disconnected, mobile users interact with a handheld device where data is stored locally, and periodically through synchronization the data is transferred from the device to the server, and updates made by other users are received from the server.

- *Intermittently Connected* – When the network, telecommunications, or Internet connection goes down, data can be stored locally. Dynamic

reconnect capabilities bring the device back online when the connection is restored, and stored data is then synchronized between the portable device and the server.

Many power and telecommunications utilities have been utilizing wireless networks for many years, building and maintaining costly private networks (Grant K., 2003). The public networks are much less costly because individual customers do not have to pay for towers and specialized equipment. In South Africa, for example, packet-switched public cellular network is an attractive option because the country already has three cellular networks providers (i.e. Vodacom™, MTN™ and Cell C™) covering a large percentage of the country. There are, however, concerns over the reliability, quality of service (QoS), mainly because the 2.5G technology is still in relatively new. Vodacom™, South Africa's largest cellular provider, for example, introduced GPRS technology only in late 2001 called "*Vodacom's MyLife*" (Vodacom, 2003b) and a similar service by MTN™ called "*MTNdataLive"* (MTN, 2003).

### 2.2.3 Mobile Application – some case studies

This section of the document entails case studies of two existing mobile applications utilized in areas of e-commerce and Location Based Services (LBS) in order to discuss some to the usability features incorporated into their designs. The first example is a handheld/PDA shopping solution called Easi-Order™ from Safeway UK (Bellamy R. *et al*, 2001). It was the first instance of an already expanding class of e-commerce applications that combine properties of pervasive technology and personalization.

Easi-Order™ allows customers to place orders from home and later pick up their orders. The shopping process begins with a customer creating a personalized shopping list. Safeway UK has a unique way to generate this list directly from logs of till transactions. The customer transactions and product information are tracked and stored in the database. The customer is given access to their personalized shopping list and other lists such as items on special offer, recommended products and so on.

Once an order has been created, the customer attaches his PDA to a modem and sends information to the store. Bellamy's (2002) paper discusses the design of Easi-Order application and they are summarized as follow:

1. *Hardware and software limitations*: Small screen (160 × 160 pixel) of PDA makes it hard to present much information (11 lines of text with 30 characters per line) on the display. The Palm Pilot OS has a limited set of UI components: icons, tables, scrollable lists, dialog boxes and buttons. It becomes necessary to augment these components to provide satisfactory user experience.

2. *Research Methodology*: The designers met face to face with intended users namely their customers and observe their shopping habits. They made two field trips to the store and spoke with a dozen customers spanned across a number of demographic characteristics (age, gender, work in/out of home etc.). Talking with customers profoundly influenced their thinking and changed many of their earlier assumptions about the application. The design team completed their prototype and went back to the customers over a period of 3 days to carry out the evaluation process. The designers found a few shortcomings with their initial design but they were corrected later in their redesign in an iterative process.

3. *Design features and user experience*: The customers were confused with use of some of the GUI components, particularly use of navigation tabs. However, the designers found that people were able to overcome this problem if they are first given a brief explanation using a list of simple instruction steps and annotated pictures of the screen. They also found that use of familiar themes such as a basket metaphor (commonly used by shopping websites) for the Order screen helps reduced the learning curve. Like many mobile applications Easi-Order coordinates audio and visual feedback mechanisms to enhance user interaction. For instance, when an item is added to the order, a distinct icon on the order tab flashes

accompanied by a soft, non-intrusive beep. This design feature was intended to provide immediate perceptual feedback that item has been successfully added to the order (thus, putting user's mind at ease) and of attracting the user's attention to the Order tab and enticing him to view the Order screen.

The second case study focuses on a handheld (PDA and WAP phone) tour guide system from Vindigo (Caceres R. *et al*, 2002). The software provides tour information in areas of dining, shopping and entertainment based on contextual information such as time of day, personal preferences and most importantly, the geographical location of the user. The user specifies this automatically by reading latitude and longitude coordinates from GPS receiver attached to mobile device, and manually by selecting from a predefined list of street intersections. Automatic location is convenient in many situations but manual location allows user to plan for a place where he will be in future. The application offers two main functions implemented as the "Go" and "Map" screens. The screens provide walking and public transportation directions and interactive graphical map respectively. By default, the map displays the locations of the user and of selected points of interest but very little else to avoid clutter on the screen. Street names and other details are labeled only when user taps on them. Maps can also highlight the route corresponding to the text directions shown in the "Go" screen. The system has been successfully deployed over 20 major cities including New York, San Francisco, and London. Caceres (2002) highlighted constraints of mobile devices such as processor speed, screen size, power, network connectivity and throughput. In the face of these considerations he recommends a number of design principles for mobile applications like Vindigo to provide a high-quality user experience. They must:

1. Offer intuitive and responsive user interface. Simplicity must be the key to UI design.
2. Be customized to the form factors and capabilities of target devices and wireless networks.

3. Exploit local processing and storage resources while making judicious use of wireless networks.
4. Make intelligent use of context: location, time, schedules and personal preferences etc.
5. Provide deep relevant content with minimal user interaction.

Mobile applications pose a considerable design challenge for software developers. They have different user profile, target platforms and operating environment, and hence different software requirements to the traditional desktop and web applications. It is clear from above examples that they must be carefully tailored to meet these demands if they are to be successful. Each type of mobile application has its own unique characteristics and designers must be able to identify and better still exploit them to produce most usable software possible. The next section of the document discuss in details these fundamental characteristics of mobile systems and more importantly, the design processes and methodologies used to achieve usable software applications.

## 2.3 Mobile System Characteristics

### 2.3.1 Constraints

Mobile computing is characterized by a number of constraints that are intrinsic to mobility (Satyanarayanan M., 1996). Together, they complicate the design of mobile applications that are to a large extent different from traditional information systems. The four constraints are as follows:

- ***Mobile devices are resource-poor****:* They are limited in terms of battery life, screen size, memory, processor speed etc. This is the price they pay for being small and lightweight. Unfortunately, mobile computers will always remain resource-poor compared to their desktop counterparts.

- ***Mobility is inherently hazardous****:* Mobile users operate in an environment that is hostile and hazardous. They are often exposed to environmental elements that constantly influence the usage of mobile device. For example, a touch screen of PDA should be equally readable in direct sunlight as in the dark. In a field data collection work, mobile devices are prone to physical damage from an occasional dropping to exposure to water, high frequency magnetic interference etc.

- ***Mobile connectivity is highly variable in performance and reliability***: A wireless networks access behaves differently in terms of bandwidth and reliability for the indoor and outdoor environments. A GPS, for example, is unreliable indoors and in built-up areas due to not able to "see" a sufficient number of satellites to locate an accurate position. Mobile devices must rely on cache to store information and have their functionality remain available throughout periods of disconnection.

### 2.3.2 Context-awareness

When humans talk with humans, they are able to use implicit situational information, or *context*, to increase the conversational bandwidth. Similarly, it is envisaged that the richness of human-computer interaction (HCI), particularly in mobile computing, will be increased by improving computer's access to contextual information and appropriately adapting to the changes in the context.

The notion of context, however, is very broad and not always clearly understood. There are a number of definitions on what context really means. A survey (Dey A *et al*., 2000) of definitions and categories of context includes location, nearby objects, user environment, time of day, temperature and so on. However, Dey (2000) argues that the definition of context should not be specific. Instead, one should take into consideration the whole situation relevant to an application and its set of users. Of course, one cannot enumerate which aspects of all situations are important, as this

will change from situation to situation. In some cases, the physical environment may be important, while in others it may be completely immaterial.

In mobile tour guides, for example, contextual information that is relevant includes personal and environmental (Cheverst K *et al.*, 2000) (Abowd G *et al.*, 1997). Personal context may be the visitor's interest, current location and personal preferences on reading language, food, music etc. Examples of environmental context may include the time of the day and opening times of attractions, and so on.

Dey defines context as follows:

*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and an application, including the user and applications themselves.*

The definition of *context-awareness* is also unclear. Dey has found that many researchers before him defined context-aware applications to be applications that dynamically change or adapt their behavior based on the context of the application and the user. He, on the other hand, defines context-awareness as follows:

*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*

This definition of context-awareness is more acceptable because it allows context to be either explicitly or implicitly indicated by the user. This is important since one of the major challenges in development of context-aware mobile application is how contextual information should be presented (Cheverst K *et al.*, 2002). Some researchers argue that the more information that can be automatically captured and turned into context, the better (Abowd G *et al.*, 1997). If the user has to explicitly inform the system about the context information, then the context is unlikely to be fully utilized. However, pre-empting the user's goal is often difficult and certain tradeoffs are needed. This is because users are required to trust the behavior of the

system's intelligence and this requires the system to have predictable behavior and the ability to successfully and consistently preempt the user's goal. Agents may incorrectly preempt the user's goal, owing to either flawed intelligence or to incorrect or out-of-date contextual information. In such circumstances, the user is likely to get frustrated because the system will either appear overly prescriptive or, worse still, present incorrect results (Cheverst K *et al*., 2001). The lessons learnt from the mobile tour guides suggest that designers of context-aware systems must not be over zealous when deciding to constrain the information or functionality provided by the system based on the current context (Cheverst K *et al*., 2000). The solution, therefore, is to provide the user with the choice so that the system does not behave in an overly authoritarian manner.

A number of researchers have also investigated sensing techniques and experimented with a wide variety of sensors that is used to detect and input relevant contextual information into mobile device (Hinckley K *et al*., 2000) (Laerhoven K *et al*., 2001). Sensors can be used to detect environmental conditions, gestures and activity to geospatial location of the user. For example, the radio frequency (RF) and infra-red (IR) sensors have being used outdoors and inside buildings respectively (Abowd G et al., 1997). The type of sensors used on a mobile device such as PDA includes microphones, light sensors, and accelerometers (for measuring movement), pressure and touch sensors etc.

### 2.3.3 Usability

While much of the underlying technology is available in handheld and mobile computing, there are challenges with respect to usability of mobile devices. Unfortunately, usability is still one of the biggest barriers for a broad usage of mobile devices. As a result, users are less productive, commit errors, and are dissatisfied with the software.

According to ISO 9214, Part 11, usability is "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and*

*satisfaction in a specified context of use*". Contrary to what some might think, usability is not just the appearance of the user interface (UI), more importantly, it relates to "*how the system interacts with the user"* (Preece J, 1994). Usability constitutes five basic attributes (Ferre X *et al*., 2001):

- ***Learnability* –** How easy it is to learn the main system functionality and gain system familiarity and proficiency to complete the job. This attribute is usually assessed by measuring the time the user spends working with the system before that he/she can complete certain tasks in the time it would take an expert to complete the same tasks. This situation is particularly important to novice users.

- ***Efficiency*** – Efficiency is measured in terms of the number of tasks per unit of time that the user can perform using the system. The higher the system usability is, the faster the user can perform the task and complete the job.

- ***User retention over time*** – It is critical for intermittent users to be able to use the system without having to climb the learning curve again. This attribute reflects how well the user remembers how the system works after a period of non-usage.

- ***Error rate*** – This attribute contributes negatively to usability. It does not refer to system errors. On the contrary, it addresses the number of errors the user makes while performing a task. Good usability implies a low error rate. Errors reduce efficiency and user satisfaction, and they can be seen as a failure to communicate the right way of doing things to the user.

- ***Satisfaction* –** This attribute shows user's subjective impression of the system.

Unfortunately, usability of small devices is significantly poor compared to a well-

equipped desktop computer. The popular reaction to the most widely available mobile services has been negative. In particular, the uptake by customers in many countries for new WAP phones has being disappointing (Buchanan G *et al*., 2001). Their experience suggests that one of the reasons for failure of WAP to date, and indeed the potential failure of similar technologies, is that not enough time has been spent really thinking about the human factors.

As more users enter the world of mobile computing, user interface designers are challenged by moving away from large screens and familiar input devices of the desktop computer, to small, pocket-sized screens and limited interaction techniques of mobile devices, such as PDAs and mobile phones. Since the screen of a mobile device is relatively small it can become cluttered with information as designers try to cram on as much information as possible. This has resulted in devices that are hard to use, with small text that is hard to read, cramped graphics and little contextual information. One of the solutions is to use sound to present information (Brewster S, 2002). This reduces clutter on the screen and allows more information to be presented on the display. Sound can also be used to improve targeting of widgets on the screen and therefore improve the usability of different selection strategies. For example, sound can be used to indicate when the stylus is on or off a target or when it is nearing a target on the interface. The audio feedback mechanism is also being used to enable people with impaired hearing to navigate (Holland S *et al*., 2002). The intensity and the number of pulses of sound are used to indicate the direction and the distance of target respectively. Other channels of communication such as tactile feedback is also proved to be useful (Poupyrev I *et al*., 2002). The same study found that 22% faster task completion when handheld device is enhanced with tactile feedback. Some researchers (Laarni J, 2002) have also discovered that reading speed and comprehension of information on a mobile device can be increased by using appropriate presentation techniques depending on a particular type of screen.

# 3 Mobile Applications for Field Use

## 3.1 Field Data Collection

To be proficient and competitive in a global market means companies must properly manage their infrastructure, workforce and other resources. Asset management enables these companies to operate at an optimum efficiency by allowing better management decisions based on accurate information about company's assets. One such case is in the utility industry (e.g. water, electricity and telecommunications) where records of their vast infrastructure of network assets distributed over a wide geographical area are captured and kept up-to-date. Subsequently, all of this data is used to manage planning in asset purchase, maintenance, and overall performance of the network.

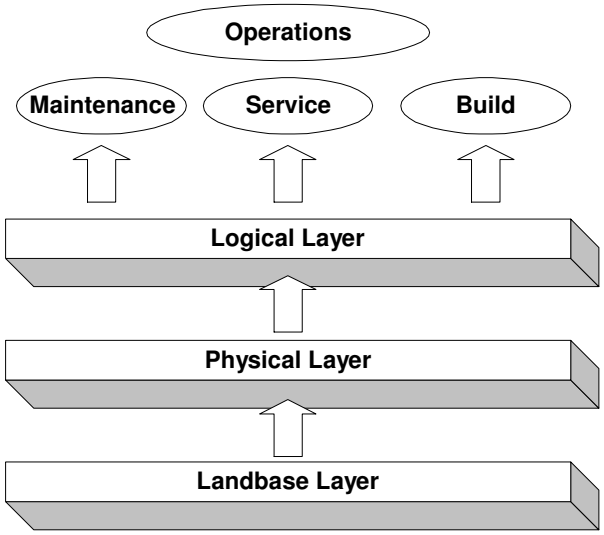



**Figure 1: Asset Information Model in central data repository**

In order to bring about a successful asset management program, public utilities need to capture and store accurate and comprehensive network related data within a central repository. According to their location, technical specifications and logical configurations, network asset information may be represented on "layers" that are interdependent (see Figure 1). In many underdeveloped countries, including South

Africa, utilities do not yet have complete information about their networks, and they are in the process of data collection and verification.

At present, appropriate techniques and processes for capturing and validating data are being explored so that the data collected is accurate and complete. Kuhl (2003) outlines three different approaches to field data collection in the utilities as well as their advantages and disadvantages. The first approach is to do a complete field audit (Kuhl L, 2003). This approach is the most effective way to gain an accurate representation of network assets as they exist in the field. Though labour-intensive, it can also be the most cost-effective approach. The second approach is to detect errors in the target system and go back to the respective sources in an iterative manner and make corrections. Unfortunately, this approach is very labour intensive and does not always produce the best results. The third approach involves the implementation of advanced data gateways to limit the amount of field verification required. Advanced data gateways detect errors, automatically correct errors, provide reports on what was fixed, and provide the tools for highlighting and fixing the remaining errors that cannot be automatically corrected. In the end, Kuhl (2003) suggested that the best approach is a combination of complete field audit and use of advanced data gateways.

## 3.2 Mobile GIS

The spatial information captured by the Geospatial Information Systems (GIS), when analyzed provides answers to complex problems which may otherwise be difficult to solve. A GIS with its ability to link and display different data sets on the basis of a common geography appears to be the perfect set of tools for supporting a decision making process. For instance, in 1996 South African national census saw the capture of spatial information about the country’s population into GIS. Since then, it has been identified for use in South Africa's integrated rural development initiative and provision of universal access to the country’s resources and infrastructure (Lester K, 2001). Although spatial information has been successfully used in the

environmental and mineral exploration fields for decades, it was only recently that GIS began to be used more and more in the socio-economic environment.

Undoubtedly, geospatial information has come to be an essential component of problem solving in modern society. The European Commission sets the goal for Europe to become “the most competitive and dynamic economy in the world” with an emphasis on the exploitation of public-sector information. Seven conditions have been identified in order to achieve this goal, five of which directly involve GIS (McKee F, 2002). The importance placed on GIS by the Commission indicates how critical it is to the future socio-economic and political growth of Europe. In South Africa and Africa as a whole, focus on GIS is established by the NEPAD which aims to lift socio-economic conditions of the people by addressing poverty and underdevelopment on the continent (Schwabe C, 2001).

For NEPAD to succeed, provision and maintenance of services and infrastructure (e.g. water, electricity, roads etc), and communication technology (e.g. radio, telephone, cellular networks) are needed. NEPAD requires spatial information to be collected in the immediate short term so that especially socio-economic and political information can be linked to administrative boundaries at a sub-national level (Schwabe C, 2001). Public municipalities and utilities play an important role in providing necessary spatial information about the country’s infrastructure and resources. The use of GIS in the public utilities environment is crucial in automating production of plans, manage services, infrastructure, and subsequently creating comprehensive information system at both national and sub-national level. This information provides an understanding of what proportion of the population has access to basic services and infrastructure, and address inequalities that may exist, as well as encourage investment by private sector.

A typical utility makes countless changes and updates to its network every day and they are depicted on maps and stored on paper. However, many of these paper-based workflows are inefficient. For example, data collected from the field using a paper sourcing form must be manually entered into a GIS database (called data

capturing process). This process is very time consuming, labour intensive, and prone to human errors. As a result, utilities are providing fieldworkers with mobile computers and mobile GIS software for field data collection and verification. It is envisaged that use of mobile applications in asset maintenance will allow instant access to up-to-date service related information. For example, work orders can be downloaded directly to a technician's wireless device and work instructions and asset history can be viewed on site (Indus International, 2002). In addition, emergency service can be requested immediately for an unexpected outage. With small, lightweight and inexpensive mobile devices as well as the increased coverage of wireless data networks, mobile solutions have become applicable for a broad usage by the mobile workforce. There are a number of mobile GIS mapping solutions (e.g. ArcPad, Intelliwhere and PocketGIS etc.) commercially available for asset data collection and management (GPS World, 2002). Some of the well-known vendors include ESRI, Intergraph and Trimble who are making significant headway into the development of mobile GIS technologies.

## 3.3 Mobile Work

The context in which mobile computers are used is very different from a traditional office setting. A mobile environment is heterogeneous (Hinckley K *et al*., 2000), where users often face several different kinds of use situations that are constantly changing. A commuter, for example, may have to stand up the first part of ride of town, and sit down the last. While sitting down in front of a table, a palm-top computer with external keyboard may be very suitable. While standing up, however, it may be very difficult to use. The diversity of the mobile use context calls for a set of interaction styles among which a mobile user can choose from in the particular situation at hand (Hinckley K *et al*., 2000). Similarly, a fieldworker's work context is a dynamic one, where a mobile computer will be utilized throughout the course of user's work, often spread over a wide geographic area. One of the characteristic difficulties of mobile work is that there is less predictability and more restricted access to information than a traditional office work (Perry M *et al*., 2001). In the study of mobile work in two settings, namely, telecommunication service engineers and

maritime consulting staff, four important features of mobile work were identified (Kristoffersen S *et al*., 1999):

- Tasks external to operating mobile computers are the most important, as opposed to tasks taking place "inside the computer" (e.g. a spreadsheet for an office worker)

- User's hands are often used to manipulate physical objects, as opposed to the user in traditional office setting, whose hands are safely and ergonomically placed on the keyboard.

- Users may be involved in tasks ("outside the computer") that demand a high level of visual attention (e.g. to avoid danger as well as monitor progress), as opposed to a traditional office setting where a large degree of visual attention is usually directed at the computer.

- Users may be highly mobile *during* the task, as opposed to in the office, where *doing* and *typing* are often separated.

In a similar study, Pascoe (2000) studied a fieldwork carried out by a game ranger, tracking and recording movements of animals in a Kenyan game reserve. He also found his fieldwork exhibit similar characteristics identified by Kristoffersen (1999). He identified three more characteristics of fieldwork and they are:

- *Dynamic User Configuration* – Fieldworkers will want to collect data whenever and wherever they like. They may be doing this work while they are standing, crawling, climbing or walking.

- *High Speed Interaction* – The subjects of some time-dependent observations are highly animated or, more commonly, have intense periods or "spurts" of activity. During these spurts of activity, they need to able to enter high volumes of data very quickly and accurately, or it may be lost forever.

- *Context Dependency* – A fieldworker's activities are intimately associated with their context or, if different, the subject's context.

The authors of both empirical studies argued that the current direct manipulation interaction style adopted in most mobile devices is unsuitable for the mobile work context. In particular, a high degree of visual attention that direct manipulation demands. To overcome some of the perceived drawbacks of the Graphical User Interface (GUI) of mobile computers, Kristoffersen (1999) and Pascoe (2000) proposed alternative user interaction styles called MOTILE and Minimum Attention User Interface (MAUI) respectively. The objective of both interaction styles is to transfer interactions to modes that take less of the user's attention away from their current task. MOTILE is based on three principles:

- *Little or no visual attention* – the "executing actions" should not demand a high degree of visual attention

- *Structured, tactile input* – "perceiving the systems state" should demand little or no visual attention. One important reason is the practical problem of finding a place for mobile computer that makes the screen easily available for the user. This calls for feedback and output methods that demand little or no video. Such methods may rely on audio, which is the third implication.

- *Use of audio feedback* - in most mobile situations they studied users could have relied on audio feedback. Even in extreme environments, this would be possible, because an "ear-plug" may be used.

To a similar effect, Pascoe proposed two principals for user interface design for fieldworkers. Firstly, through an appropriate utilization of interaction modes, he argues that the user interface can be designed to draw minimum amount of attention away from the user's activity. And secondly, to automatically record and filter contextual information using knowledge of user's current environment.

## 3.4 User Acceptance

IT projects are often proposed, developed and implemented without much regard for the end users. Recent literature suggests that issues over user acceptance remain a concern for successful deployment of mobile technology (Randell B, 2003). To a large extent, a project's success depends on their acceptance and usage. This is particular true for fieldworkers who are the key to data collection and maintenance process as they provide the source information for the updates that are to take place in the GIS.

It is important to recognize and respond to the user resistance that comes with new technologies (Griffin B, 2003). Unfortunately, user training alone is not enough. One preconceived notion about software is that users can always be trained to put up with the usability drawbacks of the system. All too often, users are trained to use awkward workarounds that mask usability inadequacies. Making too many assumptions about users' expectations and levels of competence can get software developers into a lot of trouble. In South Africa, and in other developing countries, fieldworkers have a lower level of computer experience and skill than their counterparts in developed countries (Dwolatzky B *et al.*, 2003). Hence, one cannot assume that the intended users are familiar with certain operating systems and metaphors that may be associated with them.

# 4 Usability Design Process

## 4.1 Usability Design Guidelines

Basic usability design guidelines help designers make reasonable decisions that lead to highly usable systems. These guidelines contribute to the goals of designing systems that are easy to learn, but efficient, reliable and satisfying to use (these etc usability attributes are discussed in section 2.3.3). The guidelines include both usability rules and design principles. The “rules” of usability define a general character of what constitutes well-designed, usable systems. These rules provide broad, overall directions for user interface (UI) design, pointing the designer towards a generally superior solution. The design principles, on the other hand, provide a narrower guidance on more specific issues in the software industry.

Strictly speaking, all of these, rules and principles alike, are mere heuristics. That is, they are simple rules of thumb that generally point the way to better designs but without any general guarantee of good results. Heuristics will not resolve design issues on their own and are no substitute for thoughtful analysis or inspired creativity on the part of the designer. The best rules can be applied ignorantly or indiscriminately. A number of usability and HCI experts have proposed many software usability principles and some of them extend to hundreds of pages. Jakob Nielsen (1990), one of the more well-known usability experts, summarized them into nine fairly broad heuristics or guidelines:

- Simple and natural dialogue
- Speak the user’s language
- Minimize user memory load
- Be consistent
- Provide feedback
- Provide clearly marked exits
- Provide shortcuts

- Provide good error messages
- Prevent errors

Shneiderman (1998a), another well-known user interface design expert proposes eight “golden rules of user interface design” which are to a large extent comparable to what Nielsen proposed. They are:

- Strive for consistency
- Enable frequent users to use shortcuts
- Offer informative feedback
- Design dialogs to yield closure
- Offer error prevention and simple error handling
- Permit easy reversal of actions
- Support internal locus of control
- Reduce short-term memory load

From a usable software development approach, Constantine (1999a) proposed five “rules” of usability and six design principles for usable user interface. They are as follows:

First rule: **Access**

*The system should be usable, without help or instruction, by the user who has knowledge and experience in the application domain but no prior experience with the system.*

Ideally, a well-designed system should enable user who knows how to perform some task to accomplish using it without consulting even an on-line help system. Highly accessible interfaces are sometimes referred to as *intuitive*, meaning that the guesses and presuppositions of users are more likely to be right than wrong and that, even when wrong, the results are reasonable responses from the system that are readily understood by the users. A well-designed system needs to support all it

users, from first time users to those who have extensive experience in working with the system (power users).

Second rule: **Efficacy**

*The system should not interfere with or impede efficient use by a skilled user who has substantial experience with the system.*

In many cases, the very design practices that make things easiest for beginners make things harder for everyone else. Designers will find it very challenging to serve both groups of users who are just learning a system and those that have already learned it well.

Third rule: **Progression**

*The system should facilitate continuous advancement in knowledge, skill, and facility and accommodate progressive change in usage as the user gains experience with the system.*

The Progression Rule means that some connection is needed between the simpler and more advanced facilities. In other words, organization and details of user interface should actively aid the user in understanding and using additional features while acquiring new skills. The rule also suggests that the various features and facilities supporting newcomers, old hands, and everyone in between need to be integrated and meaningfully related. The designers need to understand how the patterns of usage change as users progress in knowledge and skill. Things that are encouraging and comforting to beginners can become annoying impediments for advanced users; facilities that advanced users employ with confidence may be daunting or even frightening for those just starting out. Yet, all these things need to be smoothly integrated into a seamless interface.

Fourth rule: **Support**

*The system should support the real work that users are trying to accomplish by making it easier, simpler, faster, or more fun or by making new things possible.*

The support rule is at the very heart of usage-centered design (see section 4.3 for details). It is based on the notion that all software systems are tools and good tools support work (as oppose to mere activity). It exhorts designers to understand what users will need to do with a system and how they will need to do it in order to perform their work better. Every software engineering effort should be seen as an opportunity for reengineering the work itself. Supporting the real work that users are trying to accomplish may mean changing that work.

Fifth rule: **Context**

*The system should be suited to the real conditions and actual environment of the operational context within which it will be deployed and used.*

This rule emphasizes that the best design intentions are inadequate if they do not take into account where and under what circumstances systems will actually be used. Every context is different. Operational context needs to be a background concern throughout much of the design process in order to produce systems well suited to the context in which they will be used.

The five rules of usability described above do not in themselves offer designers enough in the way of specific direction when it comes to resolving practical problems in UI design. For that, Constantine (1999a) proposed more focused design principles. Each principle incorporates a number of closely related, more detailed considerations within a general class of issues. These design principles are:

**Structure Principle:**

*Organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another.*

This principle is concerned with the overall user interface architecture and directly reflects the notion of user interface design as a dialogue between designers and users. Good user interfaces are deliberately organized in ways that reflect the structure of the work being supported and the way in which users think about that work. All too often, especially using modern visual development tools, the placement of visual components within forms or dialogues and their distribution among these is almost haphazardous.

**Simplicity Principle:**

*Make simple, common tasks simple to do, communicating clearly and simply in the user's own language and providing good shortcuts that are meaningfully related to longer procedures.*

Naturally, all designers would like to produce user interfaces that are simple and easy to use. But neither everything about a user interface nor every task to be performed can be made simple. Design is always a matter of trade-offs.

**Visibility Principle**

*Keep all needed options and materials for a given task visible without distracting the user with extraneous of redundant information.*

This principle is about designing UI that make things visible and available to users

based on what they are trying to accomplish. The goal is to go beyond What You See Is What You Get (WYSIWYG) to What You See Is What You Need (WYSIWYN). On the one hand, the design object is to make all the necessary and relevant options visible and explicit. On the other hand, good designs do not overwhelm users with too many alternatives or confuse them with irrelevant information.

**Feedback Principle**

*Keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.*

Good user interfaces are good conversationalists, telling the user what is happening inside the system. The Feedback Principle tells designers some of the rules of this conversation. A message that is not seen or heard communicates nothing, so part of successful feedback is to present information in such a way that it is noticed, read, and interpreted correctly.

**Tolerance Principle**

*Be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonably.*

Fewer errors are better than good error messages. Not only does it accept varied input and actions from users, but, when something unexpected arrives, it also does not punish them. Good software programs interpret reasonably any reasonable action by the user.

**Reuse Principle**

*Reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.*

A user interface is more predictable and understandable to users when they see the consistency and predictability with UI components, their arrangements, functionality etc.

## 4.2 User-Centered Design (UCD)

"*Computer programming and software development did not always have a concern for users or a focus on the usability of systems. Although software designed in a technically-oriented manner performs processing adequately, the code that is produced rarely meets human needs. Looking back to the 1950s and 1960s, when modern business and scientific computing began to come into its own as an industry and a profession, users – that is, the ultimate end users of the results of computations – did not typically get anywhere near computers. The machines – large, expensive, and often more than a little bit temperamental – were attended like electronic idols by duly anointed operators and fed their programs and data by properly initiated programmers. Only the operators, the service technicians, and a few select others actually flipped switches or pressed buttons on the control console of one of those sluggish giants. The users of information were handed a report or a table of numbers and considered themselves lucky if the columns were formatted so that the numbers could be easily read. Now, the user population for office automation, home and personal computing, and digital libraries is so vastly different from the original that programmers' intuitions are inappropriate. Current users are not dedicated to the technology, their background is more tied to workflow, and their use of computers is discretionary. In the best designs, the techno-centric style of the past*

*is yielding to a genuine desire to accommodate to the users' skills, goals, and preferences.*" (Constantine L *et al.*, 1999a)

The User-Centered Design (UCD) is a user-centric software design methodology that focuses on users and their needs in an effort to design usable software. The UCD had its origins with the seminal work of Norman and Draper (1986) and its rise represented a gradual shift from a focus on technology to a focus on people (users). The UCD places people at the very heart of the system design process. It has become the dominant force in UI design for software in recent decades. The UCD is a practice of the following principles, *the active involvement of users for a clear understanding of user and task requirements, iterative design and evaluation and a multi-disciplinary approach* (Vredenburg K *et al.*, 2002). Two key components of UCD are the User Analysis and Task Analysis.

**User Analysis** – this stage of UCD is critical for understanding user groups for which the documentation is intended. The deliverables from this stage is an audience definition that describes skills and abilities of the target users and job characteristics of the user group. A user profile also includes the willingness and preparation for learning, and expectations and interests (Murray J *et al.*, 1997).

In order to determine user profile users are broken into multiple user segments. Each segment has a profile that answers questions about user demographics, psychographics, skills, knowledge, and background. They are obtained through interviews, surveys, focus groups, observation and reviews of other information such as industry reports, press or marketing materials. In addition, it is important to analyze user environments. For example, where do users perform their tasks? The conditions that exist in the working environment that impact on how users do their work includes information about physical work environment (lighting, noise, space, temperature, telephones, people), user location (work at home, on site, mobile), and human factors (vision, hearing, keyboard abilities, sitting versus standing). User requirements are gathered by asking "*What do users expect the application to do for them?*" and matching requirements to tasks. That is, reviewing user tasks and

requirements is a reality check to make sure the requirements are in line with the tasks that the users are trying to do.

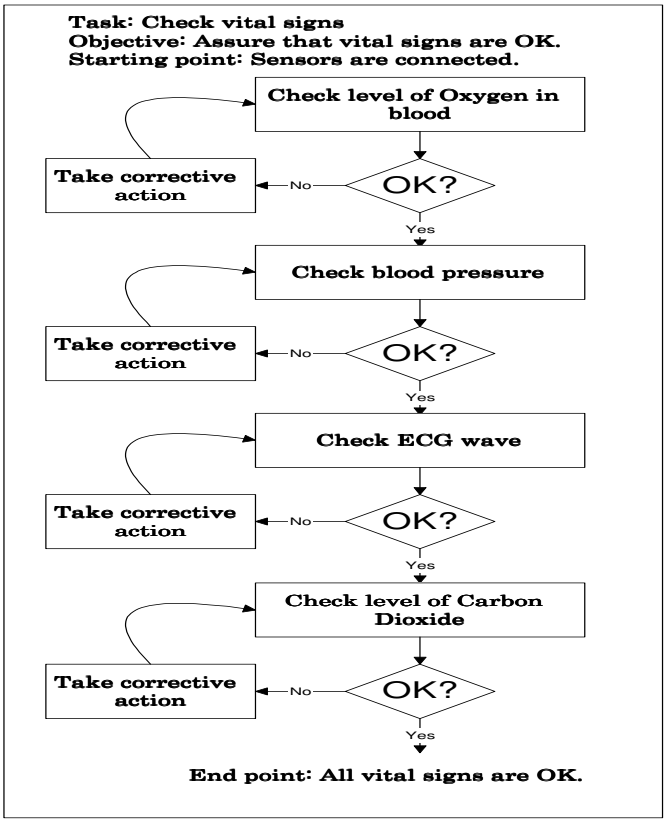



**Figure 2: An analysis of one task for an anesthesiologist. Further analysis would be necessary to understand subtasks like "take corrective action". (Source: Dumas J *et al.*, 1999d)**

**Task Analysis** – the main goal of a user interface is to support the user's tasks. In order to ensure that a UI appropriately supports tasks, a designer must understand how people actually work. A task analysis is really a method for determining this. Many task analysis techniques exist, but few are simple to understand and use. Most are based on abstract concepts such as formulas or diagrams (see example in Figure 2). As a result, require substantial documentation that users will neither read nor understand. Another problem appears to be involving users in a traditional task analysis process. One deliverable from this stage of UCD is a complete list of current tasks and future tasks, including scenario modeling. Tasks and scenarios are used in the design of all user information; for example, planning, diagnosis, marketing,

tutorial, and help information (Murray J, 1997). The following questions are addressed during the task analysis process:

- What tasks do users perform?
- What steps are taken to perform tasks?
- What tasks are most critical?
- What information is needed to perform each task?
- What tools are currently used to complete each task?
- What output is generated from user tasks?
- How do users interact with others in the task?
- How do tasks flow in the business process?

A survey by Vredenburg (2002), however, suggests that the methods used in the UCD include anything from informal usability testing, low-fidelity prototyping, heuristic evaluation and navigation design to scenario-based design. It appears that informal and less structured methods tend to be used much more widely than more formal and structured methods in UCD process. Another key finding in this survey is that cost-benefit tradeoff is an important consideration in the adoption of UCD methods. Hence, there is a major discrepancy between the commonly cited measures and the actually applied ones. For example, field studies were generally ranked high on practical importance but relatively infrequently used because they were considered costly, whereas heuristic evaluations were heavily used because they are relatively easy and inexpensive. Interestingly, only 13% of the projects engaged in a full UCD approach in the sense of user involvement at all three stages of the development cycle. Task analysis was common activity, but was usually derived from indirect sources (i.e. not from users directly). They have concluded that UCD methods are generally considered to have improved product usefulness and usability, although the degree of UCD method adoption was quite uneven across different organizations.

## 4.3 Usage-Centered Design

"*To design dramatically more usable tools, it is not only users who must be understood, but more importantly, usage – how and for what ends software tools will be employed*".

This emerging view of software systems as tools is referred to as usage-centered design (Constantine L, 1996). Since good tools support work in order to make someone's job easier, faster simpler, more flexible, or more fun, what is really important is not building software around users (as in the User-Centered Design), but around *uses*. In other words, usage-centered design focuses on the work that users are trying to accomplish and on what software will need to supply via the UI to help them accomplish it. Constantine (1996), the founder of usage-centered design, argues that it is necessary to understand users, but what really matters is to understand what users are doing or trying to do.

The usage-centered design incorporates five key elements (Constantine L *et al.*, 1999a):

- *Pragmatic design guidelines* – The usage-centered design provides a narrower guidance on more specific issues in software usability.

- *Model-driven design process* – The usage-centered design employs a set of simple, interrelated models to model both the nature of uses or usage to which a system will be put through and the organization of user interface that effectively supports those uses.

- *Organized development activities* – The activities of usage-centered design can be incorporated into almost any software development life cycle model,

particularly the Unified Process and other object-oriented software development processes.

- *Iterative improvement* – The usage-centered design incorporates successive refinements based on the findings from usability inspection and tests. Actual implementation is completed in a series of iterations.

- *Measures of quality* – The usage-centered design is supported by an innovative suite of software metrics that allow developers to assess quality of UI designs. These metrics can augment usability inspections, reviews, and testing.

The following questions are addressed carefully in the development of usable software using usage-centered design:

- Who are the users and how will they relate to the system?
- What tasks are users trying to accomplish through the system we are designing?
- What do they need from the system in order to accomplish their tasks and how should it be organized?
- What are the operating conditions under which the system will be used?
- What should the user interface look like (or feel like or sound like) and how should it behave?

The usage-centered design employs abstract models to solve concrete problems. This is not surprising when abstraction is at the very foundation of modern software development practice, from abstract data types to abstract classes. In a usage-centered design, these abstract models are referred to as the **essential models**. They are intended to capture the essence of problems through technology-free, idealized, and abstract descriptions. The outcome is that the models are more flexible, leaving open more options and more readily accommodating changes in technology. The essential models are as follows:

- **Role model** – the relationship between users and the system.
- **Task model** – the structure of tasks that users will need to accomplish
- **Content model** – the tools and materials to be supplied by the user interface, organized into useful collections and the interconnections among these collections.
- **Operational model** – the operational context in which system is deployed and used.
- **Implementation model** – the visual design of the user interface and description of its operation.

The first three models represent the structure of usage and the architecture of the user interface that will support that usage. The role model identifies the roles which users can play. That is, it represents the various forms and patterns of relationships possible between a system and its users. The task model is based on essential use cases that represent specific cases of use in terms of the sundry goals that users can undertake in using a system to accomplish work. The implementation model which is really a final visual design of UI is based on the architectural models and adapted to the actual environment in which the system will be used. The operational model helps adapt the final visual design to the conditions and constraints of the **operational contexts** within which system will be deployed and used. The constituents of operational contexts are:

- *Incumbents* – characteristics of the actual users who will play a given role.
- *Proficiency* – how usage proficiency is distributed over time and among users in a given role.

- *Interaction* – characteristic patterns of usage associated with a given role, use case, or set of use cases.

- *Information* **– the** nature of information manipulated by users or exchanged between users and the system.

- *Functional support* – specific functions, features, or facilities needed to support users in a given role or for a specific use or set of use cases.

- *Usability criteria* – relative importance of specific usability objectives for a given role or for a specific use case or set of use cases.

- *Operational risk* – type and level of risk associated with a given role or for a specific use case or set of use cases.

- *Device constraints* – limitations or constraining characteristics of the physical equipment used.

- *Environment* – relevant factors of the physical environment.

Figure 3 outlines the principal logical relationships among all five models. This diagram should not be taken as a kind of flowchart for usage-centered design. In practice, the usage-centered design models are developed concurrently (as opposed to sequentially as in a waterfall model) with analysts or designers moving back and forth among alternative views presented by the models. It is important to emphasize that the usage-centered design essentially focuses on the user interface and interaction design. Hence, for a complete design of software system, data models such as object-oriented class diagrams are also necessary.

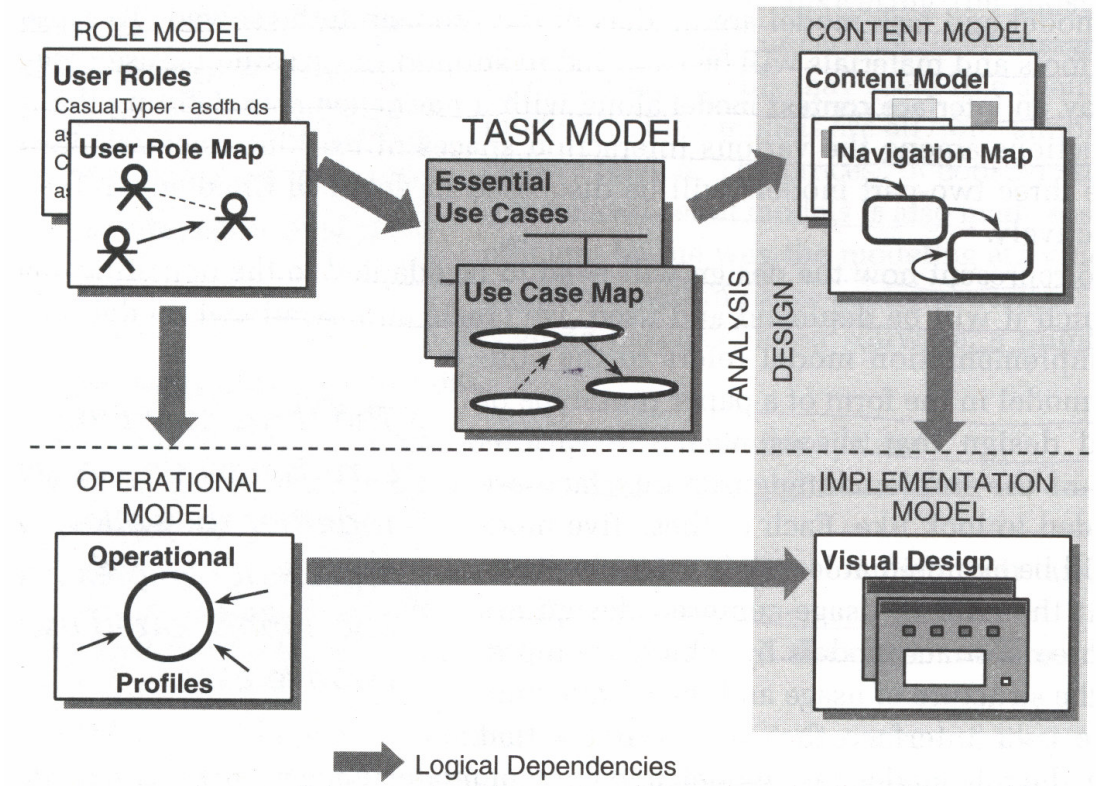



**Figure 3: A Usage-Centered Design Process (Source: Constantine L *et al.*, 1999a)**

## 4.4 Usability Evaluation

Software usability can be evaluated using a number of techniques; *formally* by some analysis technique, *automatically* by computerized procedure, *empirically* by experiments with test users, and *heuristically* by simply looking at the interface and passing judgment according to one's own opinion. The sections below briefly describe the two most widely used usability evaluation methods, namely the heuristic evaluation and empirical usability testing.

### 4.4.1 Heuristic Evaluation

Heuristic evaluation is a usability engineering method for finding usability problems in a user interface design so that they can be attended to as part of an iterative design

process (Nielsen J *et al*., 1990). Heuristic evaluation involves having a small set of evaluators (typically, 3 to 5 people) examine the interface and judge its compliance with recognized usability principles (the “heuristics”). Heuristic evaluation thus falls in the category of usability inspection methods rather than the category of empirical user testing. Furthermore, heuristic evaluation is one of the most informal usability inspection methods and is explicitly designed as a “*discount usability engineering*” method that is easy and cheap to use in practice.

Unfortunately, the number of usability guidelines can be quite large; up to a few hundreds (Smith S *et al*., 1986), and are therefore seen as rather impractical by developers. Instead, most people perform heuristic evaluation on the basis of their own intuition and common sense. A relatively small set of *heuristics* such as the nine basic usability principles (see section 4.1) from Nielsen (1990) seem more suited as the basis for practical heuristic evaluation. Usability researchers have also found that some people are better than others at doing heuristic evaluation (Nielsen J *et al*., 1990). They do not have enough evidence to form a firm conclusion but it seems that it might be the case that there is very little consistency in the ability of evaluators to find usability problems. They also believe that the usability experts will be better at heuristic evaluation than average computer professionals.

Heuristic evaluation is difficult to do (Nielsen J *et al*., 1990). Even in the best case only half of the usability problems are found, and the general case is rather poor. It is necessary to supplement the heuristic method with other evaluation methods to increase the numbers. Individual evaluators are mostly quite bad at doing heuristic evaluation and they usually find between 20 and 51 percent of the usability problems in the interfaces they evaluate (Nielsen J *et al*., 1990). The results are much better if several people conduct the evaluation, and they should do so independently of each other. It is recommended that heuristic evaluation be done with between 3 and 5 evaluators and any additional resources be spent on alternative methods of evaluation (Nielsen J *et al*., 1990).

The heuristic evaluation is particularly valuable in situations where other methods are

difficult to use because of extreme time or resource constraints or because representative users are hard to bring in for user testing. Major advantages of heuristic evaluation are that it is cheap, intuitive and easy to motivate people to do it, does not require advance planning, and it can be used early in the development process. A disadvantage, however, is that it sometimes identifies usability problems without providing direct suggestions for how to solve them.

### 4.4.2 Empirical Usability Testing

Usability testing comes in many variations but the theme is simple: users sit down in front of some version of software and watch them as they try to use it. They are given some tasks or work to complete and they are observed, and their performance is analyzed. The favored scheme is to build a usability testing laboratory (see Figure 4), equipped with an array of computers and audio-video equipment. The usability testing staff consists of psychologists to technicians, and HCI specialists.

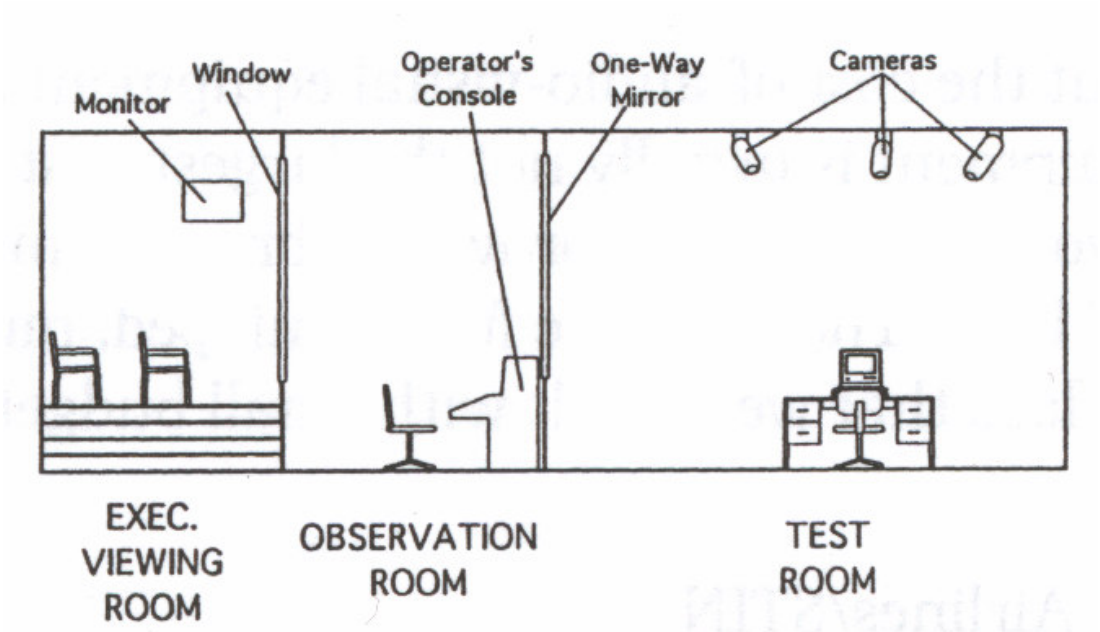



**Figure 4: Usability Test Laboratory (Source: Dumas J *et al.*, 1999c)**

Field testing is the poorer sister in usability. There is no need for need for a usability

lab, and the only absolutely required equipment is a notebook and a pencil. Usability testing in a field setting looks at what people actually do when they are doing real work in an ordinary work setting. This is clearly an advantage considering that the people tend to think and act differently when removed form their work environment and brought into a lab to be videotaped.

Some usability experts believe that only objective testing directly with potential users can ultimately resolve some gray areas where it is unclear just what will work and what will not. This is something that design guidelines or expert inspection could not achieve.

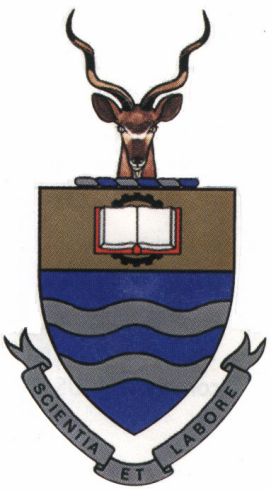
# 5 Conclusion

The literature survey informed the reader about a number of topics pertinent to the subject of this research. They include most recent literature on mobile technology, mobile field applications and usability design process. The section on mobile technology highlighted recent technological development in terms of hardware, wireless networks and mobile applications. But more importantly, it discussed unique characteristics that are intrinsic to mobile computing such as, hardware constraints, operating context and usability requirements. The argument here is that mobile computing is a relatively new paradigm that is fundamentally different from traditional desktop computing and as a result software designers need to take these factors into careful consideration if they are to produce usable mobile applications.

An extensive use of mobile technology for field data collection work is discussed in details. The survey began with discussion around field data collection in public utilities and some case studies in which mobile GIS technology is used to obtain data from the field. But the main focus was given to the way and environment in which fieldwork is carried out. Mobile workers operate in operating context and environment that is inherently heterogeneous where users are faced with different situations that change constantly. In general, mobile work can be described as dynamic, contextual and demands high level of user attention. This demands innovative HCI techniques and usability features from designers. In general, mobile applications require a much higher level of usability than their desktop counterparts.

The survey gave a summary of a number of usability design guidelines, techniques and methodologies currently used by designers. The usability guidelines include definitions of usability rules and design principles widely recognized by the software community. Two model-driven UI design processes called User-Centered Design and Usage-Centered Design are discussed. Both represent a move away from techno-centric way of designing software. The difference however is that the latter focuses on user to obtain software requirements and the latter on the operation

context and usage. The survey ends with a section on various techniques used in usability testing. They are essential for designers if they are to design and effectively evaluate software that meets a high level of usability requirements.




# Software Requirements

Doc. No. 2

# CONTENTS

CONTENTS ..... I

LIST OF FIGURES ..... III

1 SCOPE ..... 1

1.1 Introduction ..... 1

1.2 Purpose ..... 1

1.3 Audience ..... 2

2 REQUIREMENTS ELICITATION ..... 3

2.1 Requirements Information ..... 3

2.2 User Involvement ..... 3

3.3 Requirements Elicitation Technique ..... 4

3 UUC DATA COLLECTION WORK ..... 6

4 USER ROLE MODEL ..... 11

5 TASK MODEL ..... 14

6 OPERATIONAL MODEL ..... 32

6.1 Incumbent Profile ..... 32

6.2 Proficiency Profile ..... 33

6.3 Interaction Profile ..... 33

6.4 Information Profile ..... 35

6.5 Environment Profile ..... 37

6.6 Device Constraints ..... 38

6.7 Operational Risk Profile ..... 39

**7 USABILITY CRITERIA ..... 40**

**8 FUNCTIONAL REQUIREMENTS ..... 43**

**9 HARDWARE REQUIREMENTS ..... 44**

**10 CONCLUSIONS ..... 45**

# LIST OF FIGURES

*Figure 1: Cables, joints and ducts inside the UUC* 6
*Figure 2: Topographic view of UUC showing routefaces* 7
*Figure 3: Data sourcing map* 8
*Figure 4: UUC data collection personnel at work* 9
*Figure 5: Sourcing team, times and site information template* 15
*Figure 6: UUC specification template* 17
*Figure 7: Placing ducts on a topographic view of UUB* 19
*Figure 8: Ducts specification* 20
*Figure 9: Sketch of cable connections inside manhole* 22
*Figure 10: Fibre Optics and Copper cables specification* 23
*Figure 11: Fibre optics and joint specifications* 26
*Figure 12: Use case map for field data collection application* 30

# 1 Scope

## 1.4 Introduction

"*It is often said that the hardest single part of building a system is deciding what to build. This is because the 'problem' of what need to be built is often less well defined, less clear, even fuzzy in many cases.*" (van Vliet H, 2000a)

During the requirements engineering phase, system requirements for the field data collection application are identified and analyzed. In doing so, the usability, software and hardware requirements for the field application are specified. A case study of data sourcing work at a telecommunication utility is used to draw user requirements. Also discussed in this document are various techniques and methodology used to facilitate in gathering, organizing and analysis of system requirements. A usage-centered design approach is used to analyze the usability requirements by taking into consideration user requirements, their tasks and the operational context of the data sourcing work.

## 1.5 Purpose

This document represents the analysis phase of the problem domain which includes intended users, their tasks and operational context in which field data collection application will be used. The purpose of this document is to communicate the results of the analysis phase to the reader. Also, the document serves as an anchor point against which subsequent work can be justified, namely, the system design phase (Doc. No. 3 and 4) and usability evaluation phase (Doc. No. 5).

## 1.6 Audience

The intended readers of this document include researchers involved in the areas of software usability, mobile computing, as well as software designers, developers, the external examiner and other interested parties.

# 2 Requirements Elicitation

## 2.1 Requirements Information

Software requirements information can be obtained from various sources. In commercial software, for example, designers look to other software, previous versions or competing products. They may also talk with people from marketing, technical support or other departments and conduct surveys or analyze customer complaints. But generally speaking, the more accurate and reliable data is obtained directly from the source – users themselves. This was the direction taken in this project whereby the bulk of user requirements for the field data collection application were obtained from actual fieldworkers.

A team of Underground Utility Closure (UUC) data sourcing personnel from Telkom, the country's largest telecommunications utility, assisted in obtaining requirements information. The fieldworkers cooperated in answering questions relating to their work. More importantly, they were accompanied during their field trips to collect network related data and were observed doing their work. Also, a number of UUC paper sourcing templates used in data collection work were made available for analysis.

## 2.2 User Involvement

Although user participation is essential in any software development process, it is not always clear how much they should get involved. One may argue that software development process should be user-driven because it is users who will end up using the software. In a user-centered design (see section 4.2, Doc. No. 1), for example, users are placed at the centre of software development process where they make most design decisions. A Taylorian approach, on the other hand, is strictly functional one. Its underlying assumption is that there is one objective truth,

which merely needs to be discovered during the analysis process (van Vliet H, 2000a).

Unfortunately, each school of thought mentioned above has its shortcomings. When a purely functional view of the world is imposed on developing software that is supposed to support people in doing their job, the outcome may well be ill-conceived systems. Similarly, by relying too much on the users to drive the software development, one is exposed to a number of practical problems. Generally, the users themselves don't always know what they want, or what the future system is capable of doing (van Vliet H, 2000a). As humans, they are inclined to be prejudiced about selecting and using information. Moreover, extensive user involvement may be costly and may lengthen the implementation period. The telecommunication workers who participated in this research did so by their own willingness to help without any remuneration. Hence, cost is not an issue but there is limitation on frequency of field visits to acquire requirements information, or generally, the amount of time that can be spent with them, because they are preoccupied with their jobs and not always able to assist.

Given the limitations mentioned above, neither the Talyorian nor "user-centric" approach would be appropriate for the project at hand. Instead, a more pragmatic approach is adopted whereby potential users assist in acquiring user requirements and evaluation of the prototype. However, they do not drive the software development process by making design decisions. But having said that, users are encouraged to influence the design through their comments and suggestions throughout the development process.

## 3.3 Requirements Elicitation Technique

The techniques used to gather requirement information for the field data collection application include form analysis, user interviews and observations. A multiple requirements elicitation technique was used in an effort to gather as much requirements information as possible. The interviews were carried out with a team of

seven data collection personnel from Telkom during their field trip to collect network related data. During the interviews, users were asked about the data collection process. The interviews were rather informal and open-ended. The intention was to let the users talk freely about their work without imposing the interviewer's views.

The workers use paper forms (see Doc. No. 6) referred to as sourcing templates to record data that they have collected from the field. The forms contain various data fields, specification tables, drawings, data entry rules and general guidelines. Data collection work is largely a form filling activity, and to understand what the work is about, it is appropriate to do a form analysis. Information contained in these forms explained the nature of information obtained from the field (e.g. text/graphic, accuracy, level of details etc.). In addition, they provided information about data objects of the domain, their properties, and interrelations. This type of information is particularly useful for the design of data models later in the Software Design (Doc. No. 4).

Unfortunately, user interviews and form analysis came short of explaining *how* the data sourcing work was actually done – in terms of interactions required, the context and environment. This important information was obtained by observing users at work. A "*motivated looking*" (Millen D *et al*., 2000) approach was adopted due to a limited amount of available field time. This was done by selecting key informants, asking more focused research questions, and using a video recorder to record field visits. From the data sourcing team, a group leader was selected as a "field guide" and he was able to not only answer a broad range of issues relating to their work, but also highlighted the most important aspects of the work as well. Video recordings allowed data from the field to be brought back to the lab for detailed analysis. Repeated viewing of recorded material was done to examine user interactions in a more thorough manner. This was necessary when activities in the field occurred so quickly that the observer was unable to identify all the important details when first encountered at the field site (Brun-Cotton F *et al*., 1995).

# 3 UUC Data Collection Work

This section gives a brief overview of the field data collection work by a data sourcing team from Telkom. This particular utility keeps its vast, complex network of cables underground and many of these cables pass through the Underground Utility Closures (UUC, also called Underground Utility Box or UUB). As part of asset management, Telkom keeps record of the network by doing regular data sourcing activity. The data sourcing team is required to go inside the UUCs and obtain information about the network assets such as cables, ducts and joints etc.

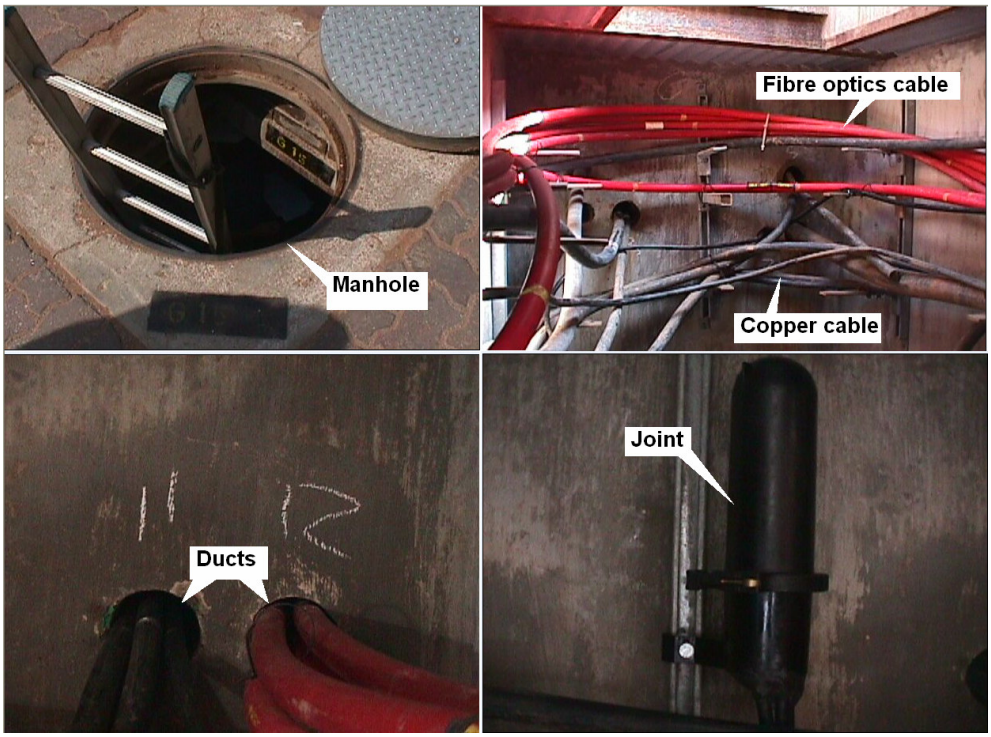



**Figure 5: Cables, joints and ducts inside the UUC**

UUCs are constructed as manholes that one typically finds in the streets, pavements, and road intersections throughout many of our urban areas. Inside the UUC, there are three most important network elements, namely a cable, joint and

duct (see Figure 1). The cables are either a Copper (Cu) or Fiber Optic type, and a joint is created when two or more cables connect each other. Ducts appear as holes in the wall (also called "routeface") from where cables enter (from exchange or other UUCs) and leave the UUC. The direction of cable may be towards or away from the exchange. As a rule of thumb, a wall facing the exchange is labeled routeface A, and naturally the rest of the routefaces (B to H) face away from the exchange (see Figure 2). The number of network elements varies depending on the size, and location of the UUC. In general, UUCs that are close to the exchange contain more network elements than those further away. Some UUCs are rather small, allowing only one person to enter, whereas others may be large enough for four or five people. The shape and dimension of UUCs differ as well. For example, if a UUC has two routefaces that are perpendicular to each other then it is said to be “L-Shaped”.

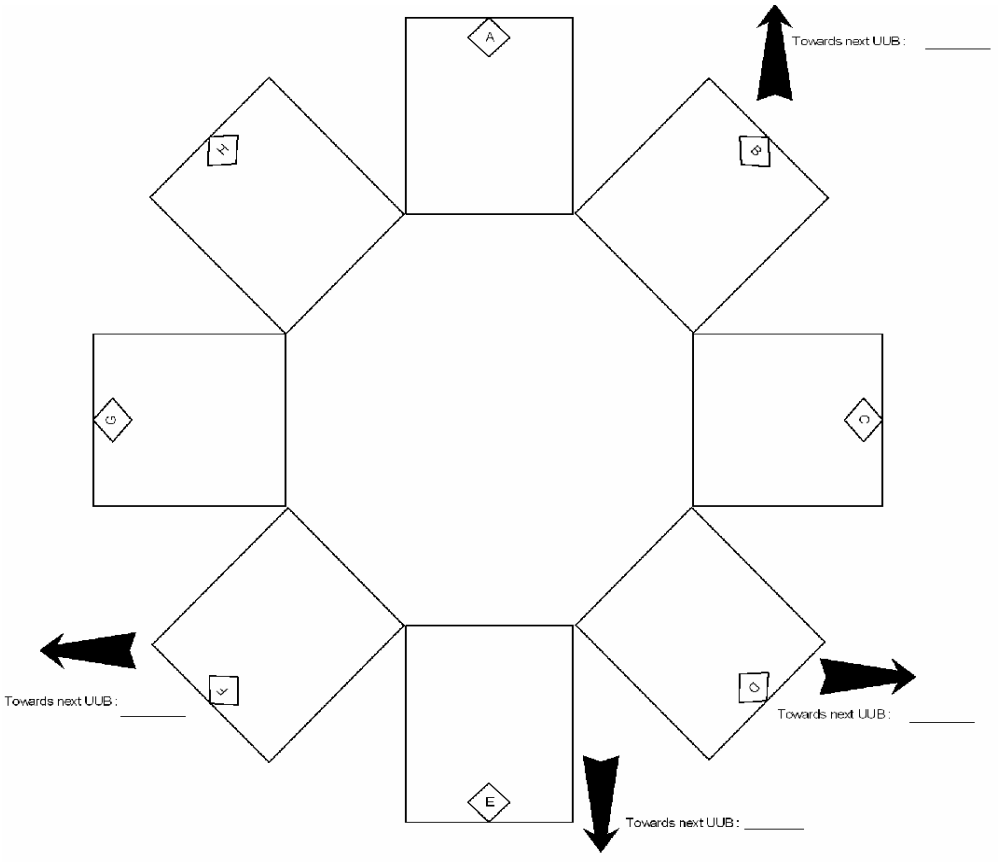



**Figure 6: Topographic view of UUC showing routefaces**

The asset information entered into data sourcing forms includes (also see Doc. No. 6 for more details):

- **Manhole** – ID, location, covers, content (e.g. water, gas etc.), dimensions, shape, construction type and construction status.
- **Duct** – size and construction type.
- **Cable** – size, construction type, path, and specification number.
- **Joint** – type, specification number, cables connected, and construction status.

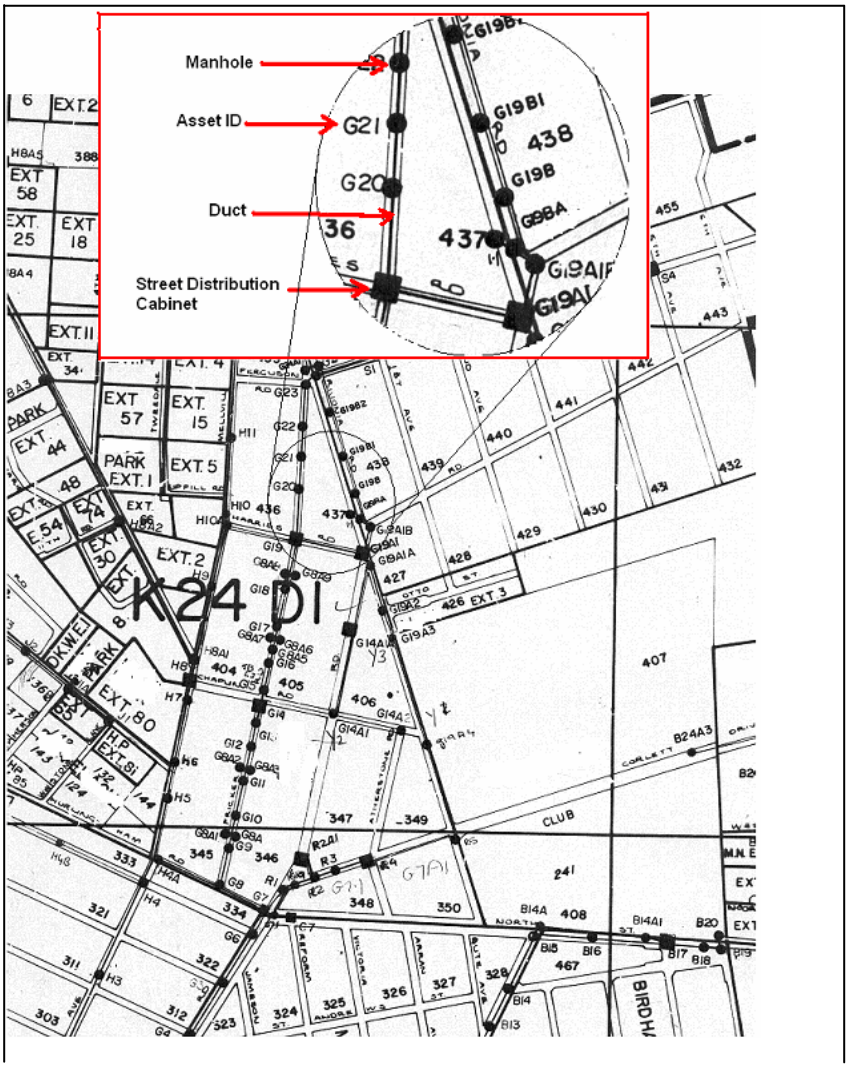



**Figure 7: Data sourcing map**

Other relevant information such as sourcing team details and sourcing times are also recorded in the sourcing templates. Data sourcing takes place both inside and outside the manhole since asset information is visible both inside and outside the manhole. The sourcing personnel are required to do a complete field data audit (i.e. no previous record of information about the assets is provided). The only previous record of the asset provided to the fieldworker is the asset identification number and the location of assets on a paper map (see Figure 3). The map shows the suburb and the streets where the UUCs are located. The asset ID and location of UUC are verified by an office clerk (called data capturer) before the asset information is finally entered into the utility's GIS data repository.



**Figure 8: UUC data collection personnel at work**

It became apparent through careful observation that the UUC data sourcing work is largely a collaborative activity. Each member of the sourcing team is responsible for a particular task. While some workers do the actual sourcing, others assist them by

opening manhole using specialized equipment, pumping water out, standing guard outside the manhole or bringing items requested by workers inside the manhole. The data sourcing work studied was carried out by three workers (see Figure 4); while one worker entered data into the sourcing forms, the other two workers obtained asset information by searching through piles of cables. The workers communicated with each other verbally, and information was read out loud. Working collaboratively was essential in order to do sourcing efficiently and to complete in a reasonable amount of time. A single worker alone cannot search for a cable specification number amongst a few dozen other cables and at the same time record the data into a sourcing template. Both of his hands are already engaged in recording data (one hand for holding a clip-board and the other for writing down the information) and hence preventing him from doing any other task.

# 4 User Role Model

A good design begins with a proper understanding of intended users. User communities may differ in age, gender, training, education, cultural or ethnic background etc. When human diversity is multiplied by a wide range of situations, tasks and frequencies of use, the set of design possibilities become enormous (Shneiderman B, 1998b). Unfortunately, no single design can satisfy all the different variations in users and situations. So, before beginning a design, they need to be characterized as accurately and completely as possible.

A wide variety of methods and techniques can be used to analyze and characterize users. The path taken in this project is to use a usage-centered design models (see section 2.3, Doc. No. 1) to represent users and their relationship with field data collection application. The rationale for this decision is threefold. Firstly, usage-centered design provides a well structured model-driven approach for the software development process. Models have being widely used in many software development processes from flowcharts to use cases and class diagrams, and they serve as unambiguous language for communication among software designers. It is envisaged that models created here will complement the data model later in the design process, and result in an integrated software design. Secondly, a usage-centered design is aimed specifically at developing highly usable software by placing greater emphasis on user tasks, operational context and HCI requirements. And finally, usage-centered design does not place users at the center of the software development process and hence it doesn't require extensive user involvement. This important aspect of usage-centered design is in line with the amount of user involvement envisaged for this project (see section 2.2).

In this section, the roles of users are described in the form of a **user role model** (or simply **role model**). The role model is a list of user roles to be supported by the system. A **user role** is not a title or job description. Instead, it is an abstract collection of needs, interests, expectations, behaviors and responsibilities between

a class or kind of users and a system. The candidate roles identified for field data collection application are as follows:

- **manholeFinder –** finds the location (e.g. street, suburb etc.) of UUC on a data sourcing map.

- **dataSourcer –** collects and records network-related data from the sourcing site (i.e. UUC) into sourcing templates.

- **dataCapturer –** verifies sourced data quality and captures information into the GIS data repository.

- **systemMaintainer –** does a regular housekeeping so that software performs optimally. For example, he ensures there is enough disk space or memory in mobile device to store data collected from the field.

Identifying a user role, however, does not require such role to be fully supported by the software system. A role may be judged to be the most common or may be deemed particularly important from a certain perspective. From a mobile application design standpoint, the **manholeFinder** and **dataSourcer** roles in particular are the most important, because these roles are carried out by fieldworkers in the field. In contrast, **dataCapturer** and **systemMaintainer** roles are usually done in the offices by technically-oriented people whose line of work heavily involves computers. Based on this argument, only those roles that are considered the most relevant to this project (called **focal roles**) are supported by the field application, and they are **manholeFinder** and **dataSourcer** roles.

In the **maholeFinder** role, the user is responsible for locating manhole using a data sourcing map. Specifically, the user needs to know the asset ID, the names of street and suburb in which it is located. The user expects to obtain this information in reasonable time and information be presented clearly. This kind of expectation is understandable when a user is required to locate the right manhole amongst

hundreds of other assets. Selecting the wrong manhole means the data sourcing team traveling over long distance to a wrong location resulting in lost of man-hours and causing disruptions in the work schedule. The user in the **dataSourcer** role, however, is responsible for collecting and entering asset information into sourcing templates. He is assisted by co-workers in obtaining asset information, but he alone is responsible for data-entry. He expects to do this quickly and accurately. In order to do so, he needs to have access to all the sourcing templates including specification lists and data-entry rules.

# 5 Task Model

This section of the document attempts to understand in substantial detail what users will be trying to accomplish in their work using the field data sourcing application, and how they will need to go about it. Establishing what users really need from software to support their work rather than what they want or what they merely think they need requires a dialogue between the designer and users. This dialogue must be in the form of a conversation in which both parties gradually build a joint understanding of the work and how to support it. The **task model** embodies this joint understanding by representing the structure of user needs and tasks.

The task model for the field data collection is implemented using the *essential* use cases taken from a usage-centered design process (see section 4.3, Doc. No. 1). An *essential* use case is defined as (Constantine L *et al*., 1999c):

*"a structured narrative, expressed in the language of the application domain and of users, comprising a simplified, generalized, abstract, technology-free and implementation-independent description of one task or interaction that is complete, meaningful, and well-defined from the point of view of users in some role or roles in relation to a system and that embodies the purpose or intentions underlying the interaction"*

Because the *essential* use case is closer to a purely problem-oriented rather than a solution-oriented view of the task, it leaves open many more possibilities for the design and implementation of the user interface. This important characteristic of essential use case is not found in a *conventional* use case (as employed in object-oriented software design process) where the focus is on a user action, which often results in making assumptions either implicitly or explicitly about the form of interface that is yet to be designed. It consists of a statement explaining overall user purpose or intention, and a two-part narrative comprising the user intention model (left column) and system responsibility model (right column). The former describes

the intent or objective that guides user's action whereas the latter reflects user's expectation regarding the responsibilities of the field application.

Beginning with the **dataSourcer** role, this section of the document entails the *essential* use cases supporting **focal roles** selected earlier in section 4. The very first tasks in UUC sourcing require the user to fill in sourcing personnel details, sourcing times, and site information (see Figure 5). They are fairly simple tasks and the information required to complete these tasks is readily obtained. For example, the sourcing times and team information are already known, whereas the site name and location are read simply from the sourcing map. To complete this task, the user needs to know what information needs to be entered. In addition, where it is to be entered and the type of format it is in. The user intention, therefore, is to obtain all that information and the system is responsible for making it available to the user. Once this information is available, the user intention shifts to entering data. When the data is finally entered, the system needs to validate if the data entered is correct.

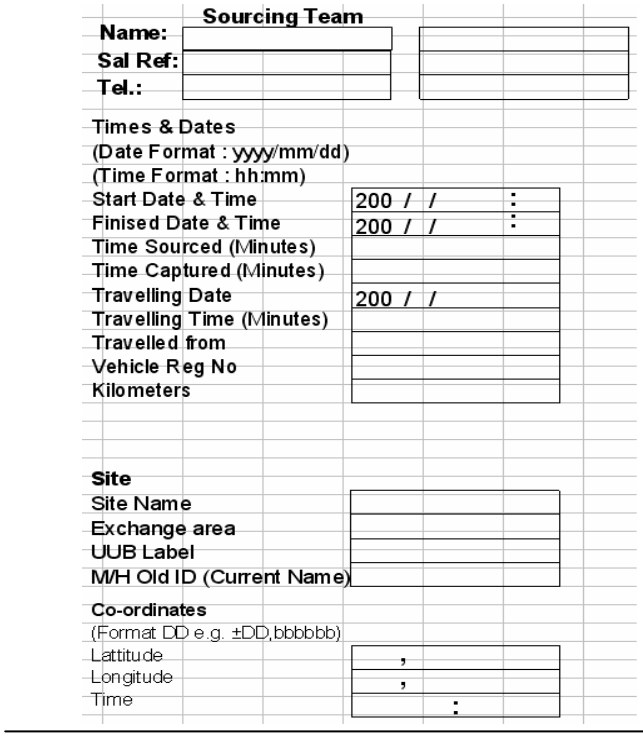



**Figure 9: Sourcing team, times and site information template**

The essential use cases for entering sourcing team, times and site information are as follows:

**enterSourcingTeamDetails**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request member information template | show member information template |
| enter member information | confirm data correctness |

**enterSourcingTimes**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request times and dates template | show sourcing times template |
| enter sourcing times | confirm data correctness |

**enterSiteInformation**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request site information template | show site information template |
| enter site information | confirm data correctness |

The use cases above do not show how sourcing information is presented, or the way user enters the data. A container called “template” is used to generalize the way sourcing information is presented. This is because the objective at this point is to understand what the user and system needs to do rather than how it is done, which is addressed later in the design process. The tasks that follow require the user to enter manhole specifications such as manhole cover type, content, dimensions, and construction status (see Figure 6).

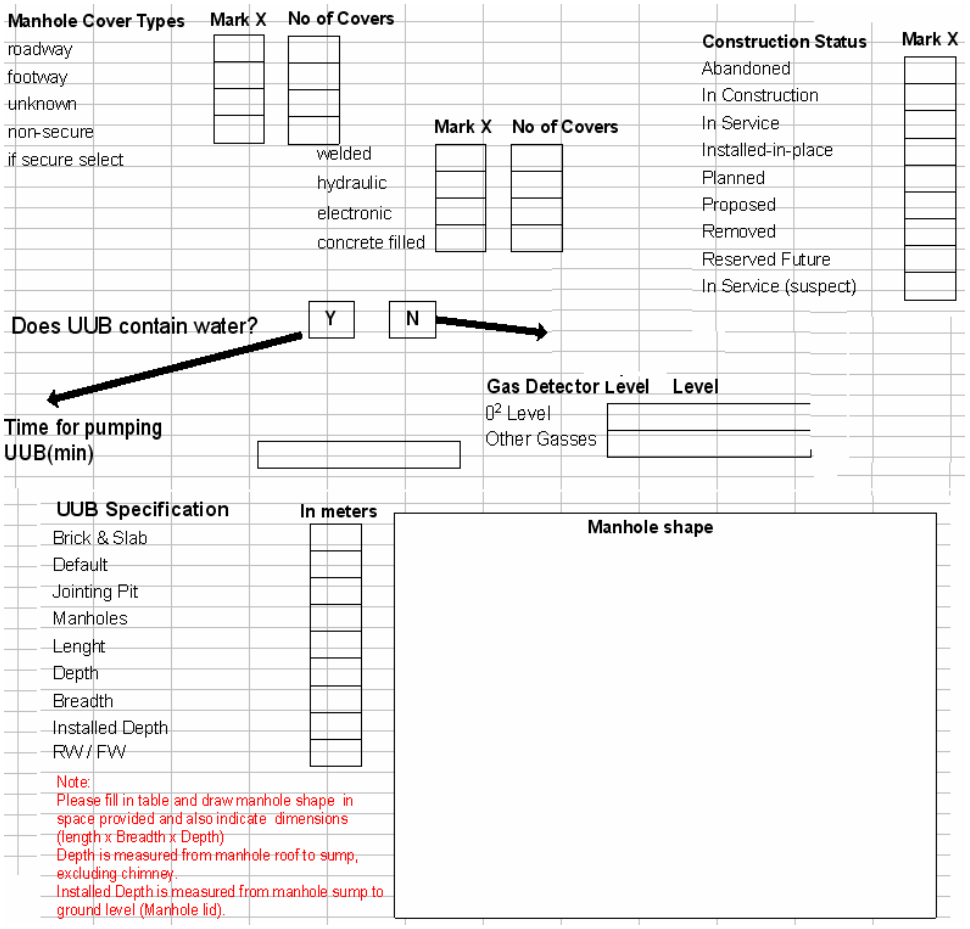



**Figure 10: UUC specification template**

Although the data sourced here is all related to the manhole, the tasks involved are carried out separately in different contexts. For example, the user alone specifies

the manhole cover by mere observation outside the manhole. The latter tasks, however, requires the user to go inside the manhole with the help of co-workers to obtain asset information. Entering the manhole may also be delayed depending on different situations that may arise. It takes time, for example, to secure a street or pavement where the manhole is located. And if the UUC contains water, it needs to be pumped out before the data collector can enter.

The characteristics of use cases supporting manhole information sourcing tasks are comparable to that of sourcing time, and site location. Again, the user needs to know specific details of manhole to be sourced, and some necessary instructions on entering data. For example, a short note explains how to measure the manhole dimensions, and how to indicate them in the sketch. The system, of course, should be assisting the user by presenting necessary information as well as provide a place to enter data. The use cases for sourcing manhole specification are as follows:

**specifyManholeCovers**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request manhole covers information | |
| | show manhole cover types |
| enter manhole cover | |
| | show list of UUC securing devices [only if UUC is secured] |
| enter correct device | |
| | confirm data correctness |

**specifyUUBcontent**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request UUC contents | |
| | show list of UUC content |
| enter what UUC contains | |
| | request time for pumping water out [only if UUC contains water] |
| enter pumping time | |

| | |
|---|---|
| | confirm data correctness |

**specifyUUB**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request UUB specifications template | show UUB specifications template |
| enter UUB specifications | confirm data correctness |

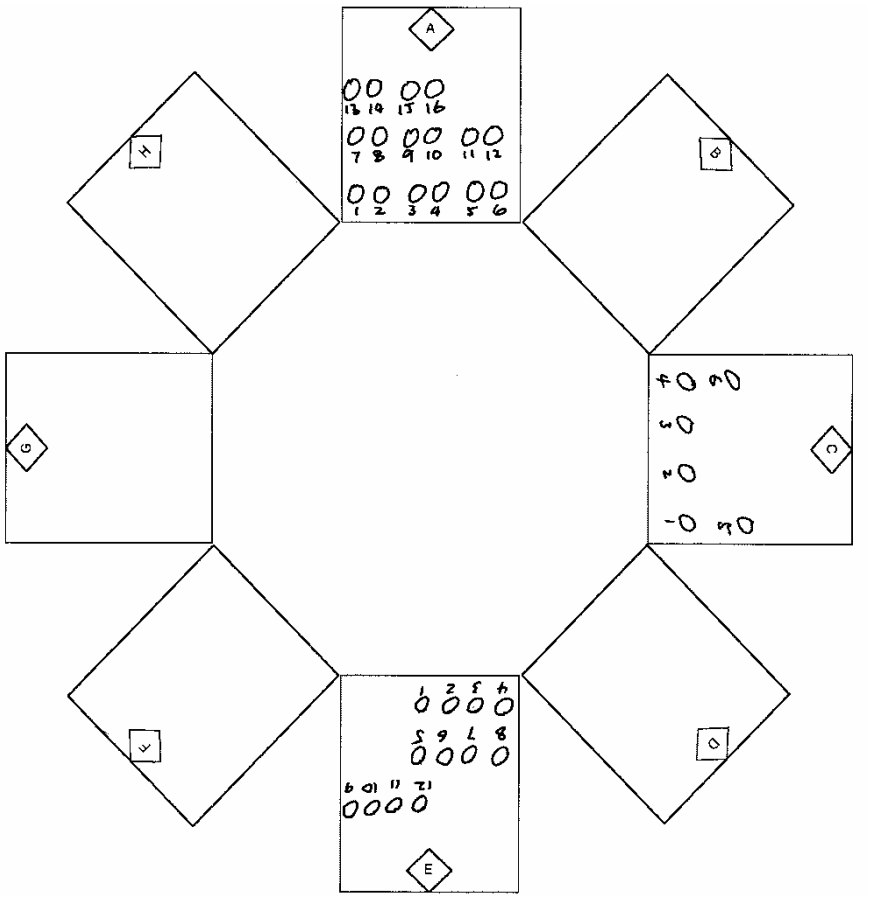



**Figure 11: Placing ducts on a topographic view of UUB**

The word “enter” is used above to generalize how the user inserts data into the system. The user may type in data, select item or draw, to mention a few. Once again, these are design issues and will only be considered later in the design process. Moving on to sourcing ducts information, the user does two tasks. First, he records on the template the location and name of ducts on each routeface for all the routefaces of the UUC (see Figure 7).

The sourcing template has a topographic view of the UUC which the user turns appropriately to face towards the wall that he is inspecting. As mentioned before, each duct is identified by the position on the routeface it occupies. By convention, the wall facing *towards* the exchange is called Routeface A, and walls facing *away* from exchange are given names from Routeface B to Routeface H. The ducts on Routeface A is labeled from bottom left to top right of the wall, but bottom right to top left used in walls facing away from exchange. Secondly, the user specifies the type of duct, and the diameter or size of duct (see Figure 8).

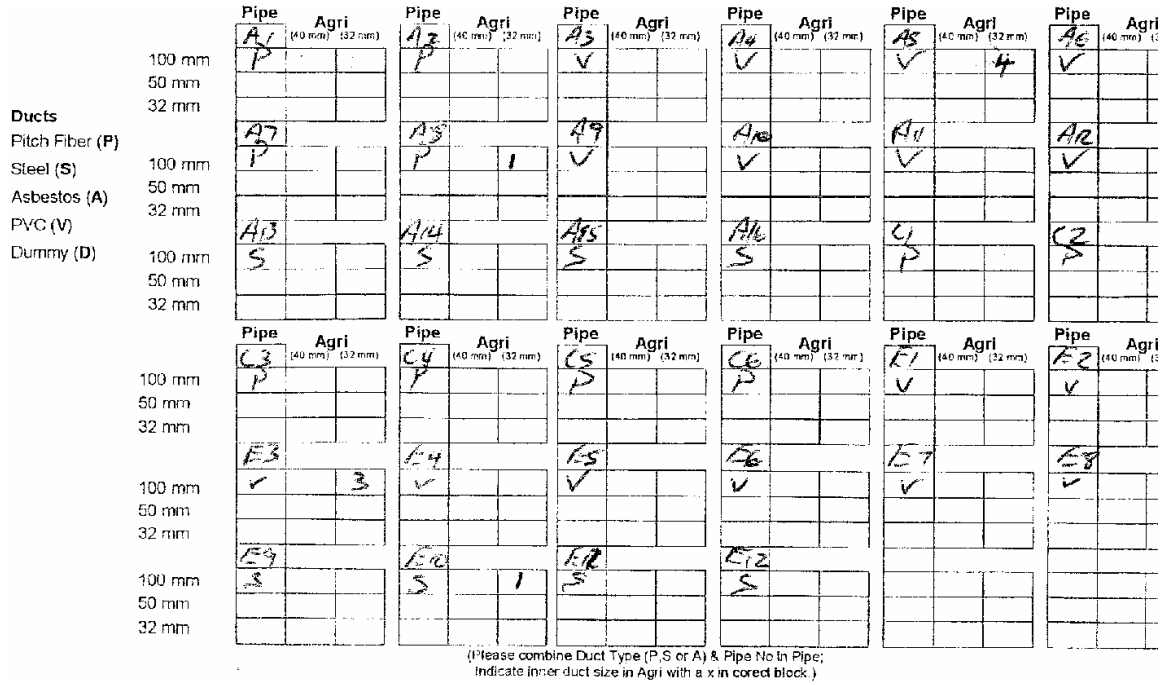



**Figure 12: Ducts specification**

The user's intention in the first task is to find the right place on the template to insert a duct. Although the maximum number of routefaces is eight (from A to H), the UUC typically has three routefaces that has ducts in them. The number of ducts in a UUC may range from a few dozen to hundreds depending on how close the UUC is to the exchange. In the second task, the user intends to specify the ducts. But before doing so, he needs to know their names (or positions) so that specifications correspond to ducts they belong. The use cases for inserting and specifying ducts are as follows:

**placeDuct**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request a place to insert ducts | |
| | provide a place for duct |
| place duct | |
| | confirm placement |
| | [repeat until all ducts are placed] |

**specifyDuct**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to specify ducts | |
| | show ducts |
| select duct | |
| | show duct specification form |
| enter details | |
| | confirm data correctness |
| | [repeat until all ducts are specified] |

Once all the ducts are inserted, the user begins to enter the cables and joints information that are connected to the ducts. Before specifying the cable, the user enters the path of the cable that is to be specified (see Figure 9). The user intention here is to identify the cable from a complex network of other cables. Bear in mind

that the user is working collaboratively with his co-workers. The user himself does not look for a cable path or its specification but rather asked his co-workers for this information. Finding a cable path in a dark UUC is not easy and neither is looking for specification numbers that are small, worn out or covered with dust.

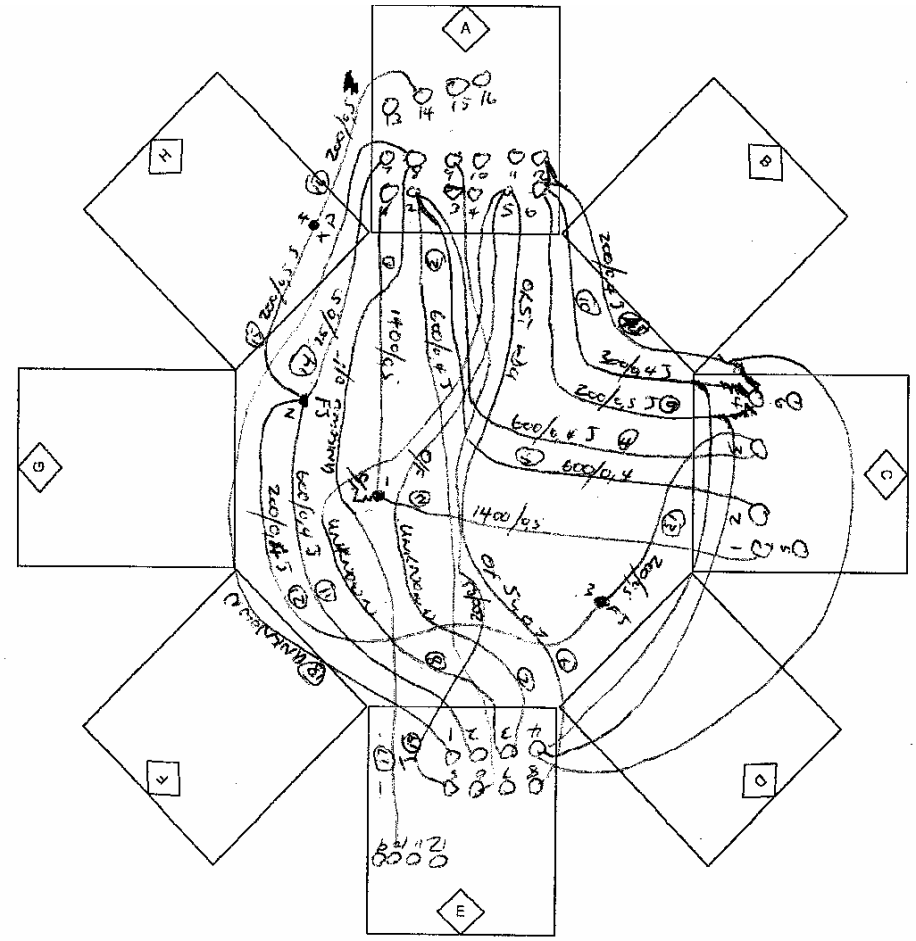



**Figure 13: Sketch of cable connections inside manhole**

Typically, it takes two workers to obtain cable specifications. The user who is entering data into the template must take an opportunity to record data quickly when the information is conveyed by his co-workers. Sketching the cable path gives the user a much clearer picture of cable connections. Also, the drawing can be done

fairly quickly compared to writing out words that describe the cable path. One other strategy employed is that user quickly jots down only the essential features of the cable and joints in the drawing, and leaving details for later. The essential features include the cable ID, cable size and gauge. He usually uses abbreviations to make the jotting down even faster (see Figure 9). For example, “Cu” and ‘P’ indicate Copper cable and pressurize joint respectively. These abbreviations, however, only make sense to the user who wrote them. The abbreviations are merely a means to identify the cable and temporarily record its unique features.

| Cable NO | CU: Pick Cable Size from 15 and write in Size | CU: Gauge : 0,4 0,5 0,63 0,9 1,25 | CU: Type: Poly-APL Cu (U)nit Twin, Poly Cu (A)ir Core, Poly-APL Cu (J)elly, (L)ead Cu Air Core, Poly Cu (S)tar Quad, Un(K)nown, Poly-APL Cu | CU: Usage: (S)ubs, (P)CM, (J)unction | Fiber: Pick Fiber Sheath Spec from 16 and write in Sheath Spec Number | Fiber & CU: Cable Construction Status: (A)bandoned, In (C)onstruction, In (S)ervice, (I)nstalled in place, (P)lanned, P(R)opposed, Re(M)oved, Reserved (F)uture, In Service S(U)spect. | CU: Construction Type: (A)ircore, Figure(8), (J)elly, (O)ther, (S)elf Support, Star (Q)uad, Unit (T)win, UT(P), (Z)-screen, (W)ire | Fiber & CU: DIRECTION From Exchange: Please supply duct number. E.g. A2 | Fiber & CU: Joint: Y/N (If Yes goto Step 18) | Fiber & CU: If No DIRECTION Away From Exchange : Please supply duct number E.g. E2 | Route: Surface Material: (A)sphalt, (C)lay, C(O)ncrete, (D)irt, (G)rass, G(R)avel, (P)aved, (U)nknown | Route: Type: (B)ored, (D)ucted, (L)ashed, (P)loughed, (R)inged, (S)elf Supporting, (T)renched, (U)nknown |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1400 | 0,5 | A | S | | S | A | A1 | Y | | | |
| 2 | 1400 | 0,5 | A | S | | S | A | | Y | C | | |
| 3 | 600 | 0,4 | J | S | | S | J | A2 | N | E6 | | |
| 4 | 600 | 0,4 | J | S | | S | J | A2 | N | C3 | | |
| 5 | 600 | 0,4 | A | S | | S | A | A2 | N | C2 | | |
| 6 | | | | | UNKNOWN | S | | A5 | N | E8 | | |
| 7 | | | | | UNKNOWN | S | | A5 | N | E3 | | |
| 8 | | | | | UNKNOWN | S | | A5 | N | EJ | | |
| 9 | 200 | 0,5 | J | S | | S | J | A6 | N | E4 | | |
| 10 | 300 | 0,4 | J | S | | S | J | A6 | N | EK | | |
| 11 | 600 | 0,4 | J | S | | S | J | E2 | Y | | | |
| 12 | 200 | 0,4 | J | S | | S | J | | Y | | | |

**Figure 14: Fibre Optics and Copper cables specification**

Later on, when the user has an opportunity to ask his co-workers about all the details, he fills in the specifications proper (see Figure 10). Inserting the rest of the specification is usually done only when the full path of the cable is sourced. Hence, this task is delayed especially when the cable path is long, and joined in many places. It is important to emphasize the need for user to practice this kind of work

pattern to adapt to the collaborative work environment in which he finds himself. When the user indicates that he intends to insert a cable or a joint, the system must provide a container for it. The system must then confirm that network element is placed where the user wanted. The use cases for inserting a cable and joint is as follows:

**placeCable**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to place cable | |
| | display cable connections |
| choose cable location | |
| | add cable to network |
| | confirm placement |

**placeJoint**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to place joint | |
| | display cable connections |
| choose joint location | |
| | add joint to network |
| | confirm placement |

As the number of cables increases, the cable network becomes complicated and difficult for the user to find a location for the cable. He may need to browse the cable network before placing a cable or a joint. In this case, the system must show the user all the cables and joints that are connected. A use case for browsing the network is as follows:

**browseNetwork**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request to view network | |
| | display network |
| browse | |
| | display cable connections |

When specifying a cable or a joint, the user is interested in doing two things. First, pick the right cable or joint and then fill in the specifications. The system, on the other hand, is responsible for showing how these elements are connected and information on the specifications to be entered. The use cases for specifying cable and joint are as follows:

**specifyCable**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to specify cable | |
| | show network connections |
| pick a cable | |
| | present specifications required |
| enter cable specifications | |
| | confirm data correctness |

**specifyJoint**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to specify joint | |
| | show network |
| pick a joint | |
| | present specification form |
| enter joint specification | |
| | confirm data correctness |

When entering cable or joint specifications, the user must select a fiber sheath specification, cable size and joint specification numbers for a list of specification (some of the specifications are shown in Figure 11). These tasks require the user to insert the right specification number. The user intention then is to find the correct specification number. The system must show the list and add the specification before exiting.

OPTIC FIBRE SHEATH SPECIFICATION

| Ref No | Specification | Description |
|---|---|---|
| 1 | AVN*L-2/S1 | CABLE OPTC FIBR O/HEAD 2 FIBRE1300NM SINGLE MODE |
| 2 | AVNSL-2/SX | CABLE,FIBRE OPTIC,O/H,2/OFDC,SELF SUPPORT GRADE A |
| 3 | AVNSL-4/S1 | CABLE OPTC FIBR O/HEAD 4 FIBRE |
| 4 | AVNSL-4/SX | CABLE,FIBRE OPTIC,AERIAL SELF SUPPORTING,4,OFDC,401 (FI |
| 5 | AVN*L-4/S2 | CABLE O/F 4/1550,24 DB/KM OVERHEAD |
| 6 | AVN*L-4/S3 | CABLE,FIBRE OPTIC,O/H,4/1310/1550,SINGLEMODE, GRADE |
| 7 | AVN*T-4/G3 | CABLE,FIBRE OPTIC,O/H,4/850/1300,MULTIMODE |
| 8 | AVN*L-6/S1 | CABLE,FIBRE OPTIC,OH,6/1300/SINGLE MODE GRADE A |
| 9 | AVN*T-6/G3 | CABLE,FIBRE OPTIC,6/850/1300,MULTIMODE |
| 10 | AVN*T-6/G3 | CABLE,FIBRE OPTIC,OH,6/850-1300NM MULTIMODE, |
| 11 | AVN*L-8/S3 | CABLE,FIBRE OPTIC,O/H,81310/1550,SINGLEMODE GRADE A |
| 12 | AVN*L-8/S3 | CABLE OPTIC FIBRE 8/1310/1550,24DB O/H S/M |
| 13 | AVN*T-8/G3 | CABLE,FIBRE OPTIC,O/H,8/850/1300,MULTIMODE |
| 14 | APNSL-12/S2 | CABLE,FIBRE OPTIC,OVERHEAD,12,SINGLE,401 (FIBRE GRADE B |
| 15 | AVN*L-12/S1 | CABLE,FIBRE OPTIC,OVERHEAD,12,SINGLE,1300 |
| 16 | AVN*L-12/S3 | CABLE,FIBRE OPTIC,O/H,12/1310/1550,SINGLEMODE, GRADE A |
| 17 | AVN*T-12/G3 | CABLE,FIBRE OPTIC,O/H,12850/1300,MULTIMODE |
| 18 | APNSL-24/S3 | CABLE 24 FIBRE O/H 24/1310/1550NM SINGLE MODE |
| 19 | AVN*L-24/S1 | CABLE OPTICAL FIBRE O/H 24 FIBRE 1300NM SINGLE MODE |
| 20 | APNSL-48/S3 | CABLE 48 FIBRE O/H 1310/1550NM SINGLE MODE |
| 21 | UPNCL-2/S1 | CABLE, FIBRE OPTIC,U/G,,2/1300/SINGLE MODE,GRADE A |
| 22 | UPNCL-4/S1 | CABLE,FIBRE OPTIC,U/G,4/1300,SINGLEMODE |
| 23 | UPNCL-4/S1 | CABLE OPTC FIBR U/GRND 4 FIBRE |

CABLE JOINT SPECIFICATIONS

| Ref No | Specification | Description |
|---|---|---|
| 1 | DMNTK/10-25 | Tank Dome Joint 10-25 pairs (TUG) |
| 2 | DMNTK/50-75 | Tank Dome Joint 50-75 pairs (TUG) |
| 3 | DMNTK/100-150 | Tank Dome Joint 100-150 pairs (TUG) |
| 4 | DMNTK/200-300 | Tank Dome Joint 200-300 pairs (TUG) |
| 5 | DMNET/100 | Egerton Dome Joint 100 pairs |
| 6 | DMNET/200 | Egerton Dome Joint 200 pairs |
| 7 | DMN*/100 | Unknown Dome Joint 100 pairs |
| 8 | DMN*/200 | Unknown Dome Joint 200 pairs |
| 9 | INLTK/100-600 | Tank Re-enterable In Line Joint 100-600 pairs |
| 10 | INLTK/400-1000 | Tank Re-enterable In Line Joint 400-1000 pairs |
| 11 | INLTK/1000-1400 | Tank Re-enterable In Line Joint 1000-1400 pairs |
| 12 | INLTK/1600-2400 | Tank Re-enterable In Line Joint 1600-2400 pairs |
| 13 | HSUFS/10-25 | 3M Foam Sealed Heat Shrink Joint 10-25 pairs |
| 14 | HSUFS/30-100 | 3M Foam Sealed Heat Shrink Joint 30-100 pairs |
| 15 | HSUFS/100-200 | 3M Foam Sealed Heat Shrink Joint 100-200 pairs |
| 16 | HSUFS/200-400 | 3M Foam Sealed Heat Shrink Joint 200-400 pairs |
| 17 | HSUFS/400-600 | 3M Foam Sealed Heat Shrink Joint 400-600 pairs |
| 18 | HSUFS/600-1200 | 3M Foam Sealed Heat Shrink Joint 600-1200 pairs |
| 19 | HSUFS/1200-2400 | 3M Foam Sealed Heat Shrink Joint 1200-2400 pairs |
| 20 | HSU*/10-50 | Unknown Heat Shrink Joint 10-50 pairs |
| 21 | HSU*/50-100 | Unknown Heat Shrink Joint 50-100 pairs |
| 22 | HSU*/100-400 | Unknown Heat Shrink Joint 100-400 pairs |

**Figure 15: Fibre optics and joint specifications**

The use case for inserting specification numbers for cable and joints are as follows:

**`insertCableSize`**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request cable size list | present cable size list |
| pick the correct size | confirm selection<br>close |

**insertFibreSheathNo**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request fiber sheath numbers | |
| | present fiber sheath list |
| pick sheath number | |
| | confirm selection |
| | close |

**insertJointSpecNo**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| request joint specification numbers | |
| | present joint specification list |
| pick joint specification number | |
| | confirm selection |
| | close |

Once the data is inserted, the user may want to go back and correct errors that may have been entered. The error may include the way the cables and joints are connected or their specification. In this case, the user will want to see previous records of data before entering a new set of data, and the system must make this information available. After the user makes the changes, the system then overwrites the previous record. The changes can be made to the duct, cable and joint specifications and their positions and connections. The use cases for these tasks are as follows:

**changeNetworkConnection**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| indicate to change network connection | |
| | present network connections |

| | |
|---|---|
| make changes | |
| | confirm changes<br>[continue until user is satisfied] |

**changeDuctSpecifications**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| locate a duct<br>request duct specifications | |
| | present duct specifications |
| make changes | |
| | confirm changes<br>close |

**changeJointSpecifications**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| locate a joint<br>request joint specifications | |
| | present joint specification |
| make changes | |
| | confirm changes<br>close |

**changeCableSpecifications**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| locate a cable<br>request cable specifications | present cable specifications |
| make changes | confirm changes<br>close |

Switching now to the **manholeFinder** role, the user intention here is to find UUC and the way to get there. In locating the UUC, the user scans through the map (see Figure 3) and looks for the names of UUC. Typically, the sourcing team chooses a particular street in a suburb and all the UUCs on that street are sourced in a single field trip. Hence, the user needs to find the names of streets and suburbs. In the use case to find UUC, the system must be able to present the user with necessary details so that he can navigate through the map and locate accurately where the next UUC is and also how to get there. The use cases for searching UUC, street and area are as follows:

**`findUUC`**

| USER INTENTION | SYSTEM RESPONSIBILITY |
|---|---|
| find suburb<br>[if necessary] | |
| | show suburb |
| find street<br>[if necessary] | |
| | show street |
| find UUC | |
| | show UUC |

Sometimes, the user is familiar with the area, especially when he is working at a particular site for days or weeks, whereby he knows the suburb or street well. He can then directly find the UUC and quickly get there. In such case, the first two steps of the finding UUC task can be skipped as shown in **findUUC** use case above.

Now that all the use cases are individually discussed, it is important to know how they are interrelated. For example, some tasks are related to each other more than others, and in many cases belong to one main task. Hence, the **use case map** (Constantine L *et al*., 1999d) is used to represent the interrelationships between

essential use cases supporting the **dataSourcer** and **manholeFinder** roles (see Figure 12).

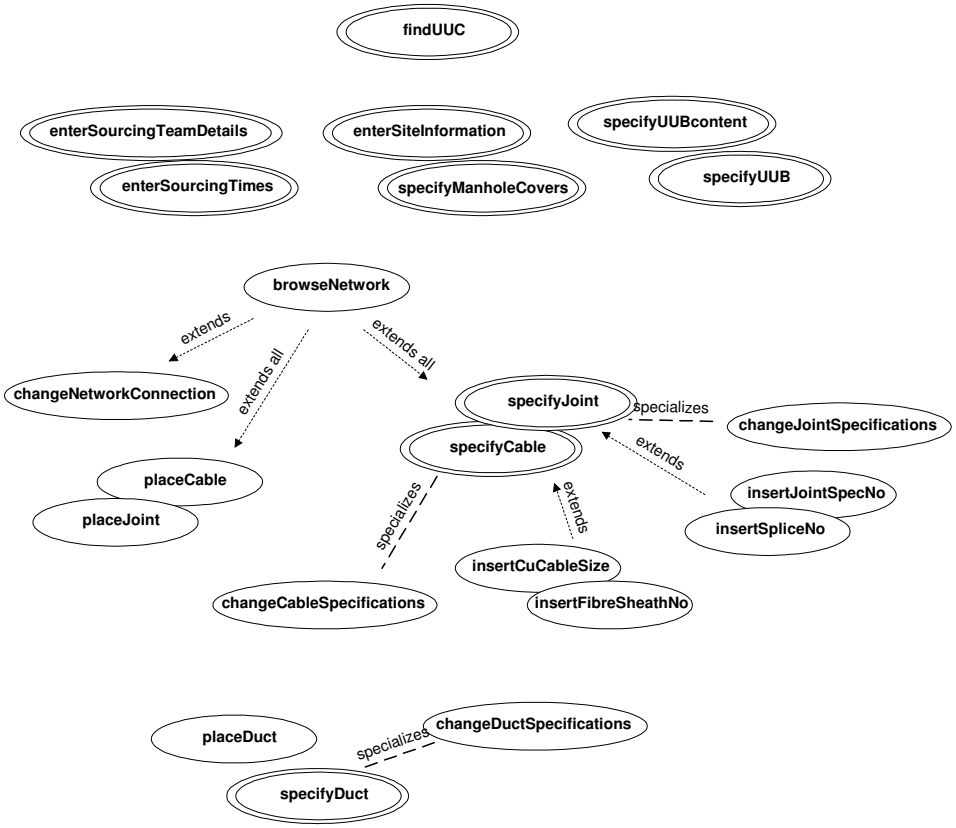



**Figure 16: Use case map for field data collection application**

The use cases above are interrelated by **specialization** or **extension**. They are defined (Constantine L *et al*., 1999d) as follows:

**Specialization:** When one use case “is-a-kind-of” another use case, then they are said to have a specialization relationship. This relationship is comparable to the class-subclass relationship is used in an object-oriented analysis and design. A double-lined arrow is used to indicate specialization in a use case map. The line is labeled “is-a” or “specializes”.

**Extension:** One use case is said to “extends” another use case if it represents inserted or alternative patterns of interaction within the course of the use case being extended. It is represented by a dotted line and arrow labeled “extends”.

# 6 Operational Model

Truly usable software is highly attuned to its environment. This view emphasizes that usable systems need to be fitted not only to the work that they support but also the *context* in which that work takes place. The **operational model** (Constantine L *et al*., 1999g) captures the operational context of UUC sourcing work. Importantly, the operational context affects various design objectives, such as the speed of operation, accuracy, ease of learning, readability and the like. It also has direct impact on highly specific design decisions and details, such as appropriate use of sound, colour, arrange of Graphical User Interface (GUI) components. The constituents (see section 4.3, Doc. No. 1) of operational model for the field data collection application are discussed in the sections below.

## 6.1 Incumbent Profile

The **incumbent profile** represents the various bits of information about the intended users who will play a particular role in relation to the system. The two categories of user information that are considered to be the most pertinent to the usability design are the **domain knowledge** and **system knowledge**. The former refers to how much users in a particular role are likely to know about the application domain that the system supports. More specifically, how much users in the **manholeFinder** and **dataSourcer** roles know about finding manholes and data sourcing respectively. And the latter refers to how much users in the two roles can be expected to know about the system, how it operates, and how to use it. But it is not possible to know exactly what the limitations are of each and every user. The incumbent *profile*, therefore, makes reasonable assumptions about users based on the interviews conducted on the UUC fieldworkers and a general impression of what a typical fieldwork might be.

The users in **manholeFinder** and **dataSourcer** roles are assumed to have the expertise and knowledge about data sourcing work. They may have acquired this

knowledge from training or experience over time. Based on this assumption, the user in the dataSourcer role to a large extent knows what to do with the sourcing template. For example, they are familiar with technical jargons, terms and vocabulary used such as specifications names, data entry labels etc. Similarly, the **manholeFinder** knows how to use a sourcing map and able to read specific information and symbols used on the map such as the UUC, areas, suburb and streets.

However, the users are assumed to be have limited skill and experience with the use of a mobile computer, mobile operating system and mobile technology in general. But having said that, users may already have some basic knowledge in use of desktop computers, operating system (e.g. Microsoft Windows™), and some mobile devices such as cellular phones and GPS equipments.

## 6.2 Proficiency Profile

The users in **dataSourcer** and **manholeFinder** roles are expected to use field data sourcing application as well-informed users but not necessarily as experts. It is expected that users will be given some basic training before they use the application. Again, the viewpoint here is that users are experts at their work but they lack the technical know-how and knowledge about the technology used to support it. However, users are expected to use the application frequently, because several UUCs are sourced in each day, for every working day. So, as frequent users (as opposed to first-time or occasional users), they may well attain expert usage level later on.

## 6.3 Interaction Profile

This section looks at how users in particular roles can be expected to interact with the system being designed. The aspects of interaction that are considered to have

significant influence on the user interface design of the UUC sourcing application include:

- *concentration* – Is usage concentrated into burst or batches or is it more distributed?
- *intensity* – What is the rate of interaction?
- *predictability* – Does the interactions happen in a particular pattern?

The **task model** in the previous section has shown that the data sourcing work consists of a number of smaller tasks. Some of these tasks are fairly simple to do while others require a greater effort from the data sourcing personnel. Looking at the **dataSourcer** role, the number of interactions is concentrated around tasks that are complicated. Specifically, tasks that collect information relating to the cable network (i.e. cables, joints and ducts) require more user interaction than those that collect sourcing team, times, site or UUC specifications. This is because the bulk of data sourced is related to network elements, which is not surprising because the main objective of UUC sourcing is to collect underground cable network information. In addition, the network information is rather comprehensive and hence contains a high level of details (see Figure 10). For example, a sketch representing the graphical view of cable network contains a great deal of details about specifications and interconnection between cables, joints and ducts.

When the user in the **dataSourcer** role is assisted by other workers, he is under constant pressure to keep up with the work pace without slowing down the rest of the team. In such a collaborative work environment, the user learns to do things simultaneously, quickly and opportunistically. For example, the user is constantly listening to co-workers while jotting down the specifications. When he hears the information, there is a sudden rush to record it. The result is a high rate of user interaction amplified by repetitive nature of the tasks. Although the steps required in each task and interactions are predefined, the order in which they are executed is difficult to predict. To illustrate this problem, consider a situation where the user sketches the cable but the co-worker is still busy finding the specification number for

that cable. Then another co-worker shouts out specification number of another cable, hence the user is forced to abandon the former, and search for a new cable in the sketch. Similarly, the user may choose to fill in the text fields in an ad-hoc manner when information is not available at that time. Generally speaking, fieldworkers' actions are fairly unpredictable because they tend to adapt to circumstances and work accordingly given many uncertainties one finds in the field.

Turning to the **manholeFinder** role, locating a UUC can be difficult on a map that covers a large area. But if the system does the searching, the user is only left with asking the system what to search for and how to present it. In that case, both the concentration and intensity of user interaction is fairly low. But at other times, the user may just want to browse through the map but not necessarily search for a particular UUC. In this case, the number of user interaction increases but the rate of interaction remains low. Unlike the **dataSourcer** role, the user does not work collaboratively and as a result he doesn't have the pressure to keep up with others. Also, the amount of tasks and their complexity are also relatively low. The tasks here are limited to finding a UUC, street, or a suburb, and occasional browsing of the map. This makes the interactions within the **manholeFinder** role fairly predictable, if not predefined.

## 6.4 Information Profile

Where information originates and how it flows between the user and system has important implications for the user interface design. For example, if the user has to listen to obtain information, the system should not use audio feedback alone, or high information complexity calls for a user interface that has comprehensibility, clarity in presentation and layout. The **information profile** compiles with what is known about the nature of the information being exchanged between the system and users in a particular role. Included in this *profile* are four aspects of information:

- Input origins – Where does the input from the user in this role originate? What is its ultimate or actual source?

- Flow direction – Does information flow predominantly from or to the user?
- Information volume – How much information is available and of interest to the user?
- Information complexity – How complex is the information available and of interest to the user?

Beginning with the **dataSourcer** role, the user obtains information indirectly from co-workers who are responsible for looking up the specifications. They communicate using their voice by speaking aloud or shouting out the information to the user who then enters them into the sourcing templates. Having said that, some of the information entered is obtained by the user himself. In particular, information about the team members, sourcing times and data collection site are obtained by the data sourcing personnel observing what is around him or information that is readily available in his mind. Overall, it is fair to say that the majority of sourcing information originates with visual and audio channels rather than mental means.

The primary purpose of data sourcing is to obtain asset information. It is not surprising then that in the data sourcing work, information flows from the user into the system. Generally speaking, this is the case for form filling or any work that involves data-entry. However, there are exceptions, for example, when the user uses templates to retrieve specification numbers from a list. But it can be argued that eventually this information goes back into the template. The amount of information obtained in data sourcing depends on the number of network nodes (i.e. duct, cable and joint). A typical UUC has around fifty cables, a few dozens ducts and joints. But if one considers approximately five specification attributes (e.g. size, type, construction status etc.) for each node, it adds up to hundreds of separate data entries. This is fairly large amount of information for a single UUC.

To a large extent, the information obtained in the **dataSourcer** role is rather straightforward. The data sourcing is not a problem-solving activity, and as a result it does not demand the user to solve complicated problems. Often, a simple selection from a given lists of items is all it's required. Having said that, information may

appear to be somewhat complex to the untrained eyes. In particular, the topological view of UUC shows a network of cables, joints and ducts. But on closer examination, it is essentially a tool to help the user build up a mental picture of the cables connection and makes very much easier for user to enter information correctly and efficiently.

Coming to the **manholeFinder** role, however, the information flows from both directions. The user asks the system where a particular asset is located and it subsequently provides the information requested. The information presented may be a UUC, street or area of a suburb, or a combination of all these elements. When the user is searching for a UUC, he is thinking about its location, and how to get there. As a result, the information that user seeks from the system originates to a large extent from the user's mind. The rest of the information can be obtained aurally and visually from co-workers and environment respectively. Volume of information is moderate if the information is presented graphically rather than using words, and so too is the complexity of the information. Under these circumstances, the graphical presentation of information will be most appropriate.

## 6.5 Environment Profile

The location and type of environment in which UUC data collection application would be used vary considerably between the two roles and even within themselves. In the **dataSourcer** role, the user starts data sourcing outside the UUC and finishes off with cables inside UUC. It takes place in two very different environments with each having various implications on the user interface design. Outside the manhole, the user is in an open-air environment either on a public street or pavement. The user is exposed to environmental elements whether it is the sunlight, rain, or wind. In addition, streets and pavements are usually noisy with motor vehicles and other road users. From the exposure to elements perspective, the user inside the manhole is in an environment that is comparable to indoors. The noise level is moderate due to verbal communication between the workers, beeps and alarms generated by equipments such as gas detectors. Because the manhole

is an enclosed area, these sounds generate echoes which further amplify the noise. Certainly, the most obvious problem in the manhole is poor visibility due to lack of light (see Figure 4). The workers generally use torches to overcome this problem although there may be some amount of sunlight that come through the manhole openings.

For the **manholeFinder** role, however, it is difficult to pinpoint exactly where it will take place. The user chooses to find the manhole when and where it is most appropriate for him. For example, he may be sourcing all the manholes in the street in one site visit, so he doesn't need to look up every time where each manhole is. Or, he may use the system to locate the UUC while he is traveling in the car to get there. This is typical of a mobile work that takes place in different environments and in many different contexts. Since the data sourcing work involves constant traveling in the car, also equally on the street at field site, it is expected that the user in the **manholeFinder** role to carry out his tasks in both of these environments. Having said that, the user is not expected to use the system while he is driving but rather as a passenger. Inside the car, the user would be protected from the environmental elements and other disturbances except from a low level of engine noise and vibrations.

## 6.6 Device Constraints

The hardware constraints and limitations of the mobile device for the field application are as discussed in section 2.3.1 of Doc. No. 1. Obviously, some of the constraints are more prevalent than others for a particular mobile device. For example, a PDA has a greater limitation on screen real estate than say a tablet computer. In order to minimize the impact of such limitations on the mobile application, the hardware should be carefully selected so that it meets not only the functionality requirements but more importantly usability requirements as well. The hardware requirements for the field data collection application are specified in section 9.

## 6.7 Operational Risk Profile

Since UUC sourcing work is essentially about acquiring asset information from the field, it is of paramount importance that the information obtained be saved and safely transported to the data repository. The user may lose important data if the computer crashes or becomes stalled which is inevitable in many operating systems. Given a hazardous nature of fieldwork, a user may drop the mobile device, causing a serious damage and result in a permanent loss of data. Failure to follow proper procedures to operate a mobile computer can also result in system errors or even cause the device to malfunction. For example, ignoring a low battery warning will cause the device to shut down and results in loss of data as well.

# 7 Usability Criteria

In addition to the universal usability goals (i.e. time to learn, speed of performance, rate of errors, retention over time, subjective satisfaction etc), other criteria that are specific to a particular user role also come into play. That is, each user role has specific usability requirements that are different from other roles. In addition, some of these requirements may be more important than others. This section highlights those aspects of usability that are most important to a particular role and hence likely to have the greatest impact on usable software design objectives.

The usability criteria for the user in the **dataSourcer** role are as follows:

1. Since users are not particularly skilled at computer usage, they cannot be expected to recognize and understand complex GUI widgets and common Windows metaphors. Hence, the interface should be readily apparent and reflect the structure of the work being supported so that the users relate it to the work they are doing.

2. Fairly predictive and repetitive nature of tasks call for user interface that is optimized for efficiency. This means that the designer must recognize and understand certain workflows and patterns, and tailor the structure and layout of user interface accordingly. In addition, as frequent users they expect to complete the data sourcing tasks fairly quickly otherwise they may become frustrated.

3. A high rate and concentration of user interaction often results in the user making faults, hence user interface should be fault tolerant, or better still, minimize and prevent the user from making errors.

4. The fieldworker often works collaboratively, drawing his attention away from the system. The user interface should not demand too much user attention.

Some form of feedback is appropriate to enable the user know what is going on with the system. Bear in mind that the work environment can be noisy, so both a visual and audio feedback is necessary.

5. Because the data sourcing work in primarily about collecting data, the system should be oriented towards data entry application. This mean the user interface must verify and validate the data entered. It must also show compatibility of data entry and allow flexibility of user control over data entry. And finally strive towards a minimal input action by the user.

6. A fairly high complexity of interaction, volume and complexity of information collected calls for clear, concise and comprehensive representation of the data collected.

7. The application must take into consideration changes in environment and limitations of the mobile device that it is run on. For example, the system must be able to work in a dark manhole as well as in direct sunlight.

8. Due to a fairly unpredictable nature of the fieldwork, the user should be given control of the system. Although there may be automation to reduce workload but the user should be able to override it when necessary. This flexibility applies also to the manner and the order in which certain tasks are completed.

The usability criteria for the user in the **manholeFinder** role are as follows:

1. The application must present data sourcing map comprehensively, and with clarity.

2. The application must be flexible by allowing the user to be in control of the task that he is doing.

3. The data sourcing map should be consistent in appearance by not changing attributes, symbols or other conventions thereof.

# 8 Functional Requirements

This section of the document gives a high level overview of the functional requirements for the UUC data collection application (for detailed software functionality, see Doc. No. 4). The objective of this project focuses primarily on the usability aspect of the software. However, no matter how many usability goals that may be envisaged, until the required functions are implemented, the system is simply unusable. In addition, usability attributes such as system response time can be measured effectively only after the functionality is fully implemented.

The mobile application for field data collection must:

1. Provide the user with all the UUC sourcing templates (see Doc. No. 6) on a mobile device for field data collection.

2. Save the asset information in a format that is compatible or can be read from corporate GIS. For example, most commonly used files are XML and .txt files. And temporarily store the asset information on the mobile device until the data can be synchronized with a database on a desktop computer.

3. Present the user with a data sourcing map that contains details such as the asset identification, location, names of streets and suburb etc. The user must be able to navigate and search for assets, streets and suburb etc.

# 9 Hardware Requirements

The field data collection application requires a hardware platform or a mobile device on which to be implemented. More importantly, the target platform must be able to support both usability requirements and functional requirements specified in section 8 and 9. In addition, the target device must also be able to operate in the operational context of fieldwork (described in section 6). From a software development viewpoint, the hardware platform will also determine the choice of programming tools and off-the-shelf products available for the design and implementation of field application. Having considered all these arguments, the criteria for the target device is that it must:

1. Be lightweight and portable (preferable to be able to carry in one hand).

2. Be water-proof, dust-proof and shock-proof (or drop-proof).

3. Operate equally well in all light conditions from darkness to direct sunlight.

4. Have a display with high resolution to view data sourcing map with a high level of detail.

5. Have a wireless network capability as well as a serial connection to a desktop computer for data synchronization.

6. Have enough disk space to store the asset information collected over a period of a few weeks or months.

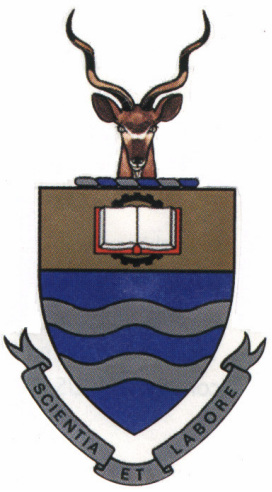
7. Support a rich set of text-entry tools or input devices for data entry.

8. Have a battery life that lasts for at least one working day (approximately 8 hrs).

# 10 Conclusions

A proper understanding about the field data collection work is obtained from the requirements elicitation and analysis process. A direct interaction with actual fieldworkers at the data sourcing site was productive and it produced relatively accurate and reliable requirements information. These requirements, however, are not necessarily applicable to all types of data collection work. This is because the case study of field data collection work is based on a UUC data sourcing work which has characteristics that are unique to this particular fieldwork.

The use of usage-centered design process in requirements analysis proved to be effective in obtaining an in-dept understanding of intended users, their background and more importantly, their tasks and operational context of their work. It is felt that the software requirements that were obtained address the usability and functional aspects of the field application in a practical manner. Furthermore, the hardware requirements for the field application reflect realistic expectations in terms of functional capabilities and sophistication from a modern mobile device.




# User Interface Design

Doc. No. 3

# CONTENTS

**CONTENTS** ..... I

**LIST OF FIGURES** ..... II

**LIST OF TABLES** ..... III

**1 SCOPE** ..... 1

**1.1 Introduction** ..... 1

**1.2 Purpose** ..... 1

**1.3 Audience** ..... 2

**2 USER INTERFACE DESIGN** ..... 3

**2.1 Overview** ..... 3

**2.2 Content Model** ..... 4

**2.3 Implementation Model** ..... 14

2.3.1 Target Platform ..... 15

2.3.2 Implementation Model illustrated ..... 19

**3 CONCLUSION** ..... 45

# LIST OF FIGURES

*Figure 1: Interaction contexts for entering sourcing team, sourcing time and site information. _____5*
*Figure 2: Interaction context for entering manhole specification_____6*
*Figure 3: Interaction contexts for adding and changing network node specifications _____8*
*Figure 4: Interaction context for placing ducts_____9*
*Figure 5: Interaction context for placing joints and cables _____10*
*Figure 6: Interaction context for finding manhole _____11*
*Figure 7: Navigation map for interaction contexts _____12*
*Figure 8: iPaq H3900 PDA_____17*
*Figure 9: Select sourcing team members _____20*
*Figure 10: View sourcing member information _____21*
*Figure 11: Update sourcing member information_____22*
*Figure 12: Data entry for sourcing times and date_____23*
*Figure 13: Form and scrollbar resized when SIP is opened_____24*
*Figure 14: Enter site information _____25*
*Figure 15: Specify manhole covers _____26*
*Figure 16: Specify manhole content _____27*
*Figure 17: Enter manhole dimensions _____27*
*Figure 18: Select manhole shape _____28*
*Figure 19: Placing ducts onto the routeface _____29*
*Figure 20: Tool to assist user orientate when placing ducts_____31*
*Figure 21: Enter ducts specifications _____31*
*Figure 22: User interface for placing cables and joints _____33*
*Figure 23: Trace cable path using tree-view of cable network_____34*
*Figure 24: Error prevention in connecting nodes_____34*
*Figure 25: Make notes about the cable specification _____35*
*Figure 26: Enter cable specifications _____36*
*Figure 27: Data-entry for joint specification _____37*
*Figure 28: Joint Specification Number List _____38*
*Figure 29: Navigation menu _____38*
*Figure 30: Select an interaction context from a main menu _____39*
*Figure 31: Exit prompt _____40*
*Figure 32: Field data collection application loading _____41*
*Figure 33: Data sourcing map on a PDA _____42*
*Figure 34: Queries for street, asset and other attributes on the map _____43*
*Figure 35: Map attributes visible only at a particular range of scale _____43*
*Figure 36: Redline functionality of IntelliWhere OnDemand™ _____44*

# List of Tables

*Table 1: Comparisons of target platforms*_____*15*

# 1 Scope

## 1.7 Introduction

This document presents the design process and subsequent implementation of the user interface for the field data collection application. It also outlines the various phases of the design process and the design decisions that were made. This document is really a representation of the usability design of field data collection application which is embodied in the design of the user interface.

The document begins with a selection of the design methodology and description of the notation used. It then takes the reader through the design and implementation of the user interface. The user interface is implemented as a functional prototype on a mobile device. The usability features of the prototype are explained through illustrations of various UI screenshots. This document also describes the behavior of the user interface but not necessarily the underlying functionality of field application. For more information on the data model and detailed functionality of the field data collection application, see the Software Design Document (Doc. No. 4).

## 1.8 Purpose

The purpose of this document is to present the reader with the user interface design and discuss a number of features and characteristics of the user interface pertaining to the usability of field data collection prototype.

## 1.3 Audience

The intended readers of this document include researchers involved in the areas of software usability, mobile computing, as well as software designers, developers, the external examiner and other interested parties.

# 2 User Interface Design

## 2.1 Overview

The primary purpose of the user interface is to interact with the user in supporting tasks that the user is interested in doing. Some of these tasks are interrelated while others take place in a different context, and require different types of resources. Given the contextual nature of work, highly usable software should have a user interface that is organized to support tasks which are carried out in various **interaction contexts** (Constantine L *et al.*, 1999e). Essentially, each interaction context is an **interaction space** which contains all the tools and materials needed for carrying out a particular task or set of interrelated tasks. This section of the document specifies the interrelationships between interaction spaces within the user interface, as well as all the functions, data containers and information required in each interaction space.

The content of interaction space is represented in the **content model** followed by an **implementation model**. In the content model, the contents and organization of the user interface is determined without worrying about how it will look or how it will behave. As an abstract model, it leaves many options open, not only in appearance but also in the behavior of the user interface and its components. In an implementation model, however, each interaction space becomes a recognizable collection comprising parts of the user interface – a button, a combo-box, a dialog box etc. Here, the designer creates a visual design of the user interface by drawing upon knowledge of HCI, graphical design, widget selection, and layout to balance competing objectives and to trade off among conflicting constraints. The visual design does not only closely resemble the final implemented interface but it also acts like one. Effectively, a functional prototype is used to capture both the appearance and behavior of the final implementation of the system.

## 2.2 Content Model

The **content model** (Constantine L *et al*., 1999e) is an abstract representation of the contents of the various interaction spaces for a system and their interconnections. Each interaction space in the content model is populated with a collection of abstract tools and materials representing the content and capability that will be supplied to the user by the user interface. These **abstract components** are placeholders for actual visual components in the implemented interface. The tools supply the functions and active capabilities required to complete a task. The materials are the data containers, displays or work areas upon which the tools of the user interface can operate.

As would be expected, the content model is derived from the *essential* use cases contained in the task model (see section 5, Doc. No. 2). Each interaction space may support one or more use cases. The more closely two use cases are related or resemble each other, the more reasonable it is to support them with a common interaction context. The **use case map** (see Figure 13, Doc. No. 2), representing the use case relationships and narratives actually reveals the overlapping procedures, and guides in establishing which interaction space a particular use case belongs to.

Looking at the **dataSourcer** role, the **enterSourcingTeamDetails**, **enterSourcingTimes** and **enterSiteInformation** use cases take place in their own interaction space. They are not related in any way, each task requires the user to enter information that is specific and independent of other tasks. These tasks simply requires user to enter information that is requested. From examining the use case narratives, they need some type of container to hold the data that the user entered and a tool to insert data whether it is selecting item or entering text. This tool could be a Soft Input Panel (SIP), a button or a stylus. However, at this stage, no decision is made on the type of container or tool to be used. These decisions come later in the implementation model.




**Figure 17: Interaction contexts for entering sourcing team, sourcing time and site information.**

The interaction contexts for the three use cases described above are shown in Figure 1. The tools and materials in each interaction context are represented by Post-it notes using different colours. Use of Post-it notes offers several advantages over sketches or drawings (Constantine L *et al.*, 1999e). The fact that a Post-it note does not look like a Graphical User Interface (GUI) widget is a constant visual

reminder that this is an abstract model intended to leave many options open, not only in appearance but also in the behavior of the user interface and its components. Also, they can be easily moved around in reorganizing the user interface architecture, and this ease encourages experimentation and exploration (Constantine L *et al.*, 1999e). The green and yellow colours represent the passive and active components of the interaction context respectively. The active component includes tools and functions the user may use, whereas the passive components are static elements such as data containers or window frames. The vocabulary used to describe these components is deliberately kept abstract and general to accommodate a broad range of possibilities without falling into certain stereotypes or other preconceived solutions.

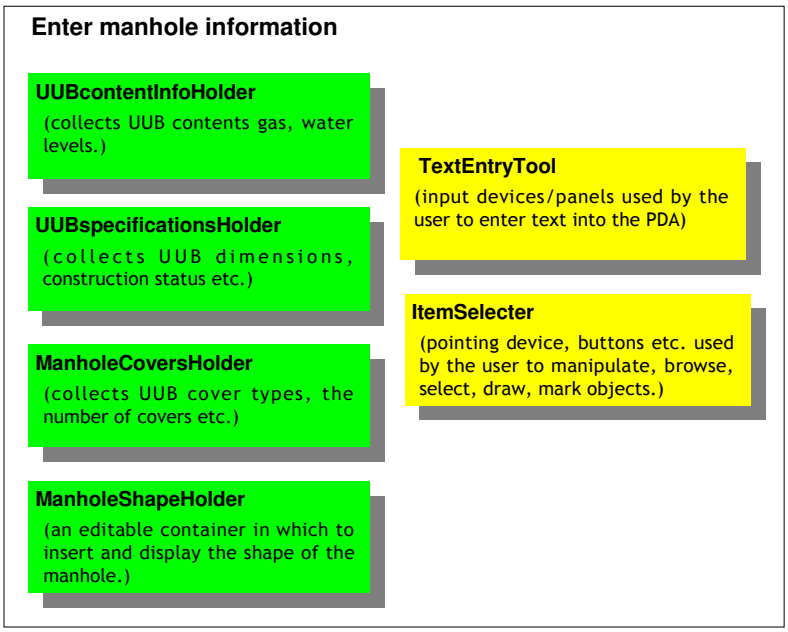



**Figure 18: Interaction context for entering manhole specification**

The next three tasks that follow involve sourcing information relating to the UUC. They are represented by the **specifyManholeCovers**, **specifyUUBcontent** and **specifyUUB** use cases. These use cases are closely related because they obtain

information from the same source (i.e. UUC) and naturally they would be carried out together as one coherent task. Hence, it makes sense to put the use cases in question into a single interaction context (see Figure 2). Generally, it is desirable to keep the number of interaction contexts to an absolute minimum by placing related tasks in the same interaction context. This is because every change of context requires a corresponding switch of thinking on the part of user. An excessive switching between interactive contexts may confuse first-time users and frustrate experienced users. The type of tools required to support these tasks is the **TextEntryTool** and **ItemSelecter** to insert data and manipulate objects respectively. As an example, the former could be a SIP or external keyboard and the latter could be a stylus to draw and select items. Moving from tools to materials, there would be two types of data containers. The one container holds textual data such as UUC specification and the other is a placeholder for the graphical data, in particular, the shape of UUC. The textual data containers could be a set of text fields and combo-boxes that user uses to insert, select where appropriate. However, the manhole shape could be drawn in a sketch pad or the user simply selects from a collection of pictures or other graphical representations of possible manhole shapes. Once again, the kind of tools and materials to be used is determined later in the visual design process.

Reading the use case narratives, there are a few use cases that are very similar. In particular, use cases that involve adding and changing (or editing) of specifications for ducts, cables and joints (e.g. **specifyDuct** is similar to **changeDuctSpecs**). These use cases require the same type of tools and materials which makes it appropriate to put them in the same interaction space (see Figure 3).

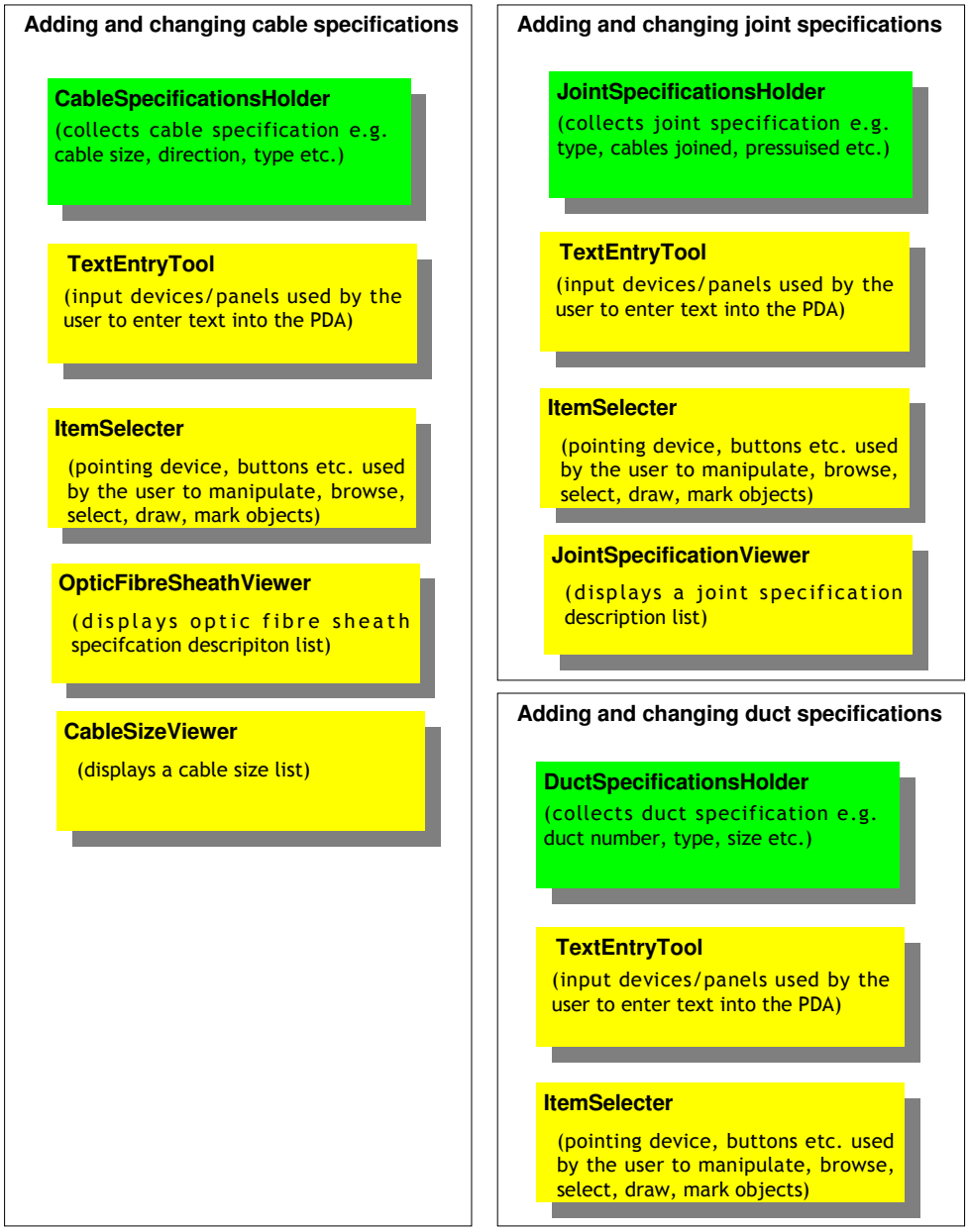



**Figure 19: Interaction contexts for adding and changing network node specifications**

The type of tools and materials required in these interaction contexts are typical of a form filling task. The **TextEntryTools** and **ItemSelecter** would be used to type in text and select item from a list respectively. Some kind of container would also be required to “hold” the data until the user is ready to finally enter data into the system.

As part of entering specifications, the user also needs to browse through the specification lists such as fibre optic sheath number, joint specification number and cable sizes. The tools to do this are **OpticFibreSheathViewer**, **JointSpecificationViewer** and **CableSizeViewer** respectively.

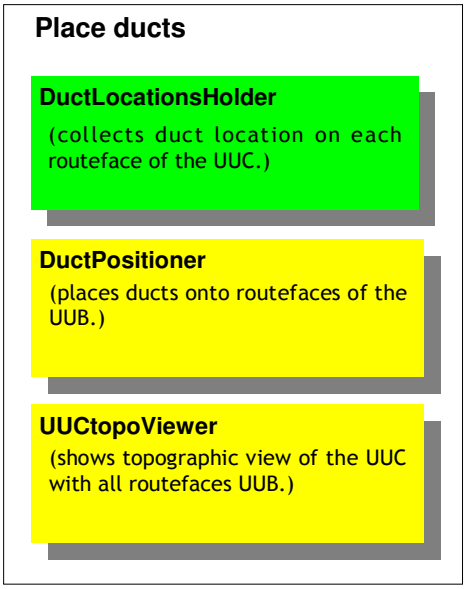



**Figure 20: Interaction context for placing ducts**

Also belonging to the same interaction context are the **placeDuct** and **changeDuctLocation** (see Figure 4). When placing a duct at a particular location on the routeface, the user is likely to make a mistake which may requires him to change the duct position to a new location. One could argue that these use cases belong to a single task because making errors and correcting them form part of any work. In order to support these use cases, the user must be provided with a topographic view of the UUC so that he can orientate himself in finding a suitable location for each duct. Bear in mind that there could be up to eight routefaces, each of them carrying about fifteen ducts. Once the user finds the right location, **DuctPositioner** tool is used to place ducts. And of course, a placeholder is necessary to “hold” and record position of each duct.

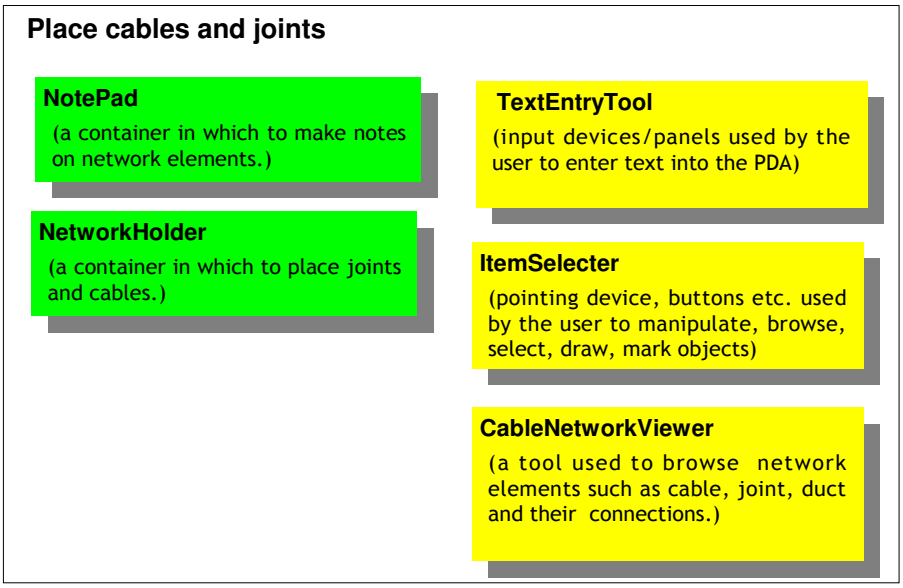



**Figure 21: Interaction context for placing joints and cables**

Once all the ducts are placed on the routefaces, information about the cable and joint connections are obtained. Unlike ducts, placing cables and joints to the network presents a more complex situation. The cable could be connected to a duct or to a joint. On the other hand, a joint could be connected to two or more cables. Adding a cable to the network includes adding all the joints that are attached to that cable (**placeCable** and **placeJoint**). A **NetworkHolder** could be used as a placeholder for the cables and joints (see Figure 5). A ItemSelecter tool may be used to point out a location for a node. Remember that when placing a node (cable or joint) the user also makes a short description about the node (**makeNote**). The **NotePad** container and **TextEntryTool** are necessary for this data entry work. The cable and joint network could be rather chaotic and complex especially when there are hundreds of nodes. So, the user needs to browse through the network to find a suitable location before placing a node (**browseNetwork**). As with ducts, the user often makes mistakes when placing cables and joints to the network and subsequently it needs to be corrected (**changeNetworkConnection**). In both situations, the user could

use **CableNetworkViewer** tool to view the interconnections of network nodes (i.e. cable, joint, duct etc.).

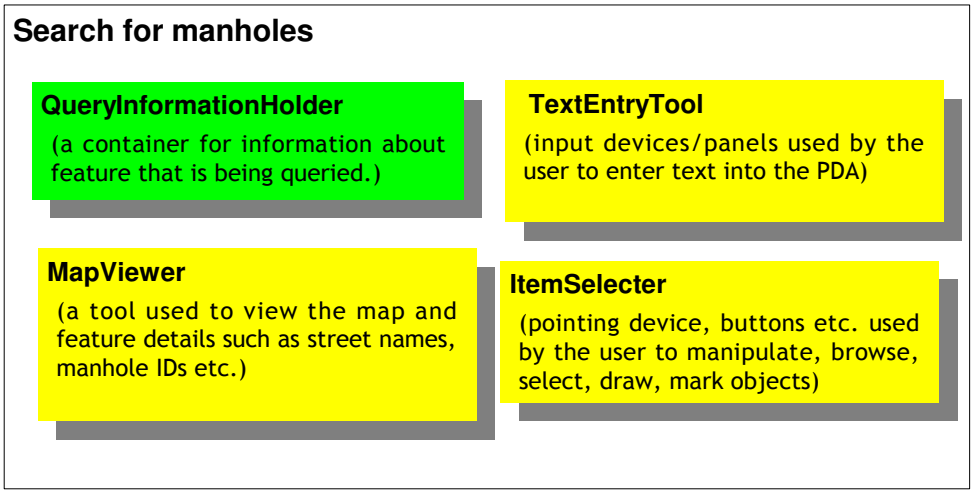



**Figure 22: Interaction context for finding manhole**

Moving on to the use cases for the **manholeFinder** role, the user would require a tool to view the map content such as streets and manhole. This tool is referred to as the **MapViewer** in the interaction context for finding UUCs (see Figure 6). The user would also need placeholders to insert details of the features (e.g. the name of the street or manhole ID number) on the map he is looking for. The **TextEntryTool** and **ItemSelecter** are also necessary to enter data and to select features on the map respectively.

The interrelationships between the various interactions contexts described above are represented in the **navigation map** (see Figure 7). The navigation map shows when the user would have to move from one interaction context to another in order to complete particular tasks. Essentially, the navigation map shows distribution of tasks across various interaction contexts and provides a rough overview of architecture and complexity of software. The notations used in the map include a rectangle representing any abstract interaction space and a *single* line with an arrow head which represents a transition between two interaction spaces. A *double* line

means that it is a context transition with implied return to the original context. Two arrowheads in the end of the line mean that the direction of transition is applicable both ways. Some of the context transitions are more common than others because they consist of tasks that are repetitive and more routine than others. The solid and dotted lines represent the most common and less likely type of context transitions respectively.

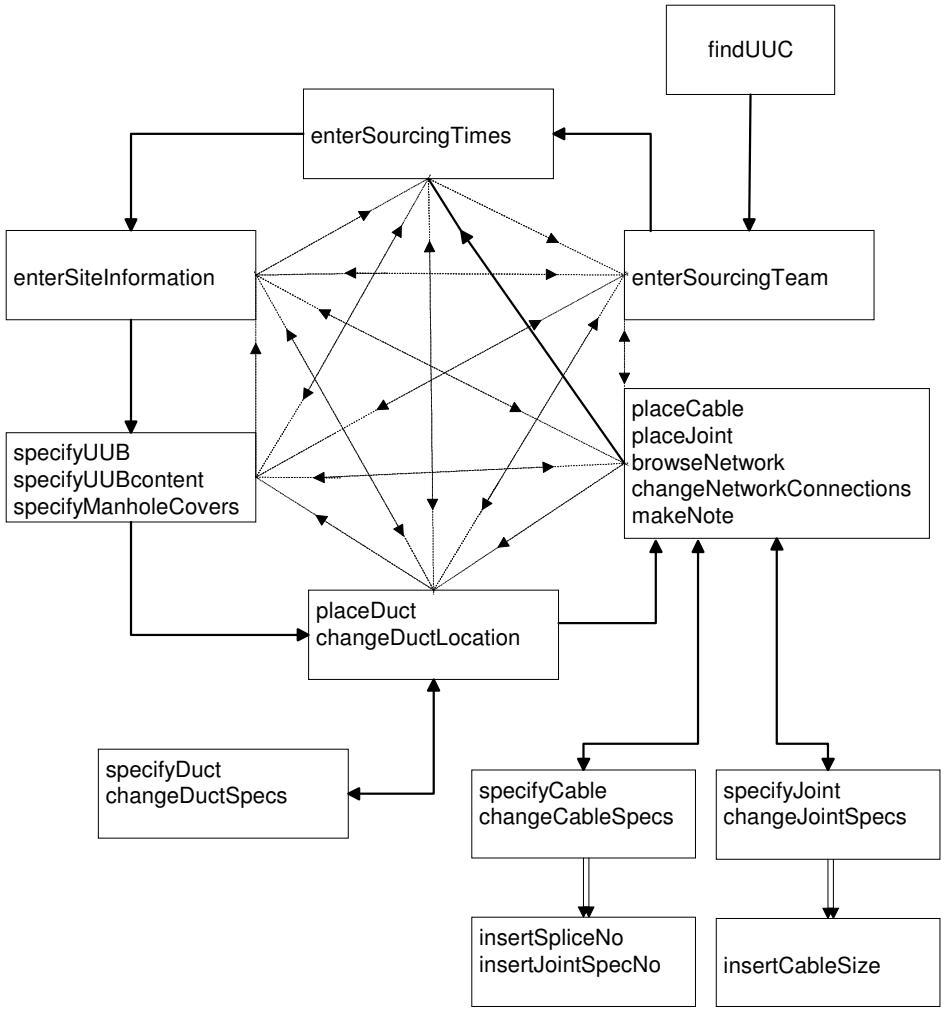



**Figure 23: Navigation map for interaction contexts**

As shown in Figure 7, the user would usually start off finding the location of the manhole to be sourced. Then the data sourcing begins with entering information about the sourcing team (**enterSourcingTeam**) and then he moves off to obtaining cable and joint information. Once complete, he goes back to **enterSourcingTimes** to enter the time the sourcing is completed. But because this work mainly involves form filling, the user has the liberty to enter data in the order that is most convenient to him. Also, due to the unpredictable nature of fieldwork, it is only fair that the user be given flexibility in the way he does his work. Hence, the dotted lines depict all the possible interaction context transitions that the user may make although they are relatively infrequent and less common. The more frequent and obvious context transition is when the user places network node (duct, cable or joint), followed by its specifications entry. This context transition is applicable both ways because from the specifications the user may also want to see where that node is place on the network. In contrast, inserting a particular specification number into the node specifications requires an implied return (see **insertSpliceNo** and **insertCableSize** interaction contexts in Figure 6), for example, after clicking a “OK” button.

## 2.3 Implementation Model

This phase of the user interface design process transforms abstract components in the content model into an **implementation model** (Constantine L *et al.*, 1999b), a prototype for the actual system specifying the layout of the user interface and defining the interaction between the user and the system. In doing so, three general classes of issues are addressed simultaneously. They are as follows:

1. Contexts – What are the implemented interaction contexts?
2. Components – What are the user interface components within contexts?
3. Composition – What is the layout and organization of components in each interaction context?

The first question asks how each abstract interaction space will be embodied in the user interface as an actual interaction context – for example, whether as a screen, a window, a dialogue or the page of a tabbed dialogue. In the second part of the transformation, particular user interface components or GUI widgets replace the abstract materials and tools in the content model. The goal here is to ensure simpler and easier operation. Again, conventional GUI components are not always suitable and alternative solutions are sought. The visual components are then organized appropriately so that user interface layout as a whole supports smooth and efficient workflow. It is worth noting that the translation is not a straightforward process even though the rule may be well defined. To some extent, a good translation is both creative and artistic.

The implementation model is a representation of what the final implemented user interface will look like and how it will function. Most commonly, it is a sketch or some kind of drawings with supporting notes and documentation. But this type of passive or non-working model has difficulty in explaining how it actually works because of its static nature. On the other hand, a functional or working prototype is more convincing and easily understood than a drawing that has to be explained in details. A working prototype is better suited when the implementation model is used for

proof of concept. In particular, situation when the quantitative measurements of usability issues such as HCI rate, efficiency and system response time etc are required. In general, functional prototypes can be used to test these characteristics far more effectively with users than their static counterparts. Based on these arguments, the UI is implemented as a functional prototype on a mobile device. The sections below illustrate the implementation model of user interface for the field data collection application. But before doing so, the target platform for the prototype is chosen.

### 2.3.1 Target Platform

The first step in building a functional prototype is to determine the target platform. The target platform is the hardware or the mobile device on which the field data collection prototype will be implemented. Three types of mobile devices, namely a tablet computer, a Personal Digital Assistant (PDA), and a mobile phone were evaluated (see Table 1) to see if they meet the hardware requirements specified in section 9, Doc. No. 2. But having said that, the boundary between these devices is not strictly defined because mobile technology and computers are fast converging (e.g. hybrids that incorporate features of all these mobile devices). Also, hardware features described in Table 1 are not found across all the mobile devices in a particular category. In general, most of these features are available only for the top-end segment of the product range.

**Table 1: Comparisons of target platforms**

| Requirements | Mobile Device | | |
|---|---|---|---|
| | **tablet computer** | **PDA** | **mobile phone** |
| 1 | No, particularly when it is ruggedized. | Yes, it fits in the palm of a hand. | Yes, it fits in the palm of a hand. |
| 2 | Yes, ruggedized | Yes, rugged | No, protective |

| | | | |
|---|---|---|---|
| | tablet computers available for field use (Symbol, 2003). | casings available from third-party vendors (Otterbox, 2003). | covers are not designed for field use – they are not really shock-proof or water-proof etc. |
| 3 | Yes. | Yes, light sensors that automatically adjust the screen backlight to the surrounding light condition. | Yes. |
| 4 | Yes, ½ VGA. High resolution and full colour display. | Yes, ¼ VGA. High resolution with full colour display. | No, 1/8 VGA. Low resolution with full colour display. |
| 5 | Yes, internal dial-up modem, WLAN or LAN network cards. | Yes, expansion cards for GSM, GPRS, Bluetooth and WLAN. | Yes, GSM, Bluetooth and GPRS capability. |
| 6 | Yes. Hard drives (internal/external) with high data storage capacity. | Yes, add-on memory cards for extra disk space. | No. |
| 7 | Yes, touch pad, touch screen and hard keyboard. | Yes, touch screen, external add-on keyboard, handwriting recognition software and soft keyboard. | No, numeric keyboard makes text data-entry difficult and inefficient. |
| 8 | No, the best batteries last for 4 or 5 hours. | Yes, last for more than 10 hrs. However, | Yes, battery lasts for a few days, even weeks. |

| | | | |
|---|---|---|---|
| | However, extensive use of screen lighting and some attachments may significantly decrease the battery life. | extensive use of screen lighting and some attachments may significantly decrease the battery life. | However, extensive use of screen lighting may significantly decrease the battery life. |

Information in Table 1 identifies a PDA as the most suitable mobile device followed by the tablet computer. The table, however, doesn't show a significant difference between them, particularly in terms of cost. Typically, a tablet computer is five times more expensive than a PDA. Hence, it is far more feasible to deploy PDA as a mobile terminal in large numbers in the field than the former. Based on these arguments, the PDA was selected as the target platform for the field data collection application. The type of PDA chosen is a iPaq™ H3000 (see Figure 8) from Compaq™ (now HP™).




**Figure 24: iPaq H3900 PDA**

Important features of the H3900 includes (see Doc. No. 6 for full technical specifications):

- Transflective TFT LCD with 65,536 colours, resolution: 240 × 320.
- Rugged case for industrial use.
- Expansion packs for WLAN, GSM/GPRS networks and barcode scanner.
- Lithium Polymer rechargeable battery with up to 14 hours operating time
- SD slot for 64 MB, 128 MB and 256 MB memory cards.
- Input methods: External micro-keyboard, Handwriting recognition software, soft keyboard, character recognition, voice recorder and inking.

A conscious decision was taken to use a Pocket PC platform primarily because it supports the most sophisticated functionality pertaining to mobile field data collection applications. Some of the functionality includes GPS, wireless network connection and data compatibility with most widely used Windows™ applications. Not surprisingly, most of the leading mobile GIS software suppliers (e.g. Trimble, ESRI, and Intergraph etc.) support Pocket PC platform. The disadvantage of using this platform, however, is that it uses information presentation techniques and metaphors derived from its desktop computers. The implication is that these desktop metaphors rely primarily on video for communication and that demands a very high level of user attention (Kristoffersen S *et al*., 1999 and Pascoe J *et al*., 2000). There are, however, a number of solutions that have being proposed to overcome this usability shortcoming (see section 3.3, Doc. No. 1).

Embedded Visual Basic (eVB) was used to program the field data collection prototype. The rationale for this decision is that eVB is most likely to be used by programmers because of its simplicity and it allows rapid prototyping of the user interface. Overall, this project has taken a pragmatic approach in selecting the platform and programming tools that are most likely to be used so that the usability design of the field data collection application can be applied in a realistic software development environment.

### 2.3.2 Implementation Model illustrated

The implementation model is illustrated using screenshots taken from the user interface of field data collection prototype. These designs have gone through a number of iterative inspections and they are representative of a final design as opposed to a first-cut design. They are not presented as definitive or flawless solutions but rather as a platform to discuss problems and tradeoffs involved in designing a usable user interface. Moving from the content model in section 2.2, the **SourcingTeamInfoHolder** container is now replaced by a listview (see Figure 9). The intention is that the user enters the names once and thereafter only selects the names of the sourcing team from a list of pre-existing team members that are stored in system memory. It is unnecessary for the user to type in all the member names and their details for every UUC they source. It is far quicker and easier to select from a list than to type in the names. A shortcut button for selecting all the list items is also used to increase the efficiency of the task at hand. A multi-select enabled listview allows the user to choose any combination of members in a single selection. The **itemSelecter** used here would be a stylus. The user may use a finger for the larger targets such as the “Select All Members” button. Unfortunately, due to lack of screen space, not every target can be made large enough for the use of finger. Research has shown that sonically-enhanced targets or visual components can increase usability despite their small sizes (Brewster S, 2002). The “Add” and “Cancel” buttons are relatively small targets and hence they are enhanced with audio feedback.

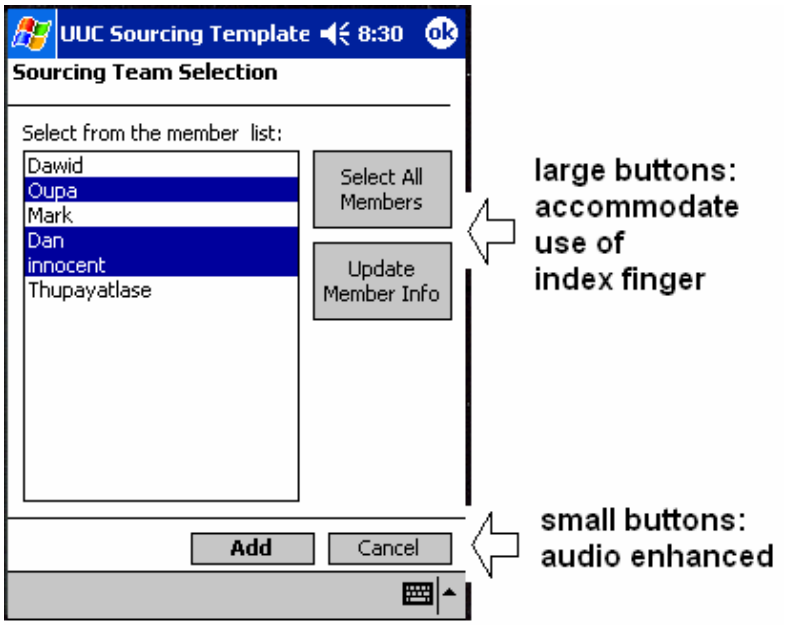



**Figure 25: Select sourcing team members**

Information about the sourcing team needs to be updated occasionally, for example, when a new member joins the team or making some changes to the information about an existing member. By clicking the **Update Member Info** button, the user gets access to detailed information about all the team members (see Figure 10). The information is displayed graphically in a tabular table format, which is suitable for a quick browse. Because the table has only three columns, all of them are visible and hence there is no need for the user to scroll horizontally to view the rest of table. This is important because the horizontal scrolling in small screen like that of a PDA reduces readability and comprehensibility of data (Laarni J, 2002). The user has the options to create a new entry, delete or edit an existing entry by clicking a **New**, **Delete** and **Edit** buttons respectively. The **Edit** and **Delete** buttons are enabled only when the user selects a particular team member from the table. The user is presented with a new screen (see Figure 11) when he is adding or editing a member information although these tasks can be done directly from the table. The intention was not to rely on hidden features of the table widget but instead to make their functionalities appear obvious to the users who have little exposure to Windows UI components.

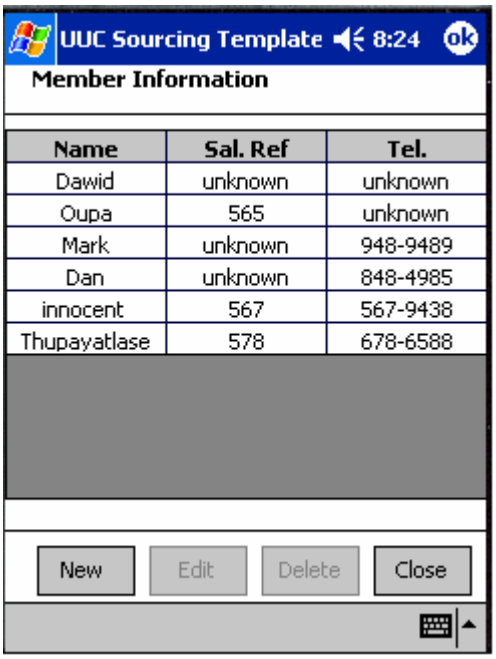



**Figure 26: View sourcing member information**

It is recommended that user does not use external keyboard or any other external input devices attached to the PDA to enter text. An external keyboard, for example, is awkward to use without a flat surface (not available inside the UUC) on which to place it. Hence, SIP is assumed to be a primary means of entering customized data into a PDA. Information is entered into the input panel in one of two ways; by tapping a miniature keyboard (also called soft keyboard) with the stylus to enter characters or by writing on a handwriting recognition panel using a stylus. The user can switch between the two modes at any time. The users who are familiar with the use of a standard desktop keyboard will find a soft keyboard easier to use than handwriting recognition software. The latter does not always accurately recognize letters and it takes a bit of practice to get it right. Although it may appear trivial, the behavior of SIP affects the usability of the mobile application. In this particular interaction context, making SIP appears automatically when the text field receives focus and removing it after clicking the 'OK' button makes the data entry work faster and more pleasant. Otherwise, the user has to repeatedly click on a keyboard icon at the bottom of the screen to activate and deactivate the SIP which can be inconvenient and time consuming. Also, when the SIP is opened it should not

conceal the textbox where the data is entered. In this particular screen (see Figure 11), however, there is enough space at the bottom of the screen for the SIP to appear.

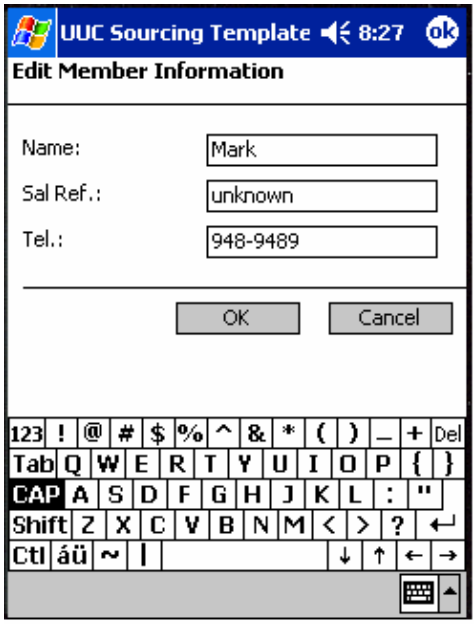



**Figure 27: Update sourcing member information**

The **Times&DatesHolder** container in the **Enter sourcing time & date** interaction context is implemented using a textbox (see Figure 12). Even so, the user is not expected to type in the information. The sourcing dates and times are most likely to be the same as the date and time at which user enters the information. Instead of typing in the date and times, the user clicks a checkbox and the system automatically inserts the current time and date from the computer memory. This *minimal input action by the user* principle is also applied to entering sourcing and traveling duration where the user selects from a list of approximate times from a combobox. In the case of sourcing duration, the system automatically calculates the time, but the user also has the option to override the automation.

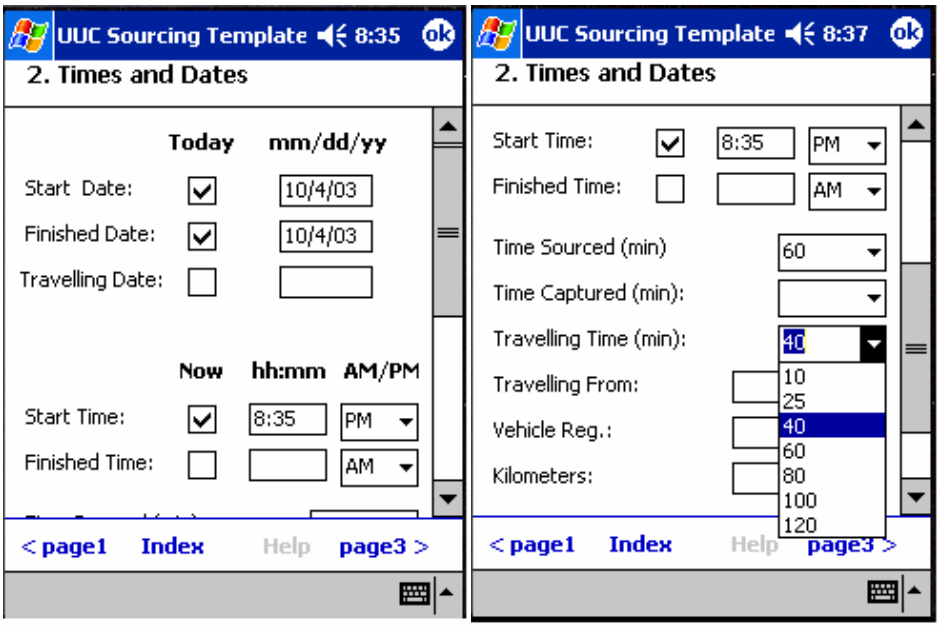



**Figure 28: Data entry for sourcing times and date**

If, however, the user decides to type in the information, the correct formats of data-entry are clearly displayed to guide him. The data fields reject the data that the user enters if the format is incorrect. A 12-hour clock with AM/PM designations is used for the time format, which is more comprehensible than a 24-hour time format (Shneiderman B, 1998c). The data containers that hold related information (e.g. start time, finished time etc.) are neatly grouped together. A fair amount of space between each group of components prevents UI from looking overcrowded. Also, they are aligned to make the user interface look tidy and visually appealing.

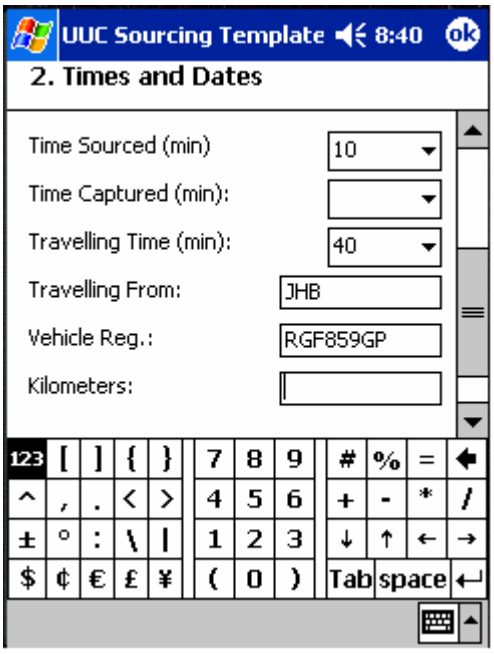



**Figure 29: Form and scrollbar resized when SIP is opened**

The **Enter sourcing time & date** interaction context (see Figure 1) is implemented as a single screen but not all the UI components within this context can fit into PDA screen at once. Hence, a vertical scrollbar is used to access the data containers which otherwise are not visible. Ideally, one would not use a scrollbar but without it the information will be spread over many separate screens resulting in an excessive context switching. However, if one needs to scroll, a vertical scrolling is the most suitable type of scrolling for a PDA (Laarni J, 2002). It is particularly effective when the user drags the scrollbar using the stylus (as opposed to a mouse for example) provided that the length of scrolled page is not too long.

As before, SIP is used as a primary **TextEntryTool**. Once again, opening one partially conceals the data fields and prevents the user from entering data. In order to solve this problem, the form is shifted upwards and the scrollbar is resized whenever the SIP is opened (see Figure 13). Both the stylus and external buttons are used as the **ItemSelecter**. For example, the “navigation” button (also called “directional pad”) of a PDA can be used to scroll the form or even browse through items in the combobox. The “record” button can also be programmed to use as a

scroll button although it works only for a downwards scroll. But this button needs to be pressed with quite a bit of pressure in order for it to work, and this could become awkward for a “fast scroll”. To a similar effect, clicking “up/down” buttons on the scrollbar is cumbersome due to their small size. The use of navigation button and dragging the stylus along the scrollbar works equally well especially for a “fast scroll”. The stylus allows a rich set of gestures on the touch screen and they too are exploited for scrolling. In particular, dragging the form directly works reasonably well provided the form does not contain too many GUI components.

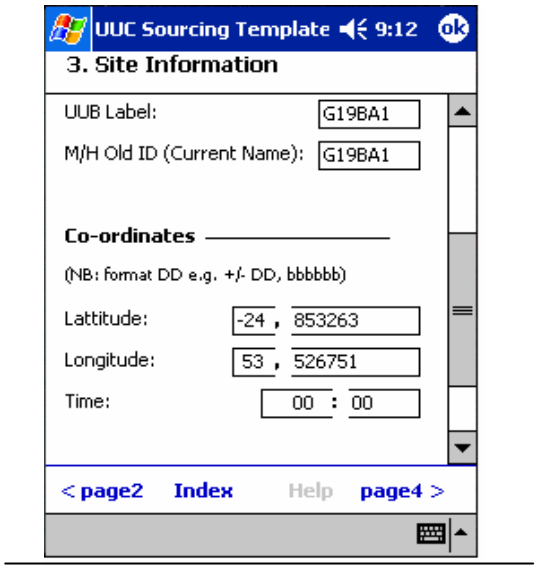



**Figure 30: Enter site information**

Moving on to the **Enter site information** interaction context, the textboxes are used as the **SiteInformationHolder** (see Figure 14). These containers are automatically prepopulated with data that is read from the data-sourcing map. Of course, the user may have to type in custom data if the UUC has changed its name or location but this seldom happens. This type of automation helps to reduce user’s workload and potential for errors. The format and organization of visual components are similar to previous interaction context where components that are related to each other are grouped together while keeping aesthetic apprehension of the user interface. The intention was to keep consistency so that user can get accustomed to the format and organization of the user interface.

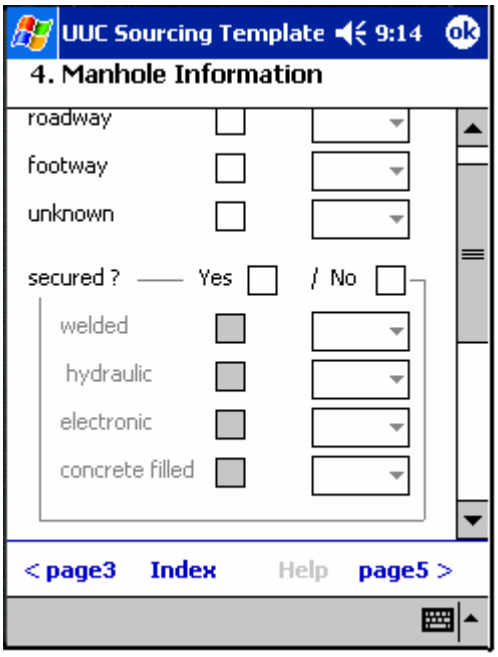



**Figure 31: Specify manhole covers**

In the **Enter manhole information** interaction context, a wide variety of visual components and widgets are used so long as they increase software usability. The **ManholeCoversHolder** is made up of a combination of checkboxes and comboboxes (see Figure 15). The user must check the checkbox in order to enable the combobox. However, users who are not familiar with these widgets may not be aware of when they are disabled. By “graying out” the components that are disabled, the user can distinguish them from active components. The user can confirm this by clicking on them, which will result in a sharp beep indicating that they are inactive. This fool-proofing mechanism prevents errors from occurring in the first place, which is better than correcting them later on. The comboboxes are prepopulated with possible numbers of manhole covers so that the user does not have to type in the information. As with the manhole covers, the data containers for the content of UUC are enabled only when they are applicable (i.e. “time pumped” is valid only when UUC contains water) to prevent user errors (see Figure 16).

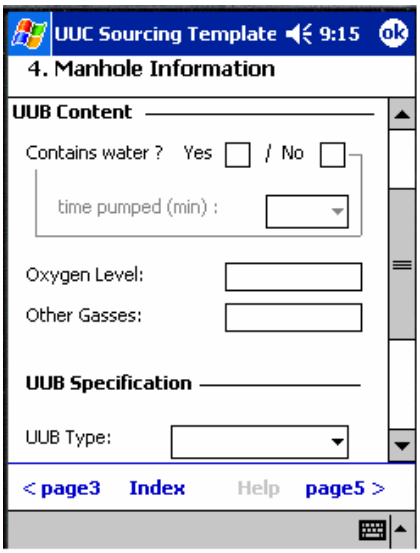



**Figure 32: Specify manhole content**

Unfortunately, the data containers cannot always be made foolproof. Such was the case with the data entry for the UUC dimensions (see Figure 17). Here, the user types in the dimension using SIP, which the system validates on entry. Otherwise, the rest of the data including manhole type and construction status are simply selected from the combobox.

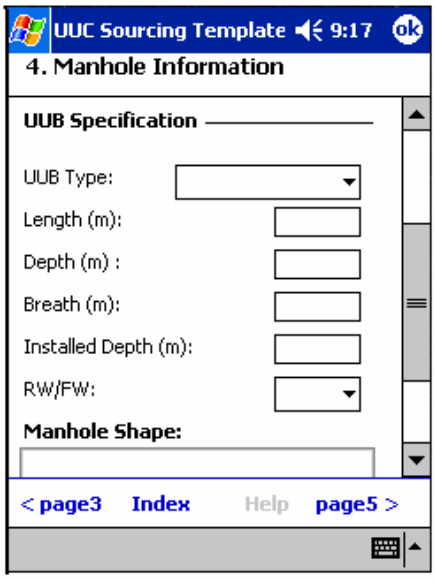



**Figure 33: Enter manhole dimensions**

A picture box is used as a **ManholeShapeHolder** from which the user selects the shape of the manhole from a predefined list of images (see Figure 18). It is far easier, quicker and less error-prone if the user selects the shape from a list than to draw it out. However, as the number of shapes in the list increases, so is the time to find the shape that the user is looking for. So, the number of possible manhole shape should be categorized or limited to a few items that the user needs to browse. This, of course, will depend on the number of manhole shapes and their variations. The image of a manhole shape does not require a high level of details and hence they can be stored in relatively small image files (a few kilobytes in a GIF file format). This is necessary for performance (e.g. to reduce system loading time, response time etc.) of field application which to a large extent affects the usability.

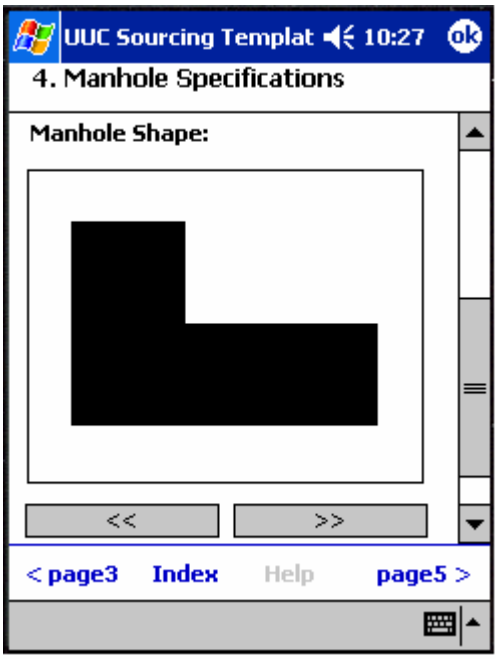



**Figure 34: Select manhole shape**

Moving on to the **placeDuct** interaction context (see Figure 4), the **DuctLocationsHolder** is realized using a frame containing a number of 'x' labeled buttons (see Figure 19). The 'x' represents a possible location or position of a duct on the routeface. The locations of the ducts are displayed graphically which is quicker and easier for the user to read and understand than lengthy text messages;

"*a picture worth a thousand words*". The users should find the graphical presentation highly intuitive given the obvious resemblance of the **DuctLocationHolder** to what they actually see on the routeface. In order to place a duct onto the routeface, user first checks the "Add" option-button and simply clicks on 'x' label where he wants to place the duct. Once the location is selected, the gray coloured 'x' label transformed into a square block labeled with the duct number. The colour transformation is accompanied by an audio feedback indicating a new duct has been placed. The user interface is designed to exploit the direct control style of user interaction that the user has when he uses the stylus to manipulate objects on the touch screen. The user has the options of adding, removing and specifying ducts by checking appropriate optionboxes. The efficiency of these tasks also increases when the user does not need to change the mode of operation (add, remove or specify) for every duct especially when a large number of ducts are placed at one time. In order to remove a duct, the user simply checks the "Remove" optionbutton (systems automatically unchecks the other two radio buttons) and taps on the duct that needs to be removed.

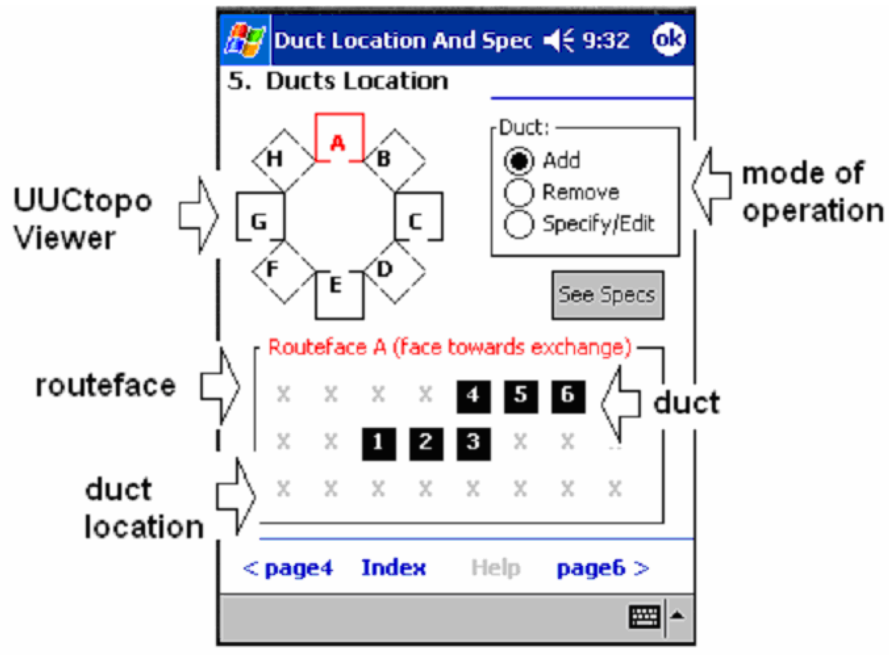



**Figure 35: Placing ducts onto the routeface**

When placing a duct, the user needs to know not only its location on the routeface but also that routeface's relationship to the other routefaces, particularly to that of routeface A. Unfortunately, the PDA screen does not have space to show locations of ducts on all the routefaces at the same time. But it can still create the same effect by using **UUCtopoViewer** (see Figure 4) which essentially shows which routeface the user is working on and where it is in relation to the rest of routefaces. This container is implemented as a dynamic (as opposed to static) image of topographic view of the UUC (see Figure 19). In other words, it interacts actively with the user by displaying the content of each routeface that is selected. In addition, it "rotates" in an anti-clockwise direction so that the routeface that was selected moves to the top location. This routeface "points" towards the wall of the UUC which the user is facing when he is carrying out the data sourcing tasks. In other words, the tool helps the user to find a sense of direction and orientation inside the UUC. For example, if the routeface C is selected (see Figure 20), the UUCtopoViewer rotates accordingly so that the routeface A and routeface E appear on the left hand side and right hand side of the user respectively. Generally speaking, to secure a sense of direction and orientation is of paramount importance to mobile applications that are used in a true mobile environment.

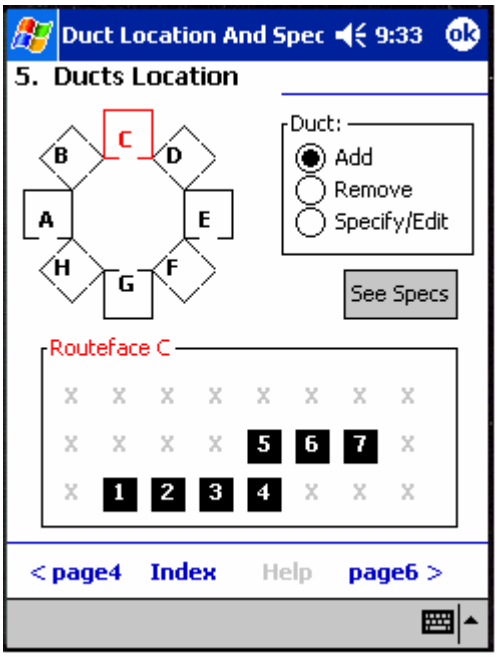


**Figure 36: Tool to assist user orientate when placing ducts**

When specifying the ducts, the user needs to know exactly which duct he is specifying. Although the name will identify the duct, it is more accurate and reassuring to "see" its location as well. This reduces the user's workload in going out to find where he has placed the duct. In order to specify the duct, the user first checks the "specify/edit" optionbutton and taps on the duct that he wants to specify. While the user is entering ducts specifications he is able to see the duct (highlighted in red colour) that is currently being specified (see Figure 21). Once the ducts are specified, they are highlighted in green colour to distinguish from those ducts that have not yet specified. Again, this reduces the amount of user attention that the system demands by providing constant feedback and helps the user keep track of the changes. If the user wants to change the specifications he would follow the same step as adding specification (both tasks belong to the same interaction context as shown in Figure 3). The only difference is that data fields are pre-populated with previous specification values.

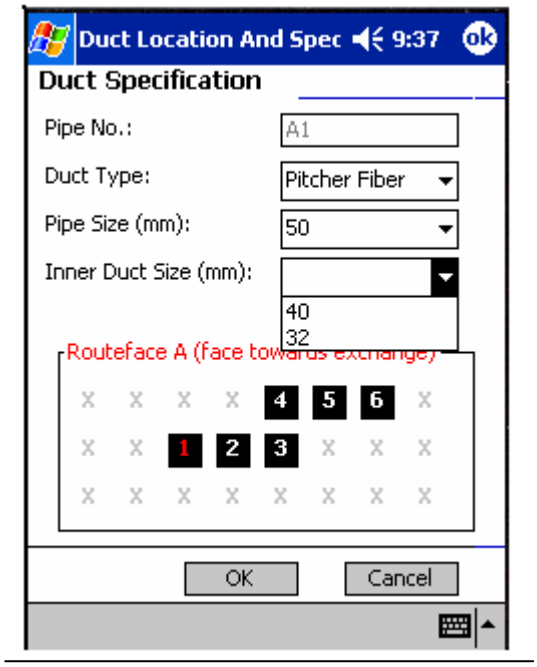



**Figure 37: Enter ducts specifications**

It is important to note that there is no menubar, pop-up or nested menu items. The intention behind this design decision was to make all the functionalities that the user needs immediately obvious and readily available without looking for them. This design approach is also referred to as What You See Is What You Need (WYSIWYN) and it is particularly effective for task-oriented applications without too many complex functions.

In the **Place cables and joints** interaction context, a tree widget is used as a template to view the cable network (see Figure 22). Looking back at the content model (see Figure 5), this component serves both as the **NetworkHolder** and the **CableNetworkViewer**. A tree-view of the network shows the interconnections of cables, joints and ducts but without the locations of ducts on the routeface. So, another container displaying the ducts works with the **NetworkHolder** in a synchronized manner to give a complete "picture" of somewhat complex cable network. The tree-view represents the direction of the cable path rather than the actual physical connections. This means that if the cable is located at a particular place, it is unique and it is not repetitively shown on other parts of the tree-structure. This way the user can create a mental map of the network connections without going into backward traversals and getting lost.

In order to trace the cable path, the user taps one of the panels on the tabbed pane ("routeface selection" in Figure 22) to select the routeface, he then clicks one of the ducts on the routeface. The **CableNetworkViewer** responses by displaying the full path of cables and joints connected to that duct. Alternatively, if the user clicks the cable (or joint) from the tree-view, the system shows the duct to which they are connected by highlighting it in red on the routeface panel. There are a number of reasons why this kind of automation is necessary. Firstly, there is insufficient space on the screen to show all the cable connections simultaneously. And secondly, it would be time consuming, error-prone and confusing for the user to look for cables in a large network. The user, however, is not expected to know how the tree widget works. Again, the goal is not to rely on hidden features on this widget but to make

them appear intuitive to the user. For example, the tree nodes are automatically expanded to display the full cable path without any user action (see Figure 23).

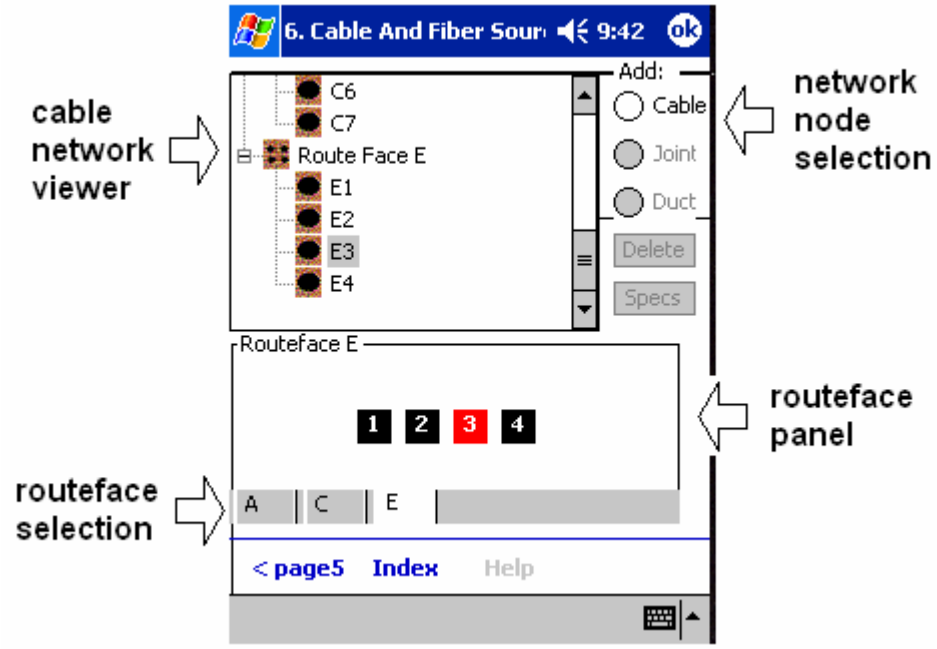



**Figure 38: User interface for placing cables and joints**

The tree-icons help the user to recognize what each node represents. Unfortunately, there are no standard icons for cables and joints, so they were custom designed. The objective was to make icons appear simple, yet “guessable” so that the user can easily draw connection to what they represent. For example, the icon for a duct is represented with a black circle with a brown background, which can be associated to a hole in the wall. Because these are small icons, they cannot contain a high level of details. The icons could be made larger to improve clarity but this is at the cost of reducing the visibility of rest of the network, which is not desirable.

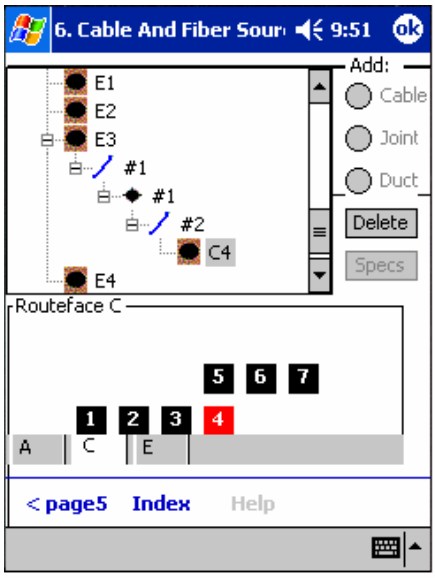



**Figure 39: Trace cable path using tree-view of cable network**

The user can place and connect cables, joints and ducts onto the network by selecting appropriate optionbuttons on top right corner of the user interface (see Figure 23). Of course, these nodes cannot be connected in any random order. For example, the user cannot connect a joint to a duct or one end of cable to more than one joint although a joint can be connected to any number of cables.

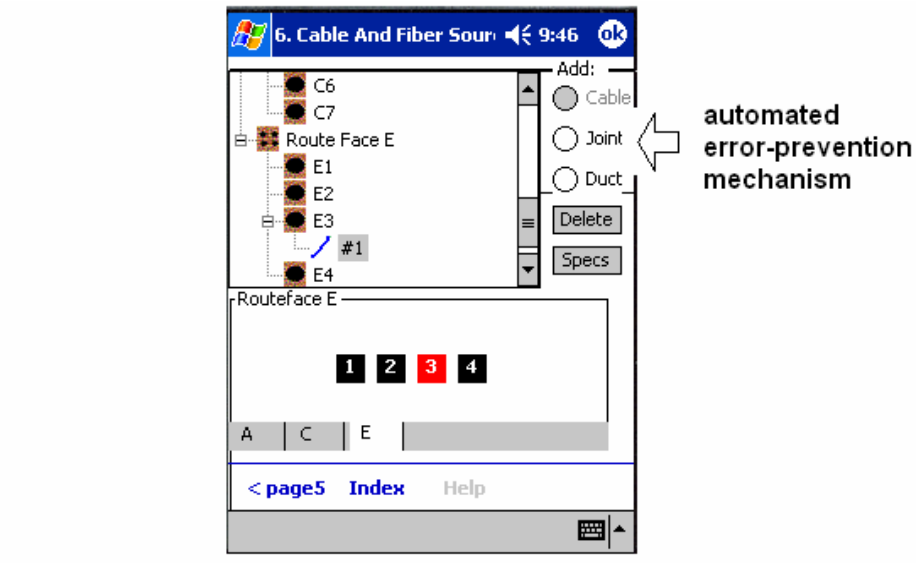



**Figure 40: Error prevention in connecting nodes**

The system automatically implements these rules by disabling optionbuttons that are not applicable for a particular type of connections. For example, when the user selects a joint from the tree structure, the optionbutton for adding duct is grayed out or disabled. This automation prevents unnecessary errors and improves data accuracy.

When adding cables to the network, they are automatically given a unique identification number (see Figure 25). The user selects whether the cable type is a Copper or Fibre Optic. The **NotePad** (see Figure 5) is implemented as a large textbox where the user makes short notes. When the user clicks the **NotePad** container, the SIP automatically opens. He then makes notes about the cable or he can click the “unknown” button to insert the words “unknown” into the **NotePad** which is quicker than typing in the words. The SIP automatically closes when the user clicks the “OK” button when the task is complete. Again, use of appropriate SIP behavior may seem trivial but it allows the user to complete the data entry task fairly quickly.

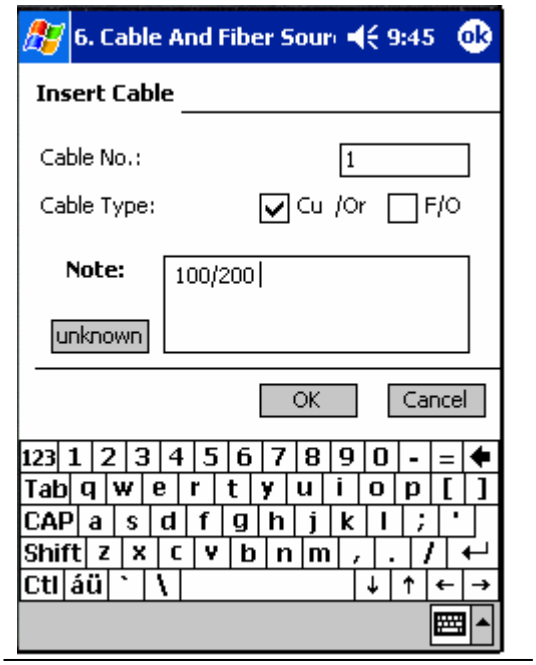



**Figure 41: Make notes about the cable specification**

The tables are used as the placeholders for the specifications of the joints, cables and ducts. Unlike the latter, the joint and cable tables have a large number of columns and not all of them fit into the screen (see Figure 26). Hence, the user needs to scroll horizontally to see them, which is difficult, awkward and ineffective in locating data fields. The table columns could be made resizable or even smaller to see more columns in a limited screen space. However, the reduction in column width conceals the name and the content of each column leading to confusion and incomprehension.

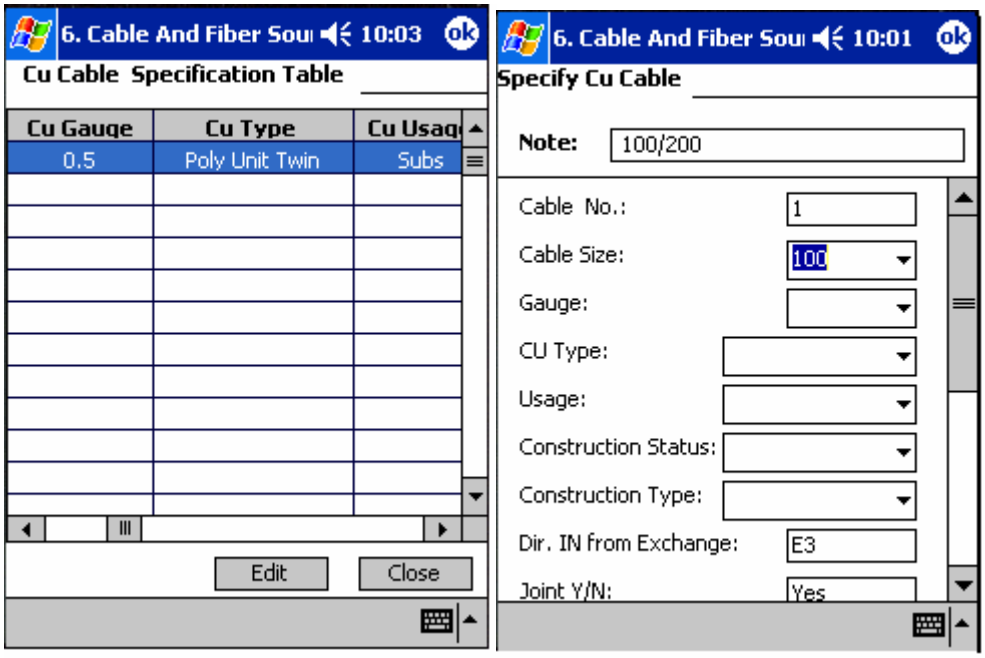



**Figure 42: Enter cable specifications**

In general, a horizontal scrolling is considered to be inappropriate for a PDA screen (Cooper A, 1995). As a result, the data containers for joint and cable specifications are displayed in a vertical format (see Figure 26 and Figure 27). Here, the user can see almost all the data containers where he enters the specifications. Some of the information is prepopulated by the system so as not to repeat tasks. For example, whenever the user places a cable onto the network, the system keeps track of its ID, direction (i.e. source and destination) and whether it has joints. Using the paper sourcing forms, the user would have to repeat such tasks leading to inefficiencies.

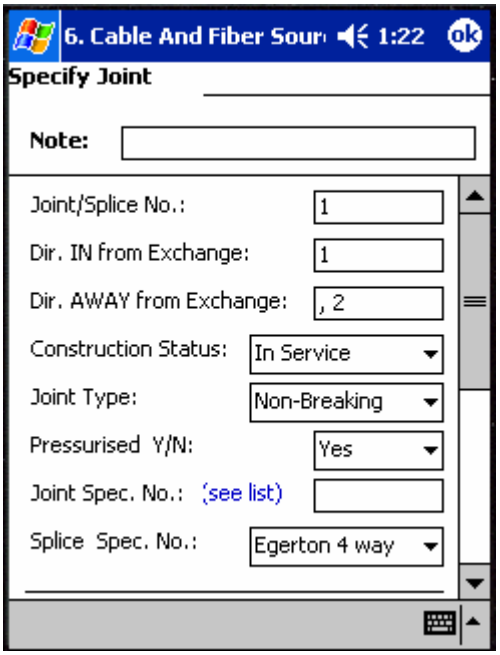



**Figure 43: Data-entry for joint specification**

The specification lists are displayed when the user clicks on the “see list” text (see Figure 27). The text is highlighted in blue to indicate that it is a navigation link that goes to another page (the blue colour is reserved for navigation controls, more on this later). The short notes now appear on the top the page inside another frame so that the user can refer to it at all times. The specification table fits well into the small screen because they have a small number of columns. The user is able to read its content comfortably without horizontal scrolling. The specification lists are lengthy (up to 50 items) which require user to do quite a bit of vertical scrolling before finding the right specification number (see Figure 28). But by putting together specifications that belongs to the same type of node (e.g. dome joint, foam joint etc) and by arranging them in an alphabetical order makes the searching easier and quicker.

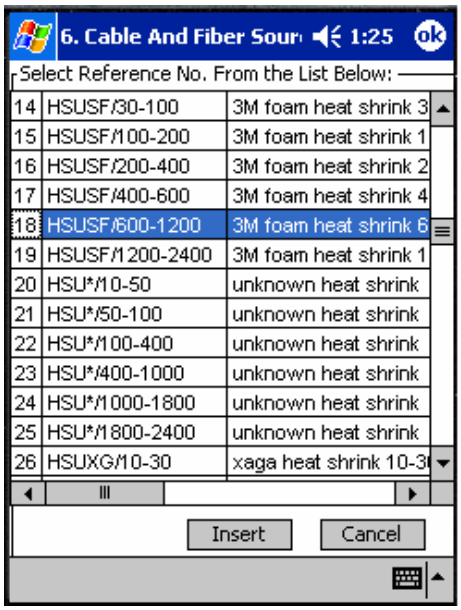



**Figure 44: Joint Specification Number List**

The overall architecture of the user interface is based on the navigation map (see Figure 7) of the interaction contexts. The user is given a high level of flexibility in navigating between interactions context and complete tasks in the order that is most convenient to him. The user is provided with the navigation menu where he selects a particular interaction context (see Figure 29). The word “page” is used to draw analogies between page turning in sourcing forms and use of navigation controls to move between different interaction contexts. This kind of association would be particularly convincing for inexperienced computer users. Keeping the number of interaction contexts to a minimum and numbering them helps the user with navigation and prevents getting lost. The system also encourages the user to follow the most common pattern of navigation by providing each interaction with a link to the previous and next interaction contexts.

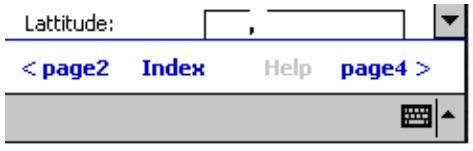



**Figure 45: Navigation menu**

Each screen has a title describing the name of task or subtask that user is doing. The titles are written in bold letters to indicate that they are in fact the page titles. Again, these page titles are motivated by the page-turning metaphor used for navigation between various interaction contexts.

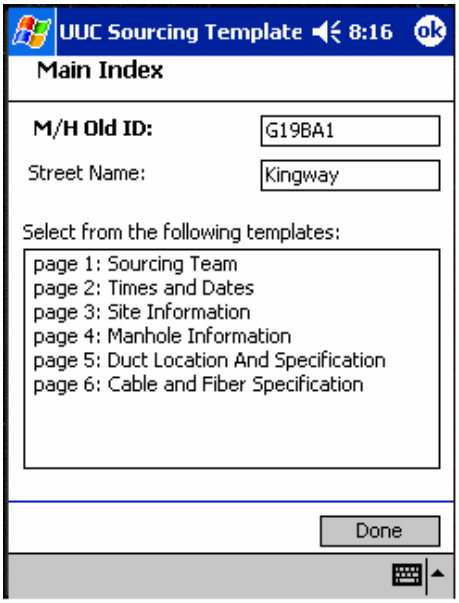



**Figure 46: Select an interaction context from a main menu**

The navigation controls are coloured in blue and they are placed at the bottom of the screen just above the menu bar (see Figure 30). The blue colour is used exclusively for embedded links. The colour and layout are kept consistent throughout all interaction contexts. The navigation controls include the following embedded links:

- "< page" – goes to a previous interaction context
- "Index" – goes to the main menu (see Figure 30)
- "Help" – to access specific helpful information. This feature is disable if there are no help available.
- "page >" – goes to the next interaction context

Once the data sourcing templates are filled in, the system prompts the user to either save the data, exit without saving or return to data sourcing work (see Figure 31). If the user selects to save data, the data is saved onto the PDA and the sourcing work is complete for that particular UUC.

An important aspect of software usability that does not feature in the user interface design is the performance of the field data collection prototype, particularly system response time. Obviously, the longer the user waits for the system to response to user action, the more likely the user becomes frustrated and the less efficient the work becomes.

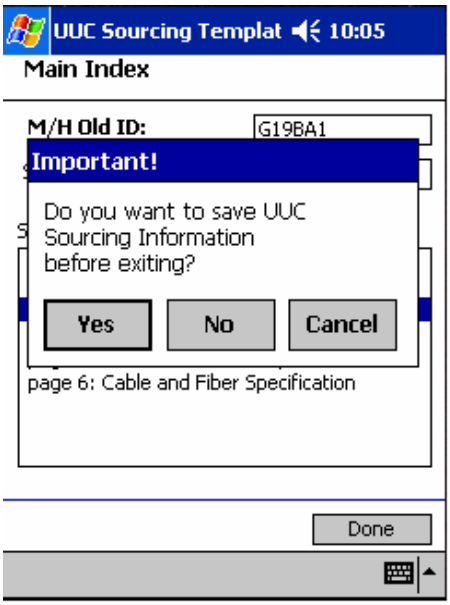



**Figure 47: Exit prompt**

The solution, therefore, was to load the software completely before it is used. This strategy is being used successfully in performance intensive application such as multimedia games. This way, the user waits only once at the beginning and not get interrupted while he is carrying out the data sourcing tasks.




**Figure 48: Field data collection application loading**

In the field data collection prototype, the GUI components, images and specification lists are loaded before the system is ready for use. The user, however, is informed about the time it takes to complete the loading process at constant intervals (see Figure 32). The loading time for the field data collection prototype is approximately 30 seconds, which is reasonable given the significant number of visual components used and hundreds of specification numbers read from system files.

The **finding UUC** interaction context is not supported by the user interface for the field data collection. But instead of leaving out entirely, an off-the-shelf spatial mapping software for Pocket PC from Intergraph called IntelliWhere OnDemand™ is used to display to the data-sourcing map (see Figure 33). The rest of the prototype runs as a custom application that is launched from the IntelliWhere OnDemand™.

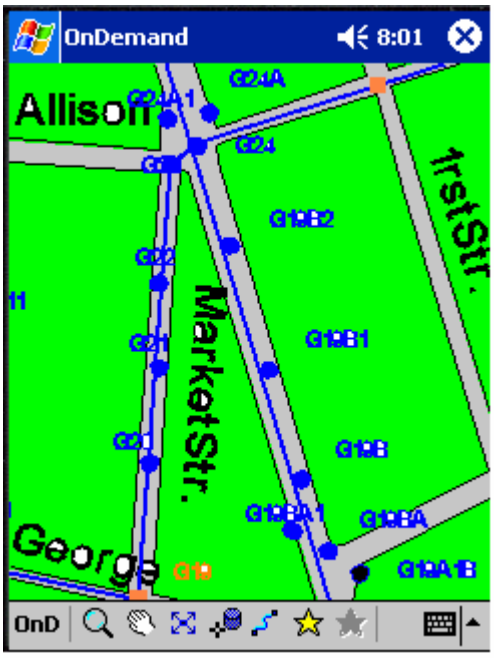



**Figure 49: Data sourcing map on a PDA**

A paper data-sourcing map (see Figure 3, Doc. No. 2) was digitized using GeoMedia™ which is desktop geospatial mapping software that works as a desktop server to IntelliWhere OnDemand™. The features used in the map include:

1. UUCs
2. SDCs
3. Ducts
4. Streets
5. Suburb/Area

These features are queried using their attributes (see Figure 34). For example, the user can ask for UUC with a particular ID, or alternatively, the IDs of the UUCs located on a particular street, suburb etc.

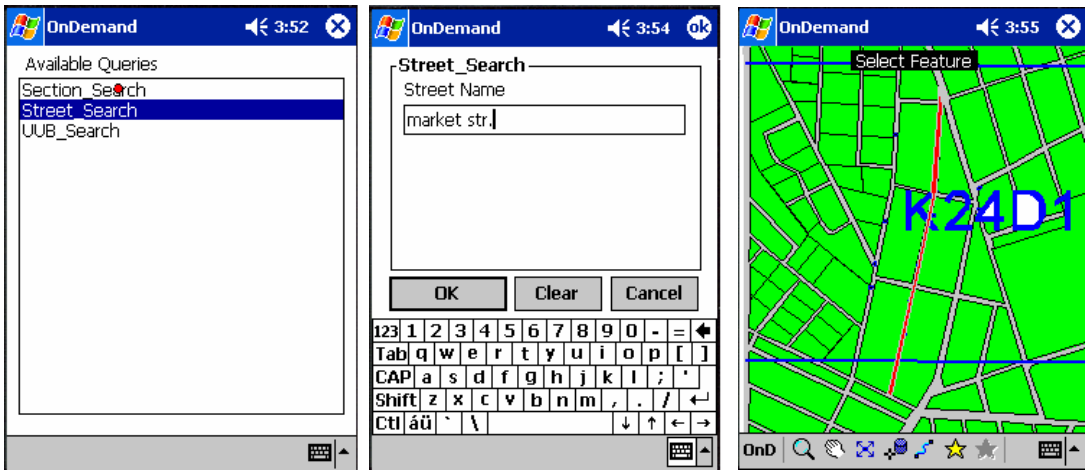



**Figure 50: Queries for street, asset and other attributes on the map**

The screen size is very limited on a PDA and not all the details of the map could be displayed simultaneously. Otherwise, the features looks crammed making it difficult to read the map. As shown in Figure 35, the solution is to make features appear only when the map is zoomed to a particular range of scale (e.g. 1:1000 to 1:5000). Naturally, the suburbs and streets would appear at much higher scale than say the UUCs and SDCs. The exact scale range is carefully determined based on the size of the map and proximity of the features.

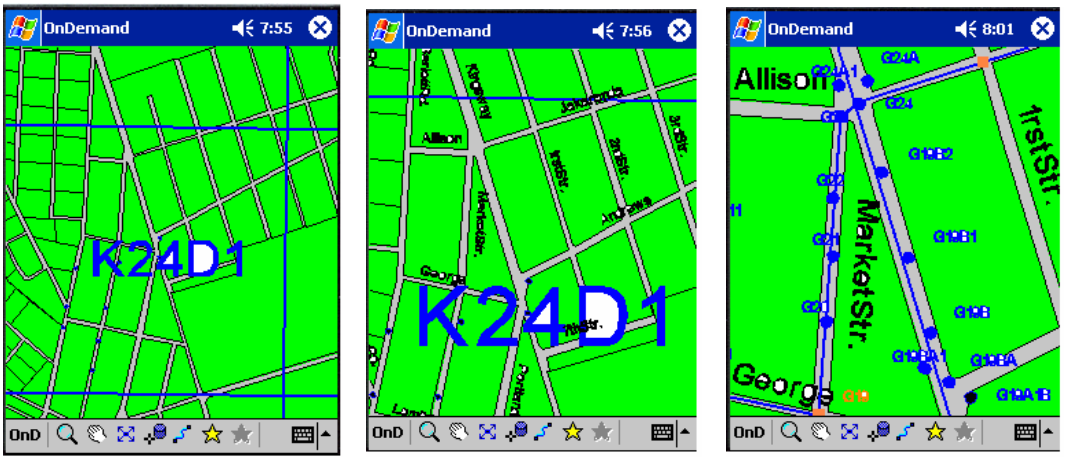



**Figure 51: Map attributes visible only at a particular range of scale**

The IntelliWhere OnDemand™ has a number of built-in map viewing functionalities such as pan, zoom in and zoom out. It allows these functions using a rich set of gestures offered by the direct interaction style of the stylus on a touch screen. One of the most important functions is the “red-lining” of assets on the map (see Figure 36). This function allows the user to make repetitive changes to the original asset information without overwriting the original data.

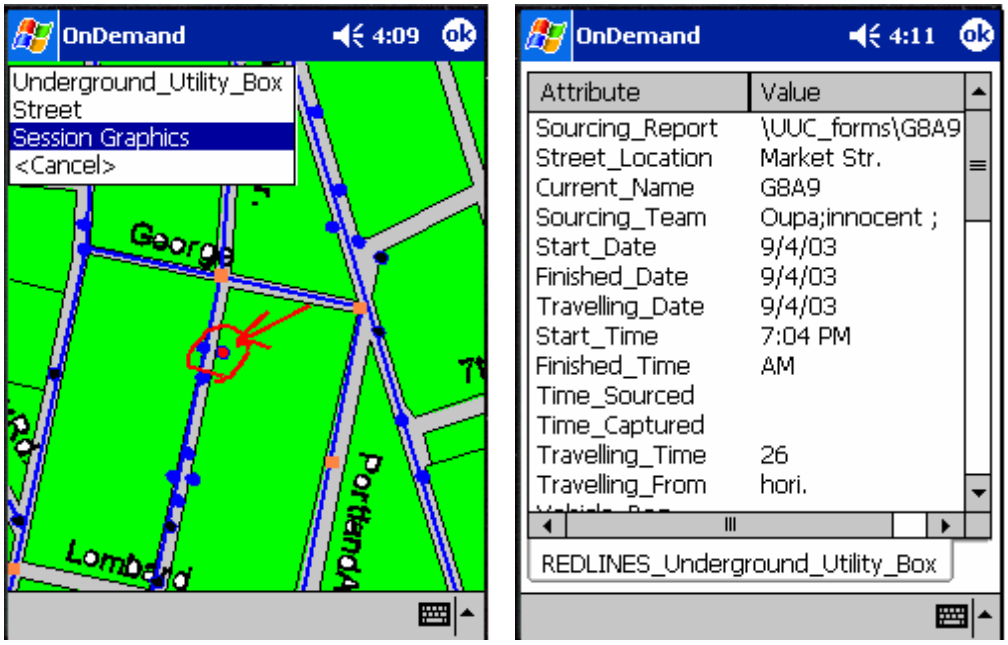
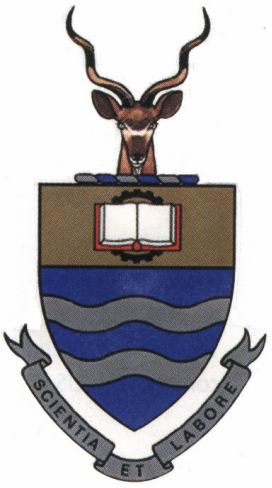



**Figure 52: Redline functionality of IntelliWhere OnDemand™**

# 3 Conclusion

The user interface is designed using a two-steps approach. The content of user interface is addressed before determining how they will look or behave. The latter is derived from the requirements analysis of users, tasks and operational context. The result is that the user interface is able to give the users what they need rather than what is available.

The user interface design is implemented as a working prototype on a Pocket PC operated PDA which has shown to be the most suitable hardware platform for the prototype to run on. This operating system is criticized for merely duplicating desktop software on a mobile device. However, the reality is that no other operating system is able to give the kind of underlying functionality in terms of multimedia and wireless connectivity required by a modern field data collection application. Again, this is a pragmatic approach and may not be able to provide a truly innovative solution. Another concern is that the programming language used, namely the eMbedded Visual Basic (eVB), restricts the flexibility of the user interface design due to its limited availability of UI components and widgets. To overcome this problem, standard GUI components are used creatively so that they can still appear intuitive to novice users. By and large, the user interface design is able to meet the usability and functional requirements successfully despite some of the limitations mentioned above.




# Software Design

Doc. No. 4

# CONTENTS

CONTENTS ..... I

LIST OF FIGURES ..... III

1 SCOPE ..... 1

1.1 Introduction ..... 1

1.2 Purpose ..... 1

1.3 Audience ..... 1

2 SOFTWARE DESIGN ..... 2

2.1 Unified Modeling Language (UML) ..... 2

2.2 ICONIX Process ..... 3
2.2.1 Use Case Model ..... 4
2.2.2 Robustness Diagram ..... 5
2.2.3 Sequence Diagram ..... 6
2.2.4 Domain Model ..... 7
2.2.5 Class Diagram ..... 7

3 USE CASE MODEL ..... 9

3.1 Use Case Map ..... 9

3.2 Use Case Documentation ..... 10
3.2.1 Select Sourcing Template Use Case ..... 10
3.2.2 Specify Sourcing Team Use Case ..... 11
3.2.3 Update Team Use Case ..... 12
3.2.4 Specify Times & Dates Use Case ..... 13
3.2.5 Specify Site Information Use Case ..... 14
3.2.6 Specify Manhole Covers Type Use Case ..... 15
3.2.7 Specify UUC Content Use Case ..... 16
3.2.8 Specify Manhole Dimensions Use Case ..... 17
3.2.9 Place Duct Use Case ..... 17
3.2.10 Specify Duct Use Case ..... 18
3.2.11 Place Cable Use Case ..... 19
3.2.12 Specify Cable Use Case ..... 20
3.2.13 Place Joint Use Case ..... 21
3.2.14 Specify Joint Use Case ..... 22

4 ROBUSTNESS DIAGRAM ..... 23

**5 SEQUENCE DIAGRAM ..... 31**

**6 CLASS DIAGRAM ..... 45**

**7 CONCLUSION ..... 48**

# LIST OF FIGURES

*Figure 1: The Vocabulary of UML (Source: Jacobson I et al., 1999a)* 2
*Figure 2: ICONIX process (Source: Rosenberg D et al., 2001a)* 4
*Figure 3: Robustness analysis bridges the gap between what and how of software design process (Source: Rosenberg D et al., 2001c)* 5
*Figure 4: Robustness diagram stereotypes.* 6
*Figure 5: Use Case Diagram for the field data collection application.* 9
*Figure 6: Robustness Diagram for Select Sourcing Template Use Case* 23
*Figure 7: Robustness Diagram for Specifying Sourcing Team Use Case* 24
*Figure 8: Robustness Diagram for Update Team Use Case* 24
*Figure 9: Robustness Diagram for Specify Times & Dates Use Case* 25
*Figure 10: Robustness Diagram for Specify Site Use Case* 25
*Figure 11: Robustness Diagram for Specify Manhole Covers Use Case* 26
*Figure 12: Robustness Diagram for Specify UUC Content Use Case* 26
*Figure 13: Robustness Diagram for Specify UUC Dimension Use Case* 27
*Figure 14: Robustness Diagram for Place Duct Use Case* 27
*Figure 15: Robustness Diagram for Specify Duct Use Case* 28
*Figure 16: Robustness Diagram for Specify Cable Use Case* 28
*Figure 17: Robustness Diagram for Place Cable Use Case* 29
*Figure 18: Robustness Diagram for Place Joint Use Case* 30
*Figure 19: Robustness Diagram for Specify Joint Use Case* 30
*Figure 20: Sequence Diagram for Select Sourcing Team Use Case* 31
*Figure 21: Sequence Diagram for Specify Sourcing Team Use Case* 32
*Figure 22: Sequence Diagram for Update Sourcing Team Use Case* 33
*Figure 23: Sequence Diagram for Specify Times & Dates Use Case* 34
*Figure 24: Sequence Diagram for Specify Site Information Use Case* 35
*Figure 25: Sequence Diagram for Specify Manhole Cover Types Use Case* 36
*Figure 26: Sequence Diagram for Specify UUC Content Use Case* 37
*Figure 27: Sequence Diagram for Specify Manhole Dimensions Use Case* 38
*Figure 28: Sequence Diagram for Place Duct Use Case* 39
*Figure 29: Sequence Diagram for Specify Duct Use Case* 40
*Figure 30: Sequence Diagram for Place Cable Use Case* 41
*Figure 31: Sequence Diagram for Specify Cable Use Case* 42
*Figure 32: Sequence Diagram for Place Joint Use Case* 43
*Figure 33: Sequence Diagram for Specify Joint Use Case* 44
*Figure 34: Class Diagram (part 1) for Field Data Collection Application* 45
*Figure 35: Class Diagram (part 2) for Field Data Collection Application* 46
*Figure 36: Class Diagram (part 3) for Field Data Collection Application.* 47

# 1 Scope

## 1.9 Introduction

This document details the software analysis and design process of the field data collection application. The detailed functionality, particularly those relating to the usability of the field data collection application is illustrated using object-oriented software design models. The user interface prototype developed during the User Interface Design (Doc. No. 3) is used as a “starting point” for the software design process. It is important to stress that the software usability does not only depends on the appearance of the user interface but how it functions and behaves in response to user actions as well.

This document is concerned primarily with high-level design architecture of the application layer (as opposed to user interface layer) of the software. Hence, the document does not include the actual implementation of the application layer (i.e. program source code).

## 1.10 Purpose

The purpose of this document is to present the software functionality and the design architecture of the field data collection application. In doing so, the reader should gain a better understanding of the behaviour and complexity of the field application.

## 1.11 Audience

The intended readers of this document include researchers involved in the areas of software usability, mobile computing, as well as software designers, developers, the external examiner and other interested parties.

# 2 Software Design

## 2.1 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standard modeling language for object-oriented software – "*a language for visualizing, specifying, constructing, and documenting artifacts of a software-intensive system*" (Jacobson I *et al.*, 1999a). Essentially, UML enables software developers to visualize their software designs in standardized blueprints or diagrams. UML represents a unified notation for expressing a variety of models and can be used by a variety of object-oriented methods. The names of the various UML diagrams and their graphical details may differ considerably but the deployment of most important diagrams (e.g. use cases, sequence and class diagrams etc.) are widely accepted (see Figure 1).

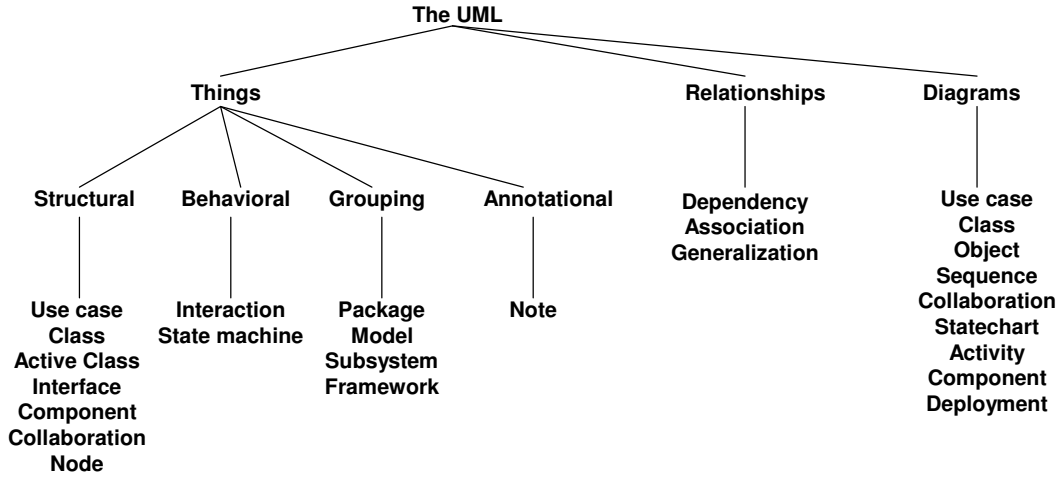



**Figure 53: The Vocabulary of UML (Source: Jacobson I *et al.*, 1999a)**

## 2.2 ICONIX Process

The ICONIX process uses a subset of UML based on Dough's analysis of the three individual methodologies developed by Ivar Jacobson, Jim Rumbaugh and Grady Booch Rosenberg D *et al*., 2001a). ICONIX process is a relatively lightweight object-oriented design process without a lot of overhead of a complete UML design process. Simply put, it "*sits somewhere in between the very large Rational Unified Process (RUP) and the very small eXtreme programming (XP) approach*" (Rosenberg D *et al*., 2001a). This makes the ICONIX process especially suitable for relatively small software projects. The key features of the ICONIX process is that (Rosenberg D *et al*., 2001a):

- It offers *streamlined usage* of the UML. In other words, it is a "minimalist" approach comprising minimal subsets of a large and often unwieldy UML.

- It offers a high degree of *traceability*. At every step along the way, the designer refers back to the requirements in some way.

- It is *iterative* and *incremental*. Multiple iterations occur between developing the domain model and identifying and analyzing use cases.

As shown in Figure 2, the ICONIX process begins with the graphical user interface (GUI) prototype whether it is a simple drawing or a working prototype. The design models used in the ICONIX process extend from the GUI prototype. For instance, the use case narratives and functions match up with the GUI components (e.g. buttons, textbox etc.). The ICONIX process describes the behavior and structure of the system using dynamic and static models respectively (see Figure 2). The constituents of these models are described in the sections below.

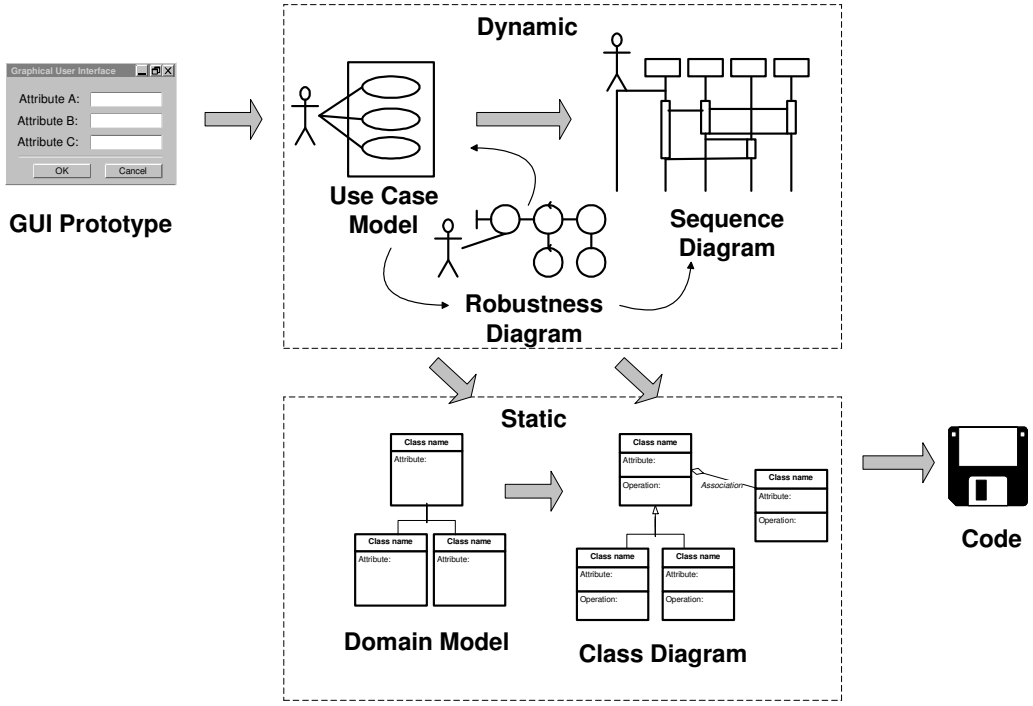



**Figure 54: ICONIX process (Source: Rosenberg D *et al.*, 2001a)**

## 2.2.1 Use Case Model

The **use case model** describes the runtime behavior of the system by addressing what the users are trying to do with the system (Rosenberg D *et al.*, 2001b). This involves the capture of user actions and the associated system responses in great detail. It is necessary to ask *"what happens?"* when the user interacts with the system (a basic course of action) and *"what else can happen?"* (other alternative courses of action). The user interface prototype helps define the use cases from the beginning. The entire dynamic part of the object model is then derived from the use case model.

### 2.2.2 Robustness Diagram

The **robustness diagram** identifies objects that are needed to complete use cases in the use case model. It is similar to a UML collaboration diagram, in that it shows the objects that participate in the scenario and how these objects interact with each other (Rosenberg D *et al*., 2001c). However, the robustness diagram makes the transition from the requirements to detailed design easier. One of the most difficult problems in the software development is moving forward from the "what" view into a "how" view of the software design (see Figure 3). The analysis of this diagram makes this transition easier by closing the gap between the requirement analysis and detailed design (Rosenberg D *et al*., 2001c).

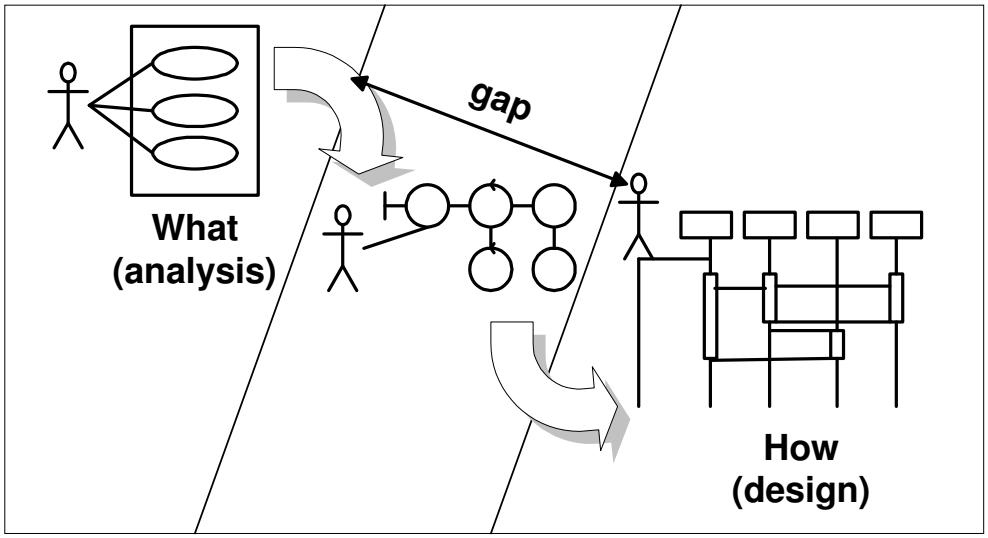



**Figure 55: Robustness analysis bridges the gap between what and how of software design process (Source: Rosenberg D *et al*., 2001c)**

The robustness diagram constitutes three stereotypes:

- **Boundary objects** – actors use them to communicate with the system. Many of the boundary objects are found in the GUI prototype.

- **Entity objects** – are usually objects from the domain model. Typically, they are database tables, and files that hold the information that needs to "outlive" use case execution.

- **Control objects** – also called **controllers** serve as the "glue" between boundary objects and entity objects. They embody much of the application logic. They are placeholders for any functionality and system behavior required by the use cases.

The Figure 4 below shows the visual icons for these stereotypes.

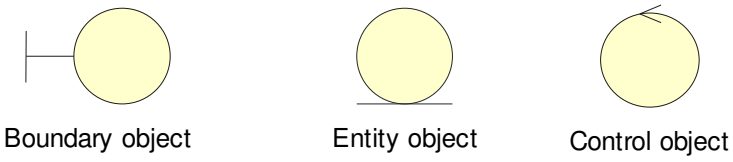



**Figure 56: Robustness diagram stereotypes.**

### 2.2.3 Sequence Diagram

The **sequence diagram** represents a detailed design of the system which is largely about allocating behavior (Rosenberg D *et al*., 2001d). In other words, it describes the system behavior in great detail in a sequential manner. In doing so, it encompasses the *basic course* and all *alternate courses of action* within each of the use case. The four constituents of a sequence diagram are:

- *The text for the course of actions of the use case* appears down the left-hand side.

- *Objects*, which are carried over from the robustness diagrams, and they are displayed with their robustness diagram stereotypes.

- *Messages* are arrows between objects. The objects can send message to each other or itself.

- *Methods* are shown as rectangle on top of message arrows. The length of these rectangles reflects the length of *focus of control* within the sequence. The use of *focus of control* is optional.

### 2.2.4 Domain Model

The **domain model** identifies the main conceptual objects that are going to participate in the system designed (Rosenberg D *et al*., 2001d). This is necessary in the object-oriented software design which has a structure that is based on real world objects. The domain model also identifies the relationship between these objects such as **generalization** ("kind of") and **aggregation** ("part of"). In general, the best sources of domain classes are derived from a high-level problem statement, lower-level requirements and expert knowledge of the problem space (Rosenberg D *et al*., 2001d). More specifically, nouns and noun phrases become objects and attributes while verbs and verb phrases become operations and associations.

### 2.2.5 Class Diagram

A class diagram is essentially a static design view of the system by describing how the code is organized (Rosenberg D *et al*., 2001d). The class diagram also shows the type of relationships between the classes and objects such as interfaces and even other systems. These relationships include:

- **Association** – Classes or objects may be connected, associated or related because they share some mutual properties or interaction between them.

The association is represented using a solid line drawn between the participating classes. Associations are assumed bidirectional. But if the line has an arrow at the end, then it indicates the direction of the relationship.

- **Aggregation** – This relationship is used when a class is composed of components which are other classes or objects. This is the “has a” type of relationship. The aggregation is denoted with a solid line with a hollow diamond on the end. The object on the diamond end “has” or “contains” the object on the other end.

- **Inheritance** – This relationship exits between one class object that is a specialized version of another class object. These classes have a “is a” or “is like” relationship where they share similar attributes and methods. Usually, the subclass “inherits” some of its properties from the parent class. The inheritance relationship in UML is depicted as a solid line with a triangular arrowhead pointing towards the parent class.

# 3 Use Case Model

## 3.1 Use Case Map

The use case map (see section 2.2.1) of the mobile application for UUC field data collection is as follows:

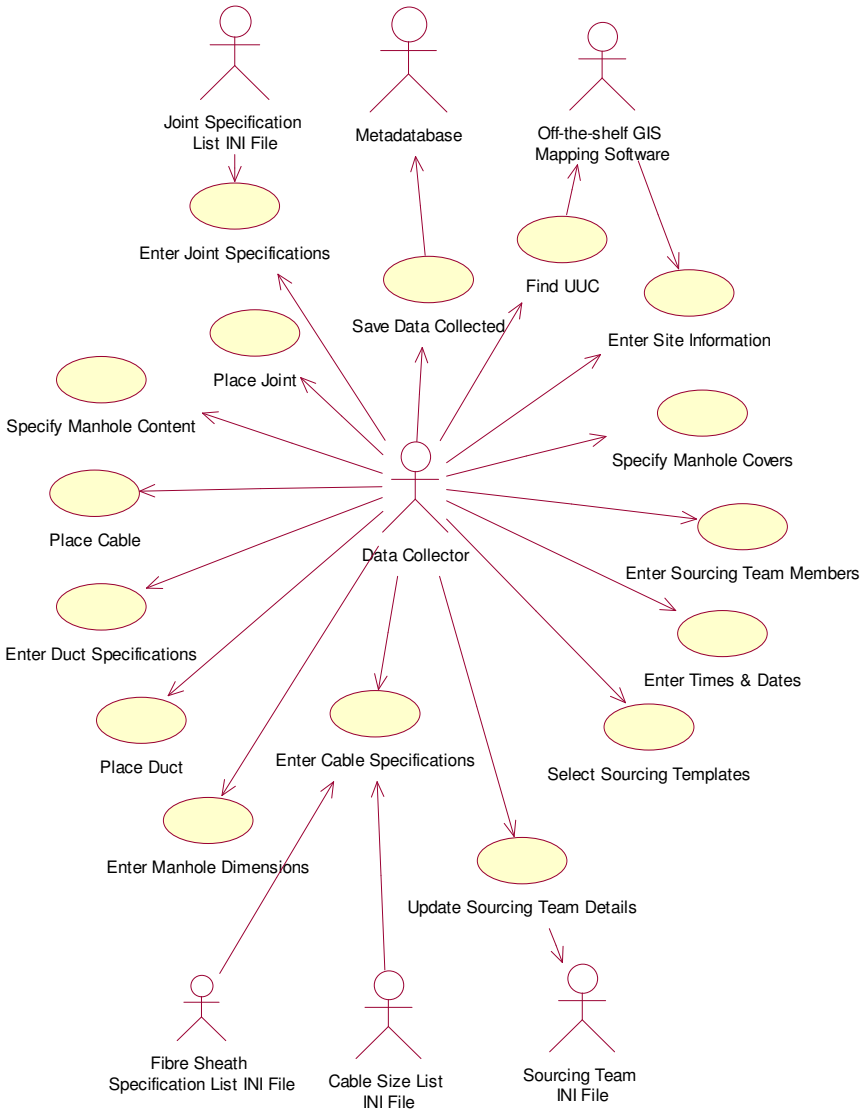



**Figure 57: Use Case Diagram for the field data collection application.**

## 3.2 Use Case Documentation

The related documentation for use cases in section 3.1 is presented below. The use case documentation follows the convention by Rosenberg (2001a).

### 3.2.1 Select Sourcing Template Use Case

**Basic Course:**

The Data Collector clicks the first item in the list box from the Main Menu Page. The system displays the Sourcing Team Page.

**Alternate Course:**

If the Data Collector clicks the second item in the listbox, the system displays the Times & Dates Page.

If the Data Collector clicks the third item in the listbox, the system displays the Site Information Page.

If the Data Collector clicks the fourth item in the listbox, the system displays the Manhole Information Page.

If the Data Collector clicks the fifth item in the listbox, the system displays the Ducts Location Page.

If the Data Collector clicks the sixth item in the listbox, the system displays the Cable Specification Page.

If the Data Collector clicks the "Done" button, the system prompts the user if he wants to save the data before exiting the application. If "Yes" button is clicked, the

system saves the data before ending the application. If “No” button is clicked, the application ends without saving the data collected. If "Cancel" button is clicked, the system returns control back to the Main Menu Page.

### 3.2.2 Specify Sourcing Team Use Case

**Basic Course:**

The Data Collector clicks the Add Member button on the Sourcing Team Page. The system displays the Sourcing Team Section Page. The system then retrieves the names of all the members from the persistent team list.

The Data Collector selects one or more members from the list. The system enables the “Add” and “Remove” buttons. The Data Collector clicks the “Add” button. The system adds the selected members to the listbox in the Sourcing Team Page. The system then returns the user back to the Sourcing Team Page.

**Alternate Course:**

If the Data Collector clicks the “Select All Members” button, the system select all the members in the listbox.

If the Data Collector clicks the “Update Member Info” button, the system passes control over to the Update Sourcing Team Details use case.

If the Data Collector clicks the “Cancel” button, the system returns to the Sourcing Team Page without adding any members to the list.

If the Data Collector clicks the “Remove Member” button, the system removes selected members from the list.

If the Data Collector clicks the “<Index” link, the system passes control over to the Selecting Sourcing Template use case.

If the Data Collector clicks the “page2>” link, the system passes control over to the Enter Times & Dates use case.

### 3.2.3 Update Team Use Case

**Basic Course:**

The Data Collector clicks the “Update Member Info” button on the Sourcing Team Selection Page. The system displays the Members Info Page. The system then retrieves all the member information and displays them onto Members Information table in the Member Info Page.

The Data Collector clicks the row of the table to select a particular member. The system enables the “Edit” and “Delete” buttons. The Data Collector clicks the “Edit” button. The system displays the Edit Member Info Page with the selected member details retrieved from the table. The system then opens the SIP.

The Data Collector types in the data using SIP and clicks the “OK” button. The system closes the SIP. The system updates the changes to the Members Information table and to the persistent member information data, and then returns the Data Collector back to Member Info Page.

**Alternate Course:**

If the Data Collector clicks the “New” button, the system displays the Edit Member Info Page with empty data fields. The user then enters the information for new data sourcing personnel. Thereafter, he clicks the “OK” button, and the system adds new member information to the persistent data. It then displays the Members Info Page and creates a new row with the information entry.

If the Data Collector clicks the “Delete” button, the system removes the table row of the chosen member, and the system deletes the member from the persistent member information.

### 3.2.4 Specify Times & Dates Use Case

**Basic Course:**

The Data Collector checks the “Start Date” checkbox. The system inserts current date into the “Start Date” textbox accompanied by a sharp 'click' sound. The Data Collector checks the “Finished Date” checkbox. The system inserts the current date into the “Finished Date” textbox accompanied by a sharp 'click' sound. The Data Collector checks the “Traveling Date” checkbox. The system inserts the current date into the “Traveling Date” textbox accompanied by a sharp 'click' sound.

The Data Collector checks the “Start Time” checkbox. The system inserts the current date into the “Start Time” textbox accompanied by a sharp 'click' sound. The Data Collector selects the sourcing time from a combobox. The system responds with a sharp 'click' sound. The Data Collector again selects the traveling time from a combobox. The system responds with a sharp 'click' sound.

The Data Collector clicks the “Traveling From” textbox. The system opens the SIP and resizes the scrollbar. The Data Collector enters the “Traveling From”, “Vehicle Registration” and “Kilometers” information. The system verifies if the “Kilometers” information is entered correctly (i.e. the data must be integer or float).

**Alternate Course:**

If the Data Collector clicks the “Date and Time” textbox (he decides to insert dates by himself), the system opens the SIP and verifies if the data entered is valid. If the

data entered is not in a correct format (i.e. mm/dd/yy), the system displays the type of error and prompts the user for re-entry.

If the Data Collector clicks the “Index” link, the systems passes control over to the Select Sourcing Templates use case.

If the Data Collector clicks the “<page1” link, the system passes control over to the Enter Sourcing Team Members use case.

If the Data Collector clicks the “page3>” link, the system passes control over to the Enter Site Information use case.

### 3.2.5 Specify Site Information Use Case

**Basic Course:**

The Data Collector clicks the “Site Name” textbox, and the eventlistener opens the SIP. He edits the Site Name (if necessary). The system checks the validity of the data entered.

**Alternate Course:**

If the Data Collector clicks the “Index” link, the system passes control over to the Select Sourcing Templates use case.

If the Data Collector clicks the “Page4” link, the system displays the Manhole Information Page.

If the Data Collector clicks the “Page3” link, the system passes control over to the Enter Times & Dates use case.

### 3.2.6 Specify Manhole Covers Type Use Case

**Basic Course:**

The Data Collector checks the “Manhole Cover Type” checkbox. The system enables the “No of Covers” combobox. He enters the number of covers by selecting from a “No of Covers” combobox. Once complete, the system makes a sharp 'click' confirmation sound.

The Data Collector indicates that the manhole is secured by checking the “Yes” checkbox. The system enables the “No of Covers” comboboxes. He then selects the no. of covers from the combobox which the system confirms with a sharp beep.

**Alternate Course:**

If the Data Collector clicks the comboboxes without checking the “Manhole Type” checkboxes the system sounds out a distinct 'disapproval' beep.

If the Data Collector checks the “No” checkbox, the system unchecks the “Yes” checkbox and disables the “No of Covers” comboboxes for security devices.

If the Data Collector clicks the “page3” link, the system invokes the Enter Site Information use case.

If the Data Collector clicks the “Index” link, the system invokes the Select Sourcing Templates use case.

If the Data Collector clicks the “page4” link, the system invokes the Place Duct use case.

### 3.2.7 Specify UUC Content Use Case

**Basic Course:**

The Data Collector checks the “Yes” checkbox on the UUC content panel. The system enables the “Time Pumped” combobox. The system makes a sharp ‘click’ sound as a feedback mechanism. The Data Collector selects the “Time Pumped” from the combobox. The system again makes a sharp ‘click’ sound to confirm the data entry.

The Data Collector clicks the “Oxygen Level” textbox, and the system opens the SIP and resizes the scrollbar. He types in the “Oxygen level” and “Other Gasses” information. The system verifies each of these data entries.

**Alternate Course:**

If the Data Collector clicks the “Time Pumped” textbox, the system opens the SIP and verifies if the data entered is in the correct format. If not, the system displays an error message and prompts the user for re-entry. When this data container loses focus, the system closes the SIP and resizes the scrollbar.

If the Data Collector clicks the “page3” link, the system invokes the Enter Site Information use case.

If the Data Collector clicks the “Index” link, the system invokes the Select Sourcing Templates use case.

If the Data Collector clicks the “page4” link, the system invokes the Place Duct use case.

### 3.2.8 Specify Manhole Dimensions Use Case

**Basic Course:**

The Data Collector clicks the “Length” textbox. The eventlistener for that data container is activated and the system opens the SIP and resizes the scrollbar. The Data Collector types in the rest of the dimensions (i.e. depth, breath, height etc.) of the manhole which the system checks on entry to see if they are in the correct format.

The Data Collector clicks the “>>” button until he finds the right manhole shape. The system locates the next shape on the Image List and displays it on to the picturebox.

**Alternate Course:**

If the data entered is incorrect, the system rejects it and prompts the Data Collector for re-entry.

If the Data Collector clicks the “<<” button the system locates the previous shape on the Image List and displays it on to the picturebox.

### 3.2.9 Place Duct Use Case

**Basic Course:**

The Data Collector checks the “Add” radiobutton. Thereafter, he clicks a particular routeface on the topographic view of UUC. The system places the selected routeface at the top of the topographic view and rotates the other routeface accordingly. It then retrieves information about the contents (i.e. ducts) of selected routeface and displays them onto the “Routeface” panel.

The Data Collector clicks the “x” label (i.e. duct location) to place a new duct on the routeface panel. The system places a duct at the selected position, and confirms the placement with a sharp beep. The system adds new duct information to the persistent duct list.

**Alternate Course:**

If the Data Collector clicks a location that is already occupied, the system displays an error message and prompts the user for re-selection.

If the Data Collector checks the “Remove” radiobutton, and then picks a duct, the system removes the duct from the “Routeface” panel as well as from the duct list.

If the Data Collector clicks the “Page4” link, the system displays the Manhole Information Page.

If the Data Collector clicks the “Index” link, the system invokes the Select Sourcing Templates use case.

If the Data Collector clicks the “Page6” link, the system displays the Cable Specification Page.

### 3.2.10 Specify Duct Use Case

**Basic Course:**

The Data Collector checks the Specify/Edit radiobutton. He then selects the duct which hasn't yet specified (indicated by different color). The system displays the Specify Duct Page. He enters the specifications from a list of comboboxes, which the system confirms by an audio feedback. The Data Collector clicks the “OK” button. The system updates the specifications data. The system changes the color of the duct to indicate that it has being specified. The system returns to the

DuctLocationPage. (The Data Collector may repeat this process until all the ducts are specified).

**Alternate Course:**

If the Data Collector chooses to edit a duct (i.e. make changes to existing specifications) by clicking the duct that is specified (highlighted by a different color), the system retrieves the old specification and displays it onto the comboboxes in the SpecifyDuct page. Once changes are made, the system updates them with the persistent duct specifications.

If the Data Collector clicks the “See Specs” button, the system displays the Duct Specifications Page with specification of all ducts in a table.

### 3.2.11 Place Cable Use Case

**Basic Course:**

The Data Collector selects a routeface tab. The system retrieves and displays locations of ducts belonging to that routeface. The Data Collector selects the duct from which the cable originates. The system highlights the selected duct and expands the duct node from the tree. It then enables the “Add Cable” radiobutton.

The Data Collector checks the “Add Cable” radiobutton. The system displays the Insert Cable page and prepares the data entry by opening the SIP. The Data Collector enters the cable specifications and other notes and clicks the “OK” button. The system saves the new duct specifications. It adds a new cable node to the tree widget and returns control over to the Cable Specification page. The system confirms the new cable entry with a sharp beep.

**Alternate Course:**

If the cable originates from a joint, the Data Collector clicks that joint from the tree widget instead of the duct from the routeface panel.

If the Data Collector clicks a cable node and then clicks the “Delete” button, the system removes the node from the cable network (i.e. tree widget) and delete its specifications from the Cable Specifications table.

If the Data Collector clicks the “Index” Link, the system invokes the Select Sourcing Templates use case.

If the Data Collector clicks the “Page5” link, the system displays the Ducts Location Page.

If the Data Collector clicks the “Specs” button the system invokes the Specify Cable use case.

### 3.2.12 Specify Cable Use Case

**Basic Course:**

The Data Collector clicks the “Specs” button. The system displays the Cable Spec Table Page. It then retrieves specifications of all the cables onto the table and highlight the cable that is selected by Data Collector. The Data Collector clicks the “Edit” button and the system displays the Specify Cable page with specification that were already known. The Data Collector completes the specification and clicks the “OK” button.

The system adds the specifications to the cable data entity and confirms with a sharp beep. The system then returns control over to the Specification Table Page.

The Data Collector clicks the “Close” button and system returns control back to the Cable Specification Page.

**Alternate Course:**

If the Data Collector clicks the Cable Size or Gauge specifications textboxes, the system opens the SIP and verifies on the data on entry to see if it is valid. If not, the system displays error message and prompts the user for re-entry.

If the Data Collector clicks the “Cancel” button, the system returns to the Cable Spec Table Page without adding the specifications to the Cables entity.

If the cable specified is the fibre optics, the Data Collector clicks the “See List” link. The system then displays the Sheath Numbers in a table format. The Data Collector selects a number and clicks the “Insert” button. The system inserts the cable specification number back to the textbox on the Specify Cable Page.

### 3.2.13 Place Joint Use Case

**Basic Course:**

The Data Collector clicks the cable node on the tree widget. The system enables the “Add Joint” radiobutton. The Data Collector checks the “Add Joint” radiobutton. The system displays the Insert Joint Page. The system opens the SIP and the Data Collector types in the specification notes, and once finished he clicks the “OK” button.

The system closes the SIP and adds a new joint to the Joints entity. The system returns control back to the Cable Specification Page. A new joint node is added to the Cable Network and to the Joint Specification Table. The system provides an audio feedback to confirm a new joint entry.

**Alternate Course:**

If the cable selected by the Data Collector is already connected at both source and destination nodes the system disables the “Add Joint” radiobutton.

If the Data Collector clicks the “Delete” button the system removes the selected joint node from the tree widget and deletes that joint from the Joints entity.

### 3.2.14 Specify Joint Use Case

**Basic Course:**

The Data Collector selects the joint node from the tree widget on the Cable Specification Page and clicks the “Specs” button. The system displays the Joint Specs Table page. It then retrieves all the joints specification from the Joints persistent data and displays them on the Joint Specification Table and highlights the row for the joint that is currently specified. The Data Collector clicks the “Edit” button and the system displays the Specify Joint Page.

The Data Collector clicks the See List link to view the specification reference numbers list. The system retrieves the Joint Ref. numbers and displays them onto the Joint Ref No Table. The Data Collector selects a particular number and clicks the “Insert” button. The system inserts the selected reference number back onto the Specify Joint Page. The Data Collector enters the rest of the specifications and clicks the “OK’ button. The system adds the specification to the Joints persistent data, and confirms data entry with an audio feedback and returns control back to the Joints Table page.

The Data Collector clicks the “Close” button and system returns control back to the Cable Specification Page.

# 4 Robustness Diagram

The robustness diagrams (see section 2.2.2) for the use cases introduced in use case model (see section 3.1) and later explained in section 3.2 are presented below.

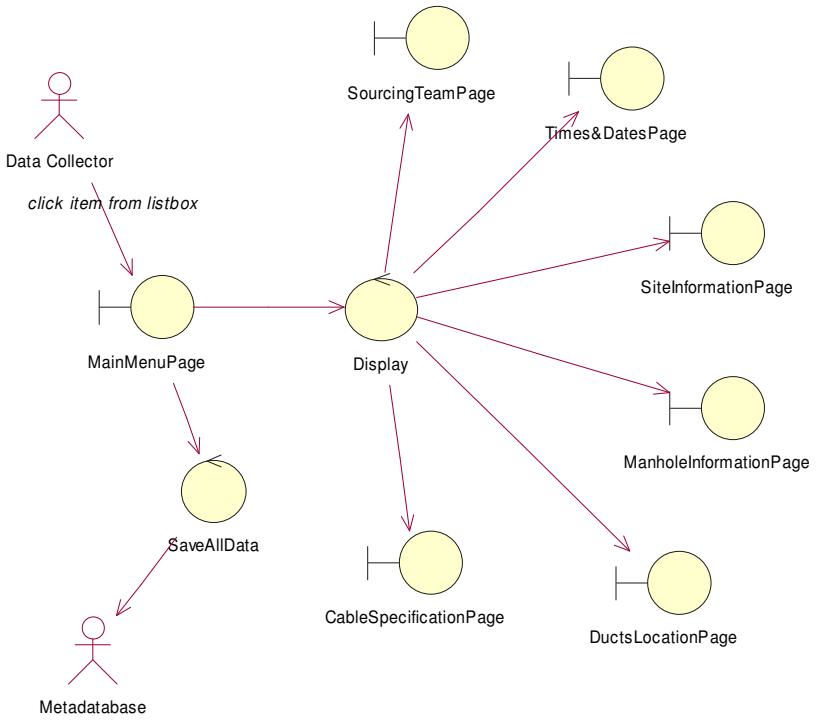



**Figure 58: Robustness Diagram for Select Sourcing Template Use Case**

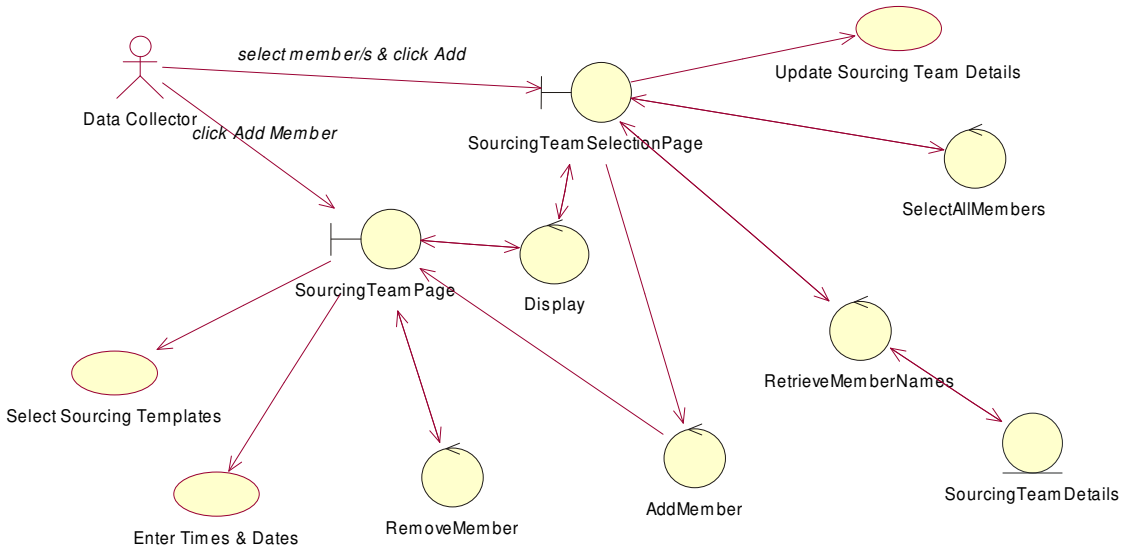



**Figure 59: Robustness Diagram for Specifying Sourcing Team Use Case**

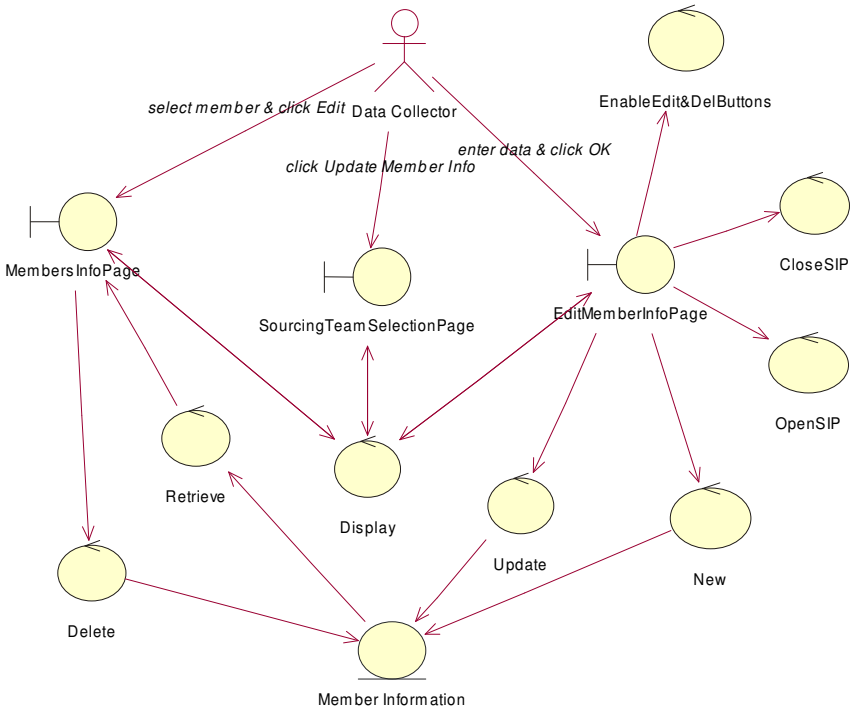



**Figure 60: Robustness Diagram for Update Team Use Case**

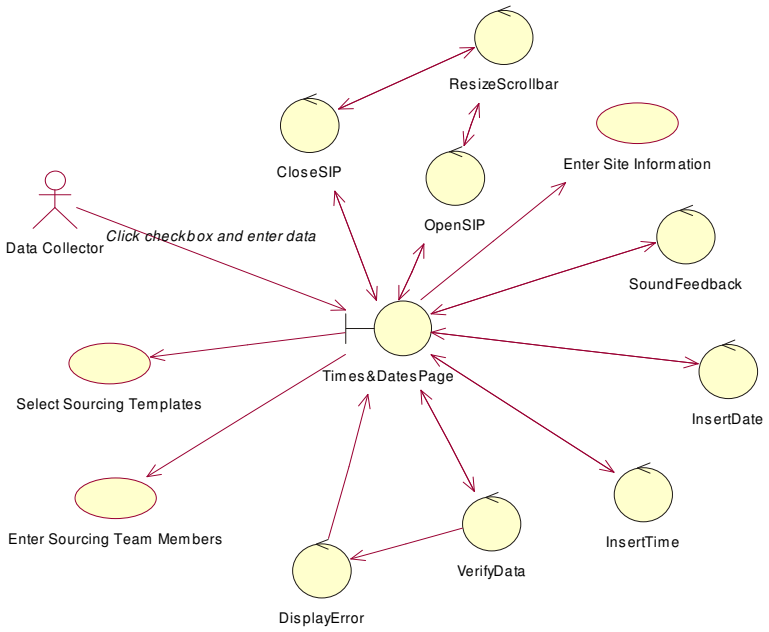



**Figure 61: Robustness Diagram for Specify Times & Dates Use Case**

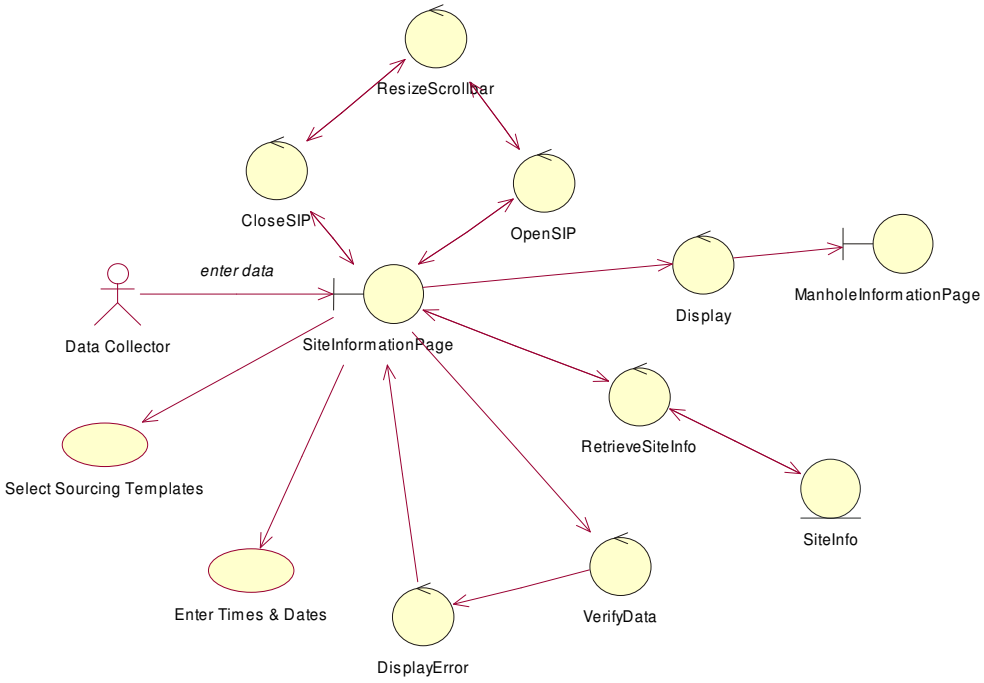



**Figure 62: Robustness Diagram for Specify Site Use Case**

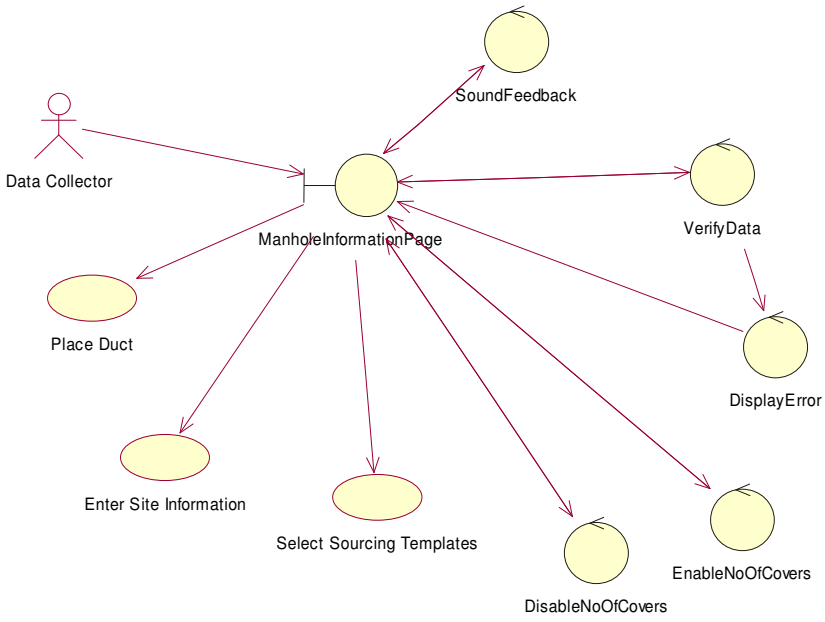



**Figure 63: Robustness Diagram for Specify Manhole Covers Use Case**

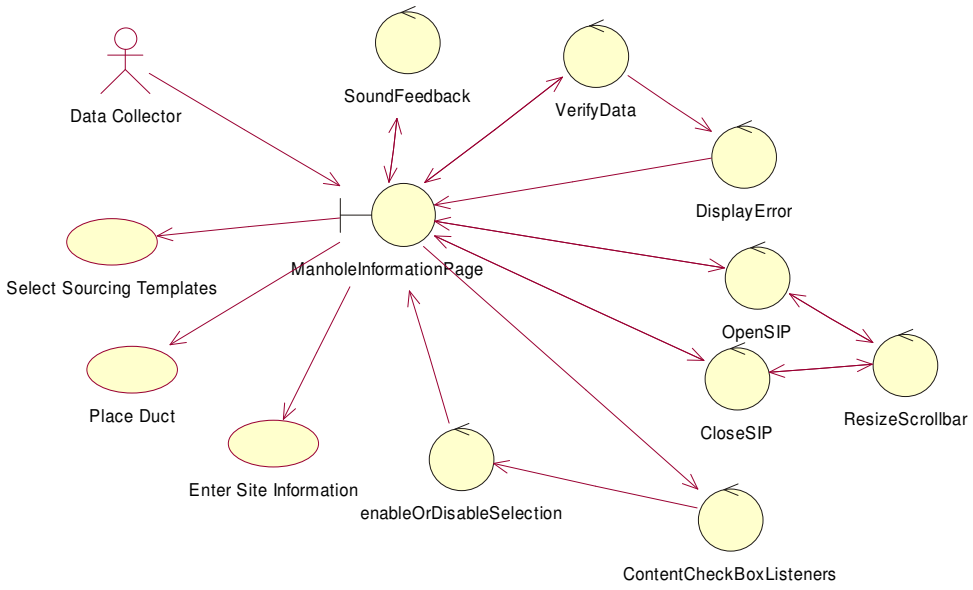



**Figure 64: Robustness Diagram for Specify UUC Content Use Case**

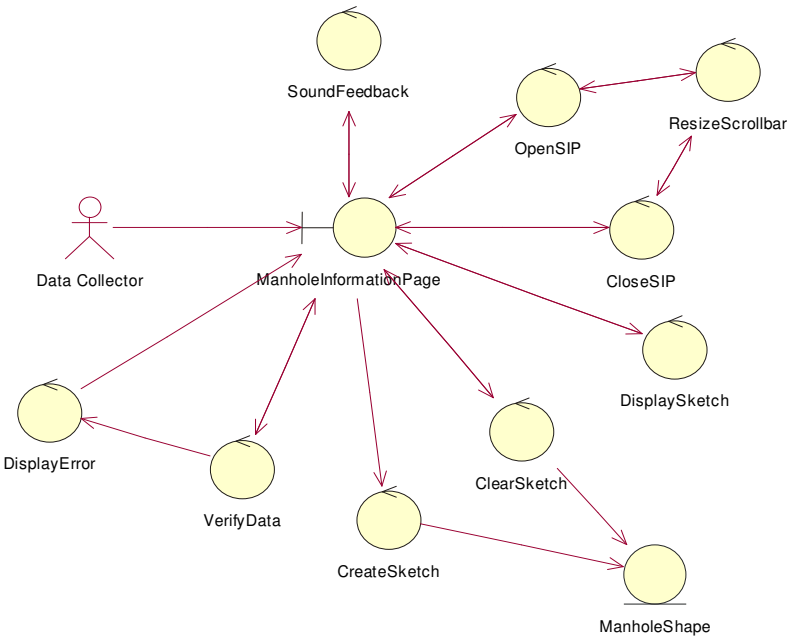



**Figure 65: Robustness Diagram for Specify UUC Dimension Use Case**

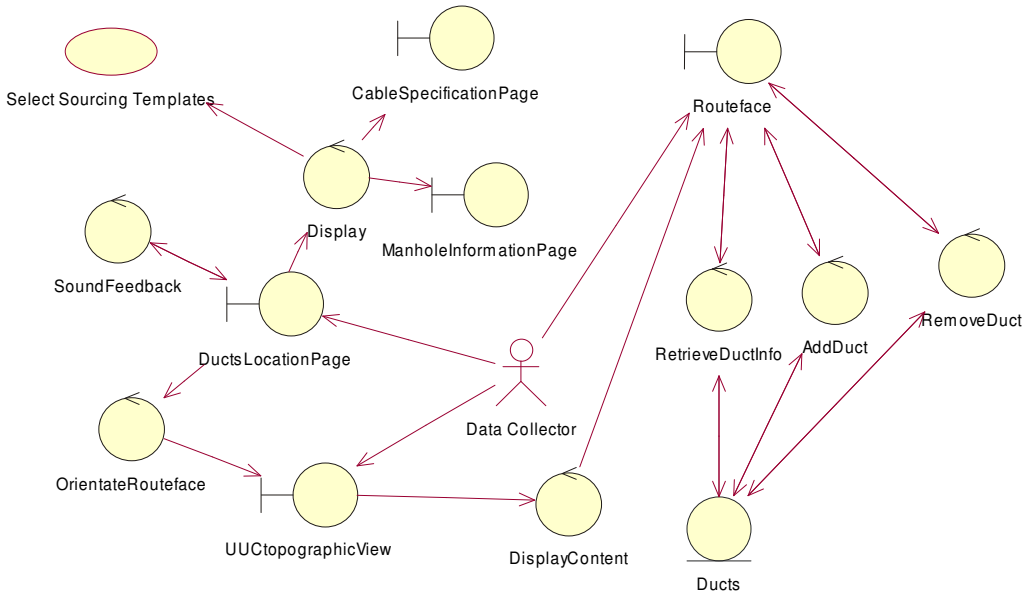



**Figure 66: Robustness Diagram for Place Duct Use Case**

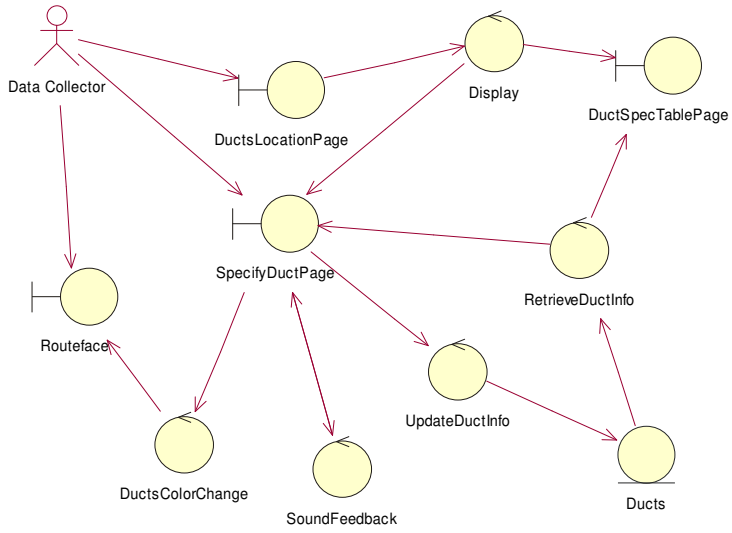



**Figure 67: Robustness Diagram for Specify Duct Use Case**

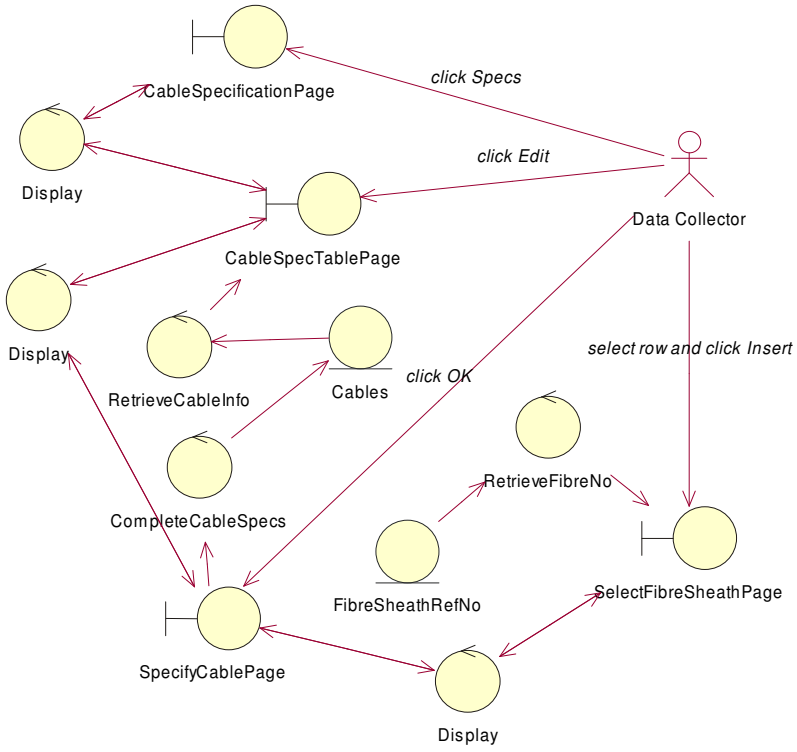



**Figure 68: Robustness Diagram for Specify Cable Use Case**

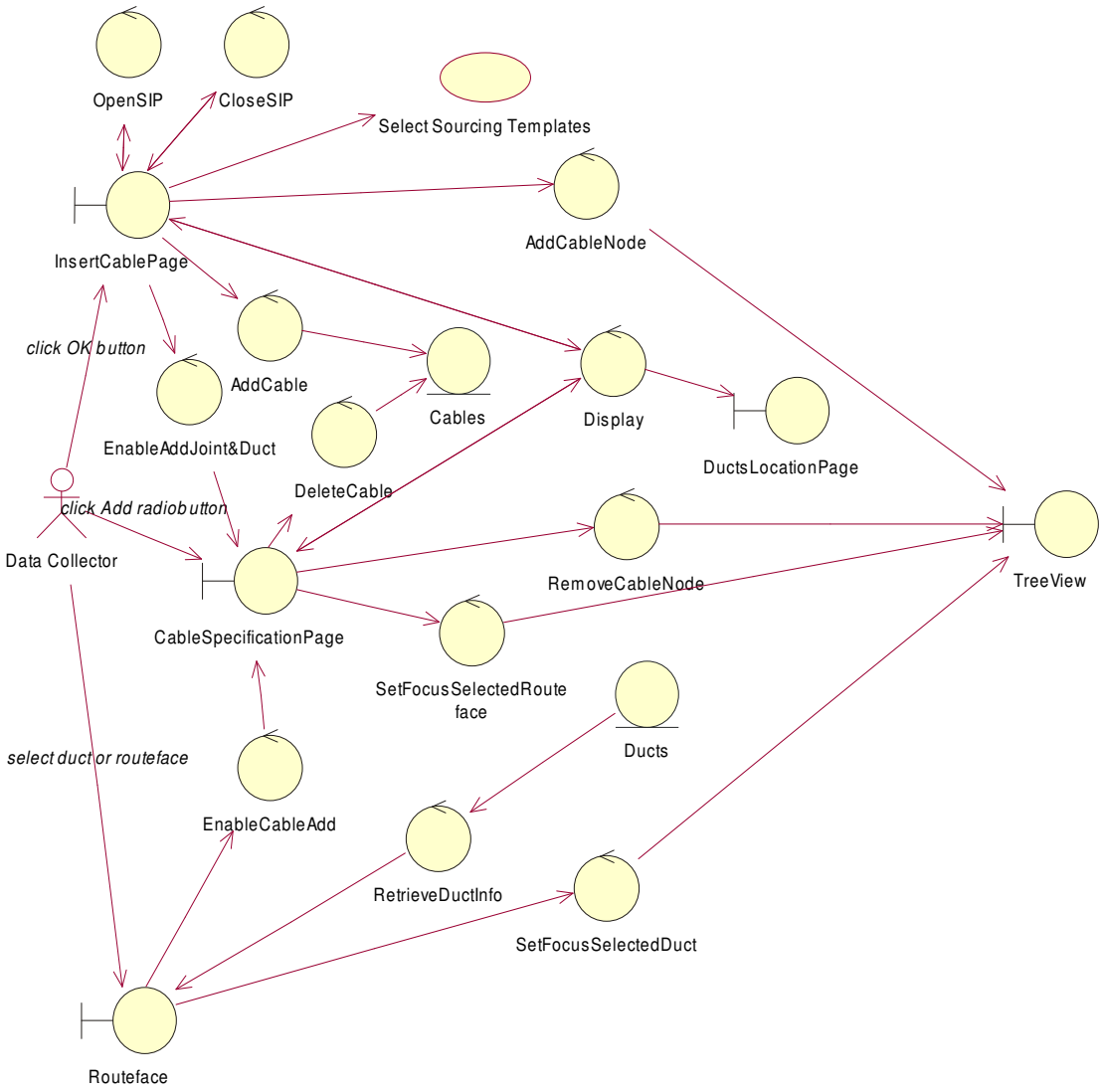



**Figure 69: Robustness Diagram for Place Cable Use Case**

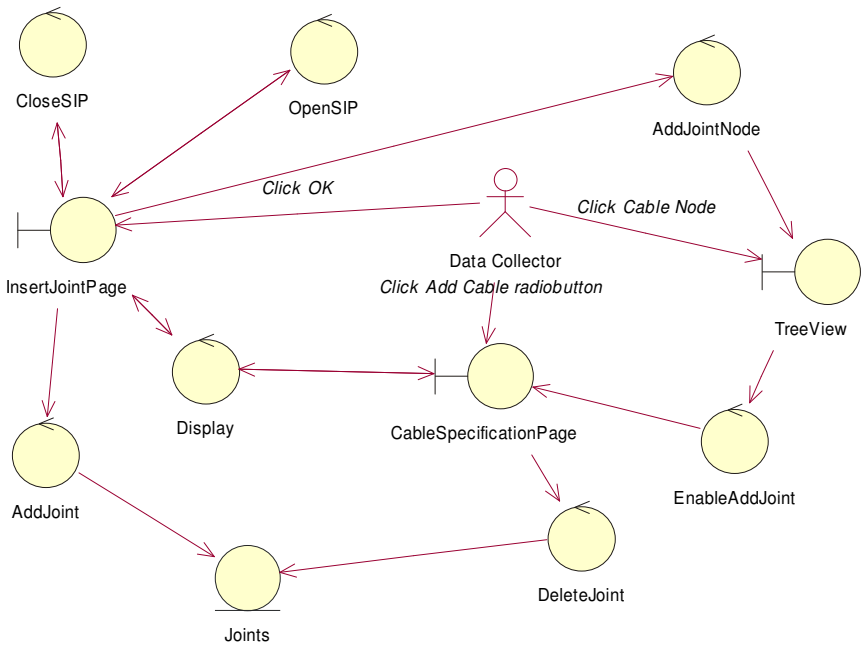



**Figure 70: Robustness Diagram for Place Joint Use Case**

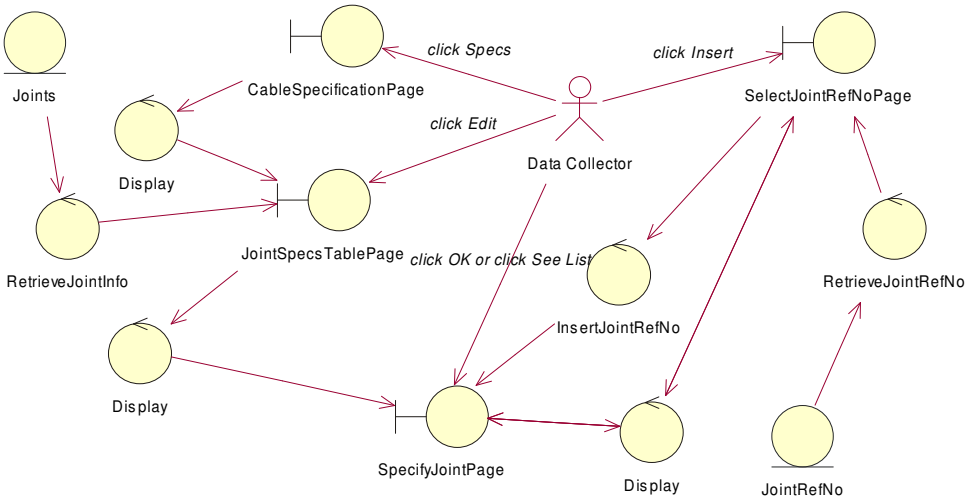



**Figure 71: Robustness Diagram for Specify Joint Use Case**

# 5 Sequence Diagram

The sequence diagrams (see section 2.2.3) for the use cases (see section 3.2) are presented below. The details described in sequence diagram follow from the robustness diagram in section 4.

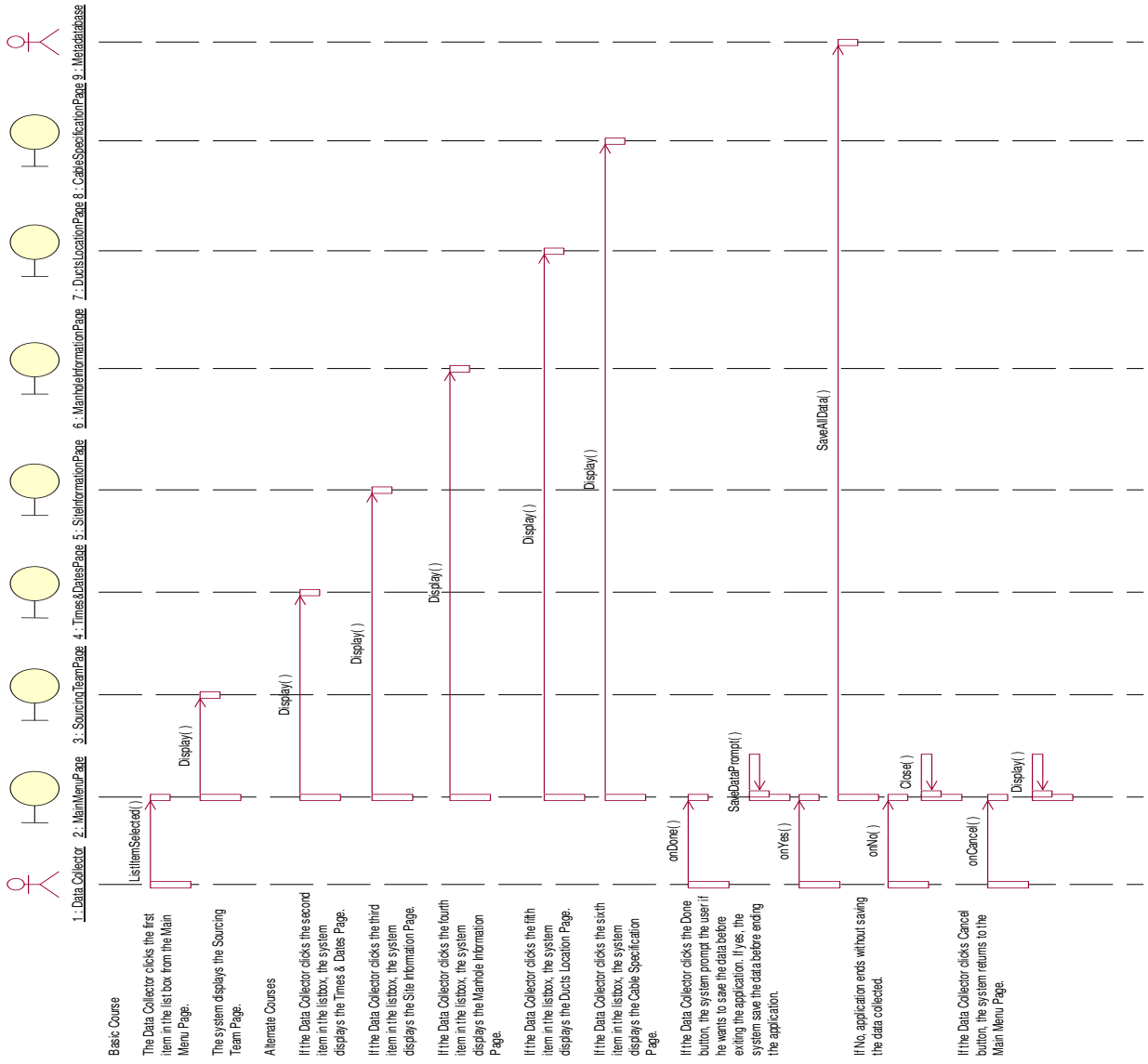



**Figure 72: Sequence Diagram for Select Sourcing Team Use Case**

Basic Course

The Data Collector clicks the Add Member button on the Sourcing Team Page.

The system displays the Sourcing Team Section Page.

The systems retrieves the names of all the members from the persistant team list.

The Data Collector clicks member/s from the list and clicks Add.

The system adds the selected members to the listbox in the Sourcing Team Page. And then the system returns the user back to the Sourcing Team Page.

Alternate Courses

If the Data Collector clicks Select All Members button, the system selects all the members in the listbox.

If the Data Collector clicks the Update Member Info button, the system passes control to the Update Sourcing Team Details use case.

If the Data Collector clicks the Cancel button, the system returns to the Sourcing Team Page without adding any members to the list.

If the Data Collector selects memeber/s and click Remove member, the system removes selected memebers from the list.

If the Data Collector clicks the <Index link, the system passes control to Selecting Sourcing Template use case.

If the Data Collector clicks the page2 > link, the system passes control to Enter Times & Dates use case.

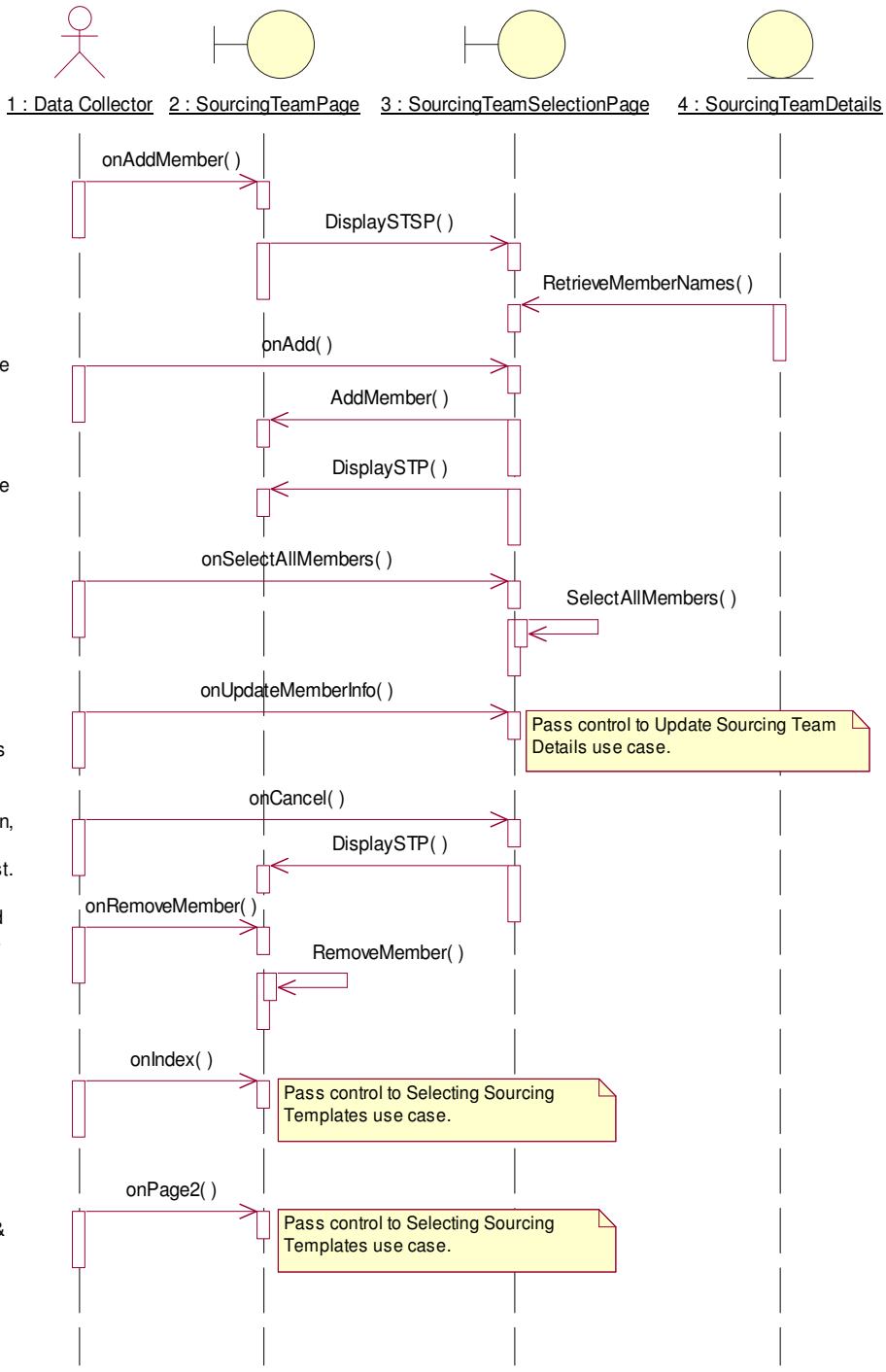



**Figure 73: Sequence Diagram for Specify Sourcing Team Use Case**

Basic Course

The Data Collector clicks the Update Member Info button on the Sourcing Team Selection Page.

The system displays the Members Info Page.

The system retrieves all the member information and displays them on to the table in Member Info Page.

The Data Collector clicks the row of the table to select a particular member.

The system enables the Edit and Delete buttons.

The Data Collector clicks the Edit button.

The system displays the Edit Member Info Page with the selected member details.

The system opens the SIP.

The Data Collector enters the data and clicks OK button.

The system closes the SIP.

The system updates the changes to the persistant member information, and then returns the Data Collector back to Member Info Page.

Alternate Courses

If the Data Collector clicks the New button, the system displays the Edit Member Info Page with empty data fields. The sequence flows as the basic flow.
After the Data Collector clicks OK, system displays the Members Info Page and creates a row for a new member.

If the Data Collector clicks the Delete button, the system removes the table row of the chosen member, and the system updates the changes with the persistant member information.

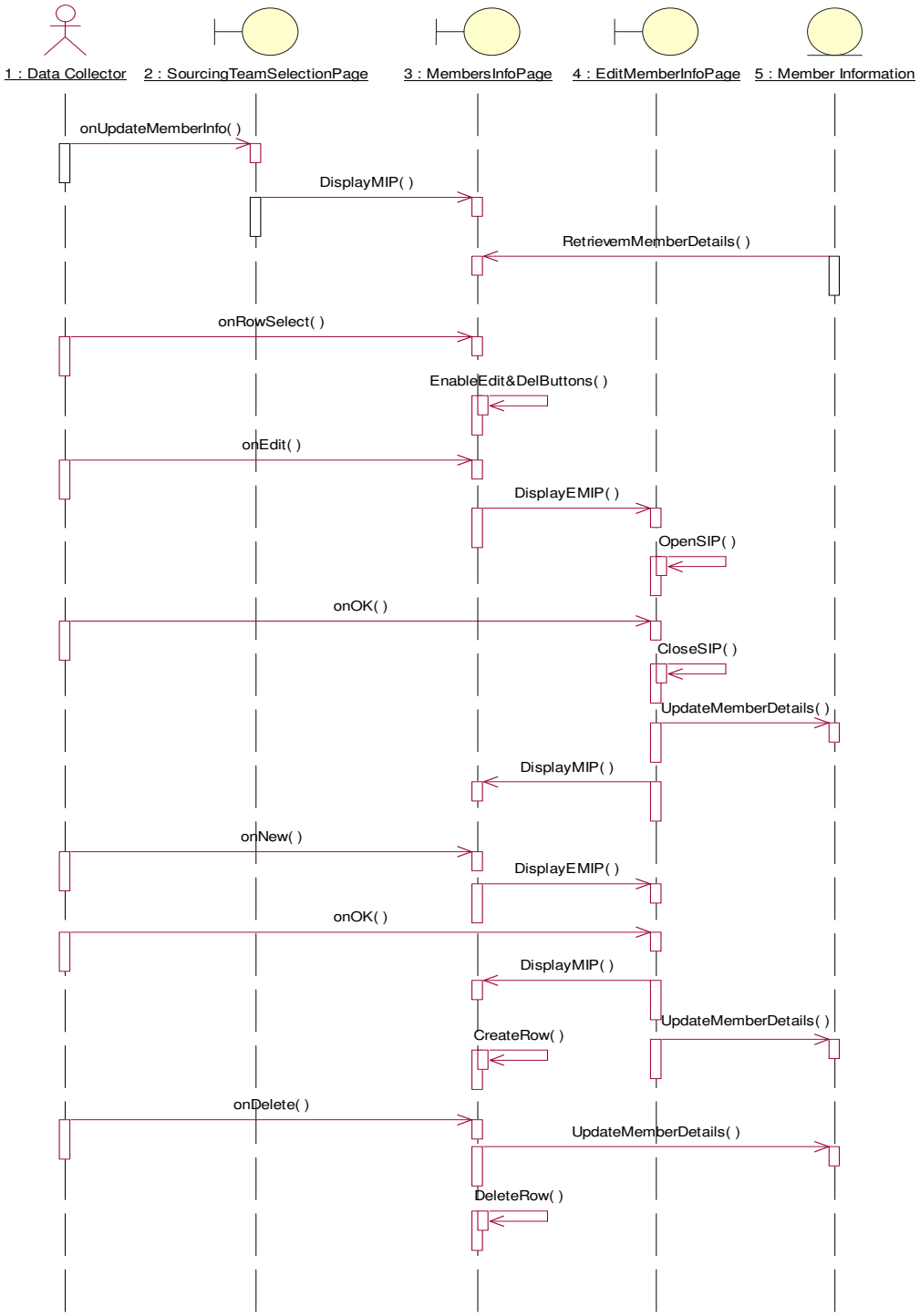



**Figure 74: Sequence Diagram for Update Sourcing Team Use Case**

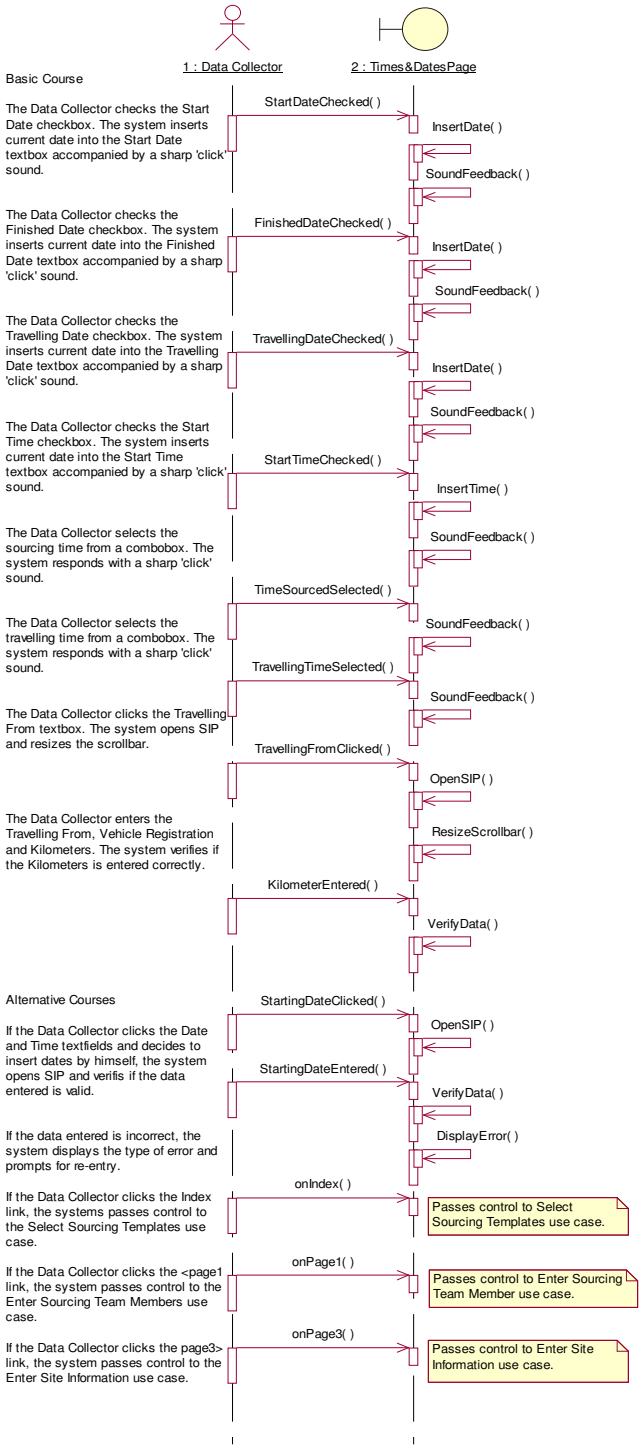



**Figure 75: Sequence Diagram for Specify Times & Dates Use Case**

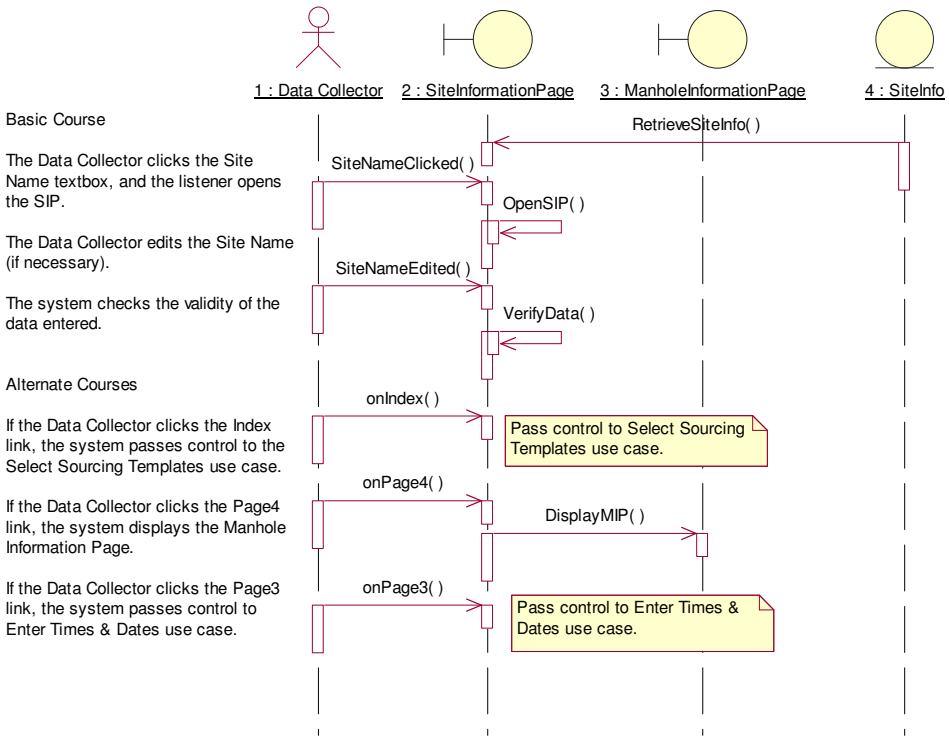



**Figure 76: Sequence Diagram for Specify Site Information Use Case**

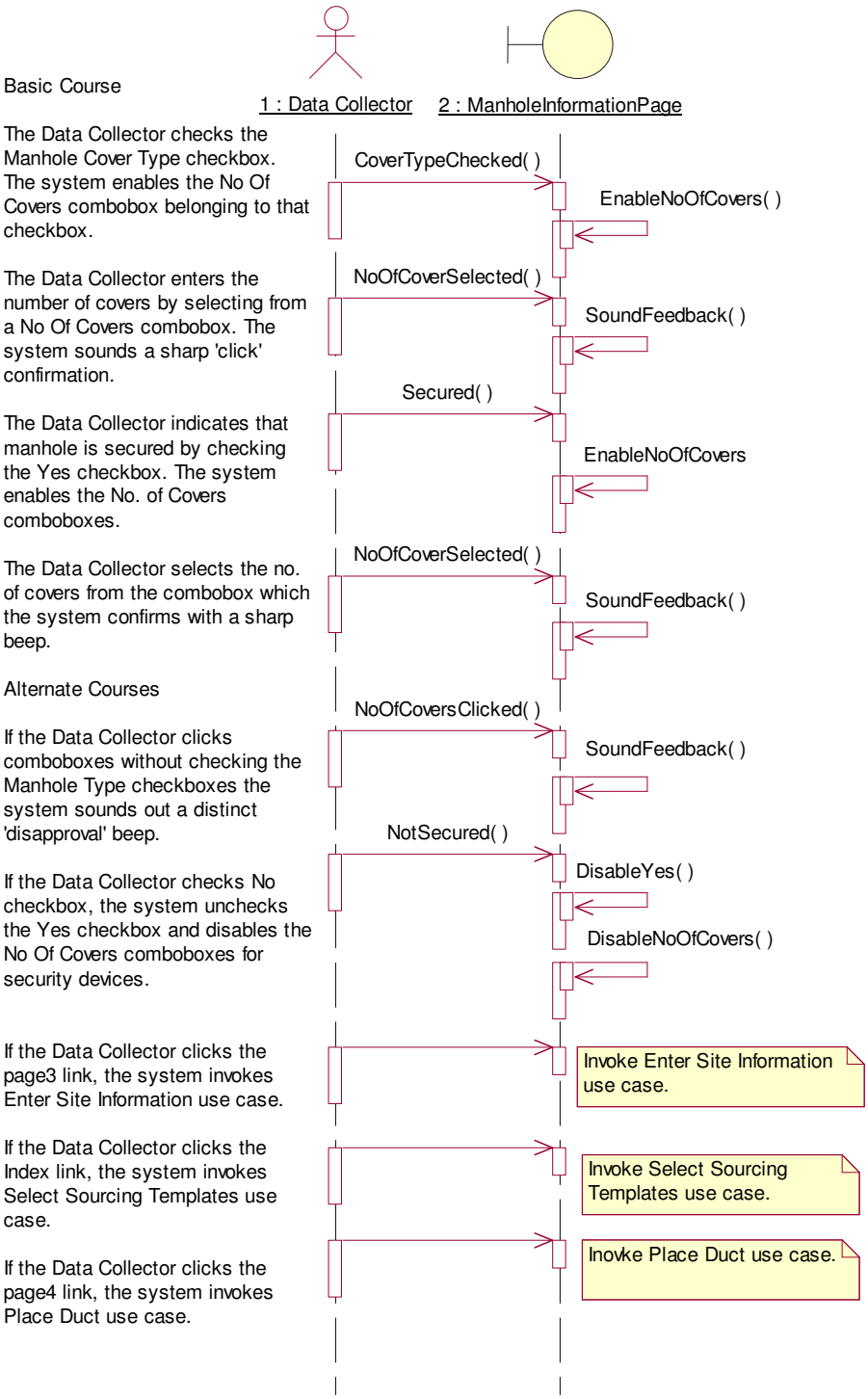



**Figure 77: Sequence Diagram for Specify Manhole Cover Types Use Case**

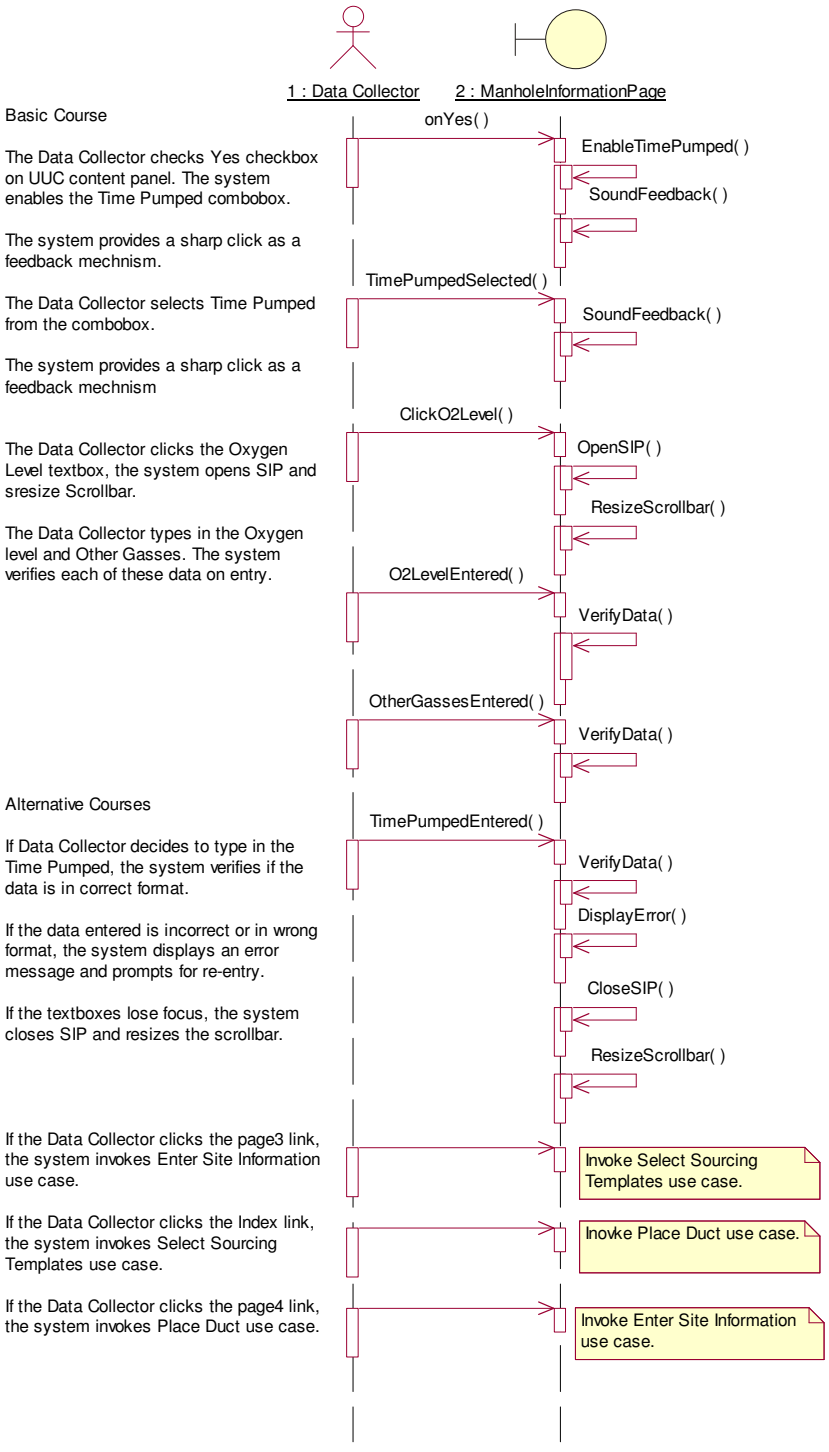



**Figure 78: Sequence Diagram for Specify UUC Content Use Case**

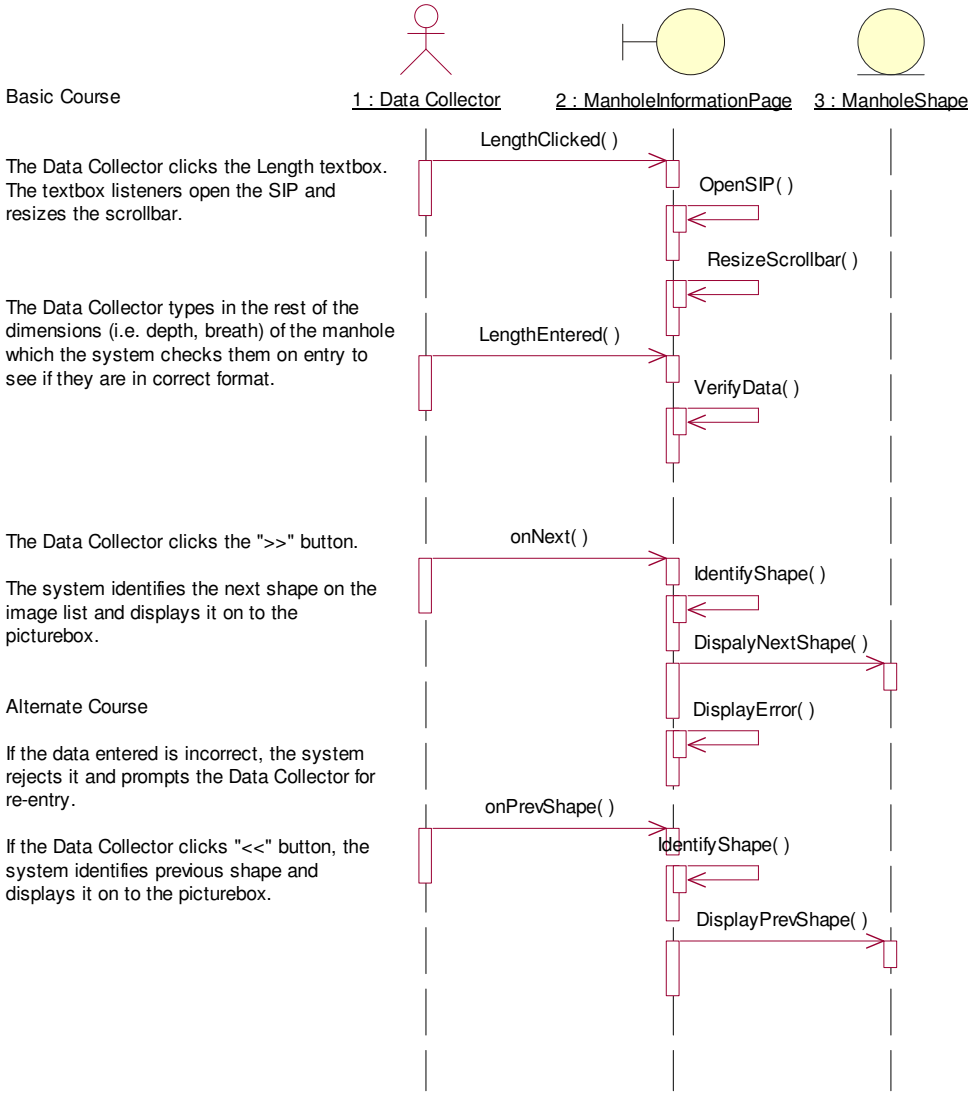



**Figure 79: Sequence Diagram for Specify Manhole Dimensions Use Case**

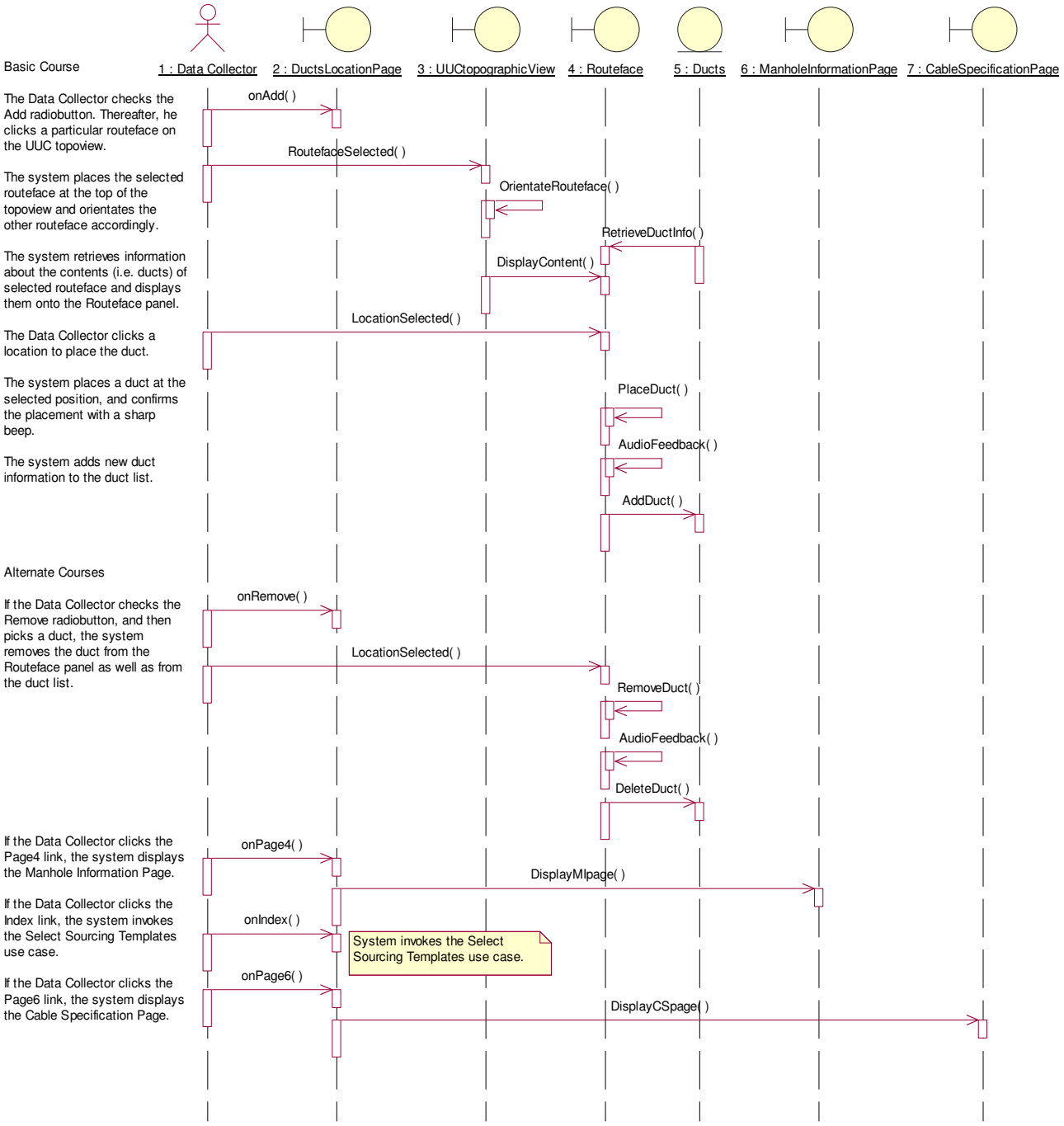



**Figure 80: Sequence Diagram for Place Duct Use Case**

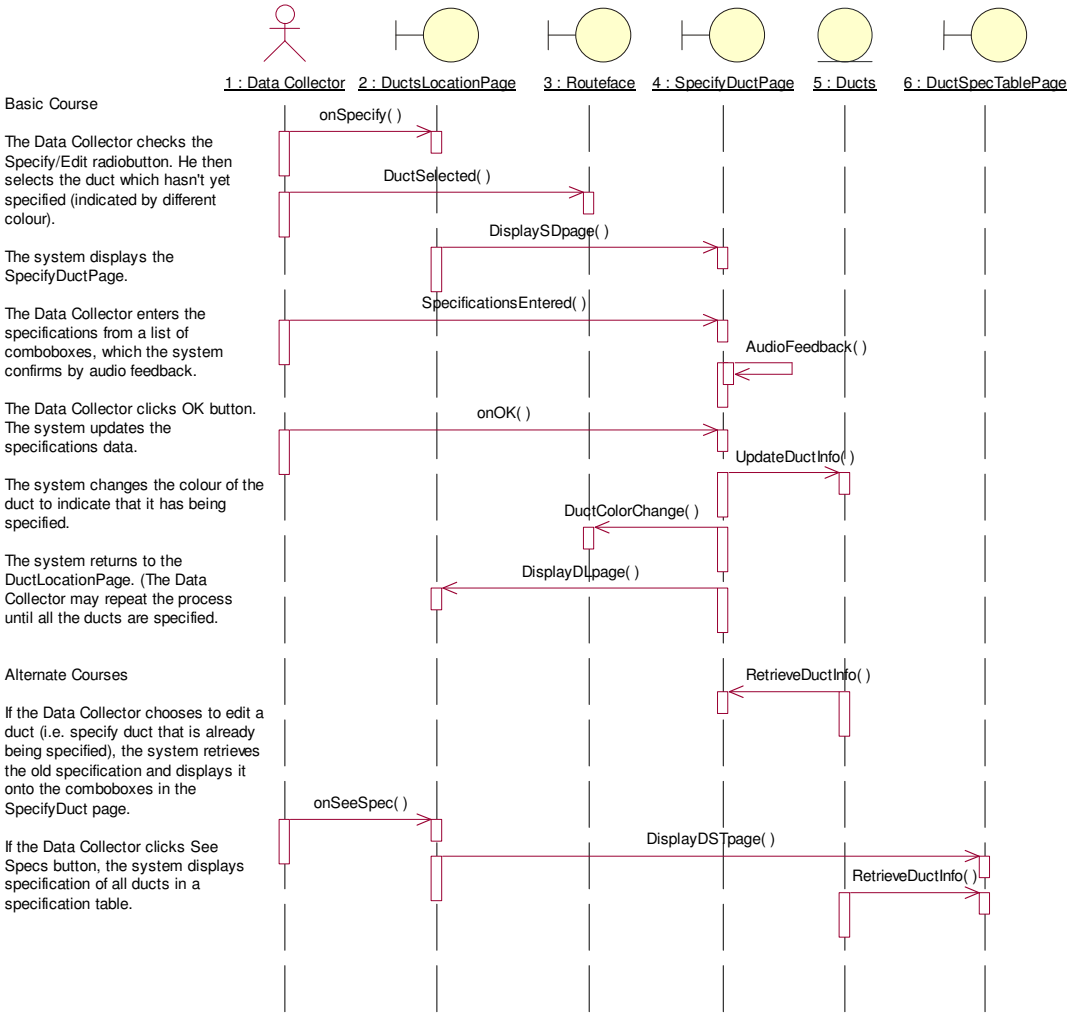



**Figure 81: Sequence Diagram for Specify Duct Use Case**

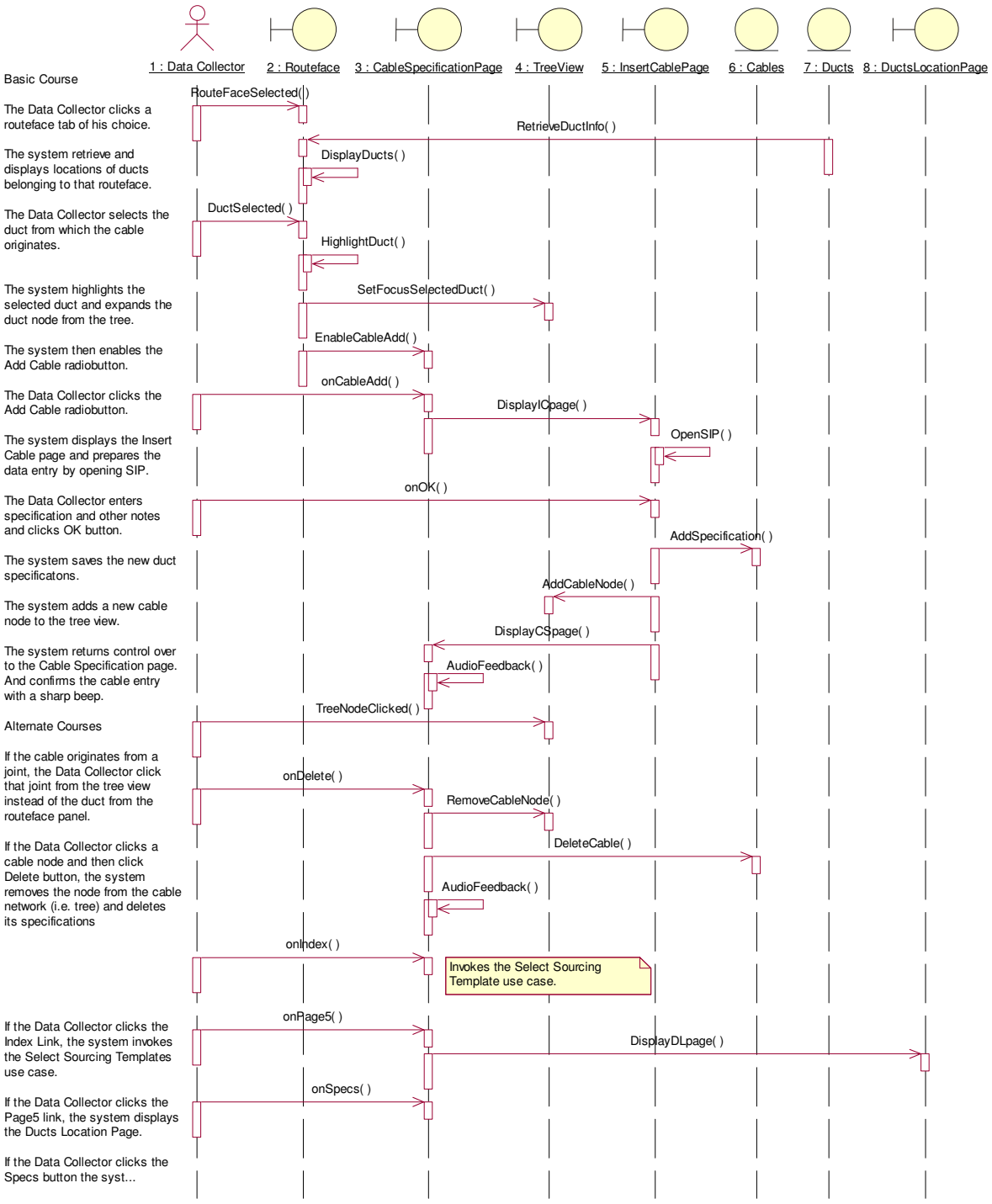



**Figure 82: Sequence Diagram for Place Cable Use Case**

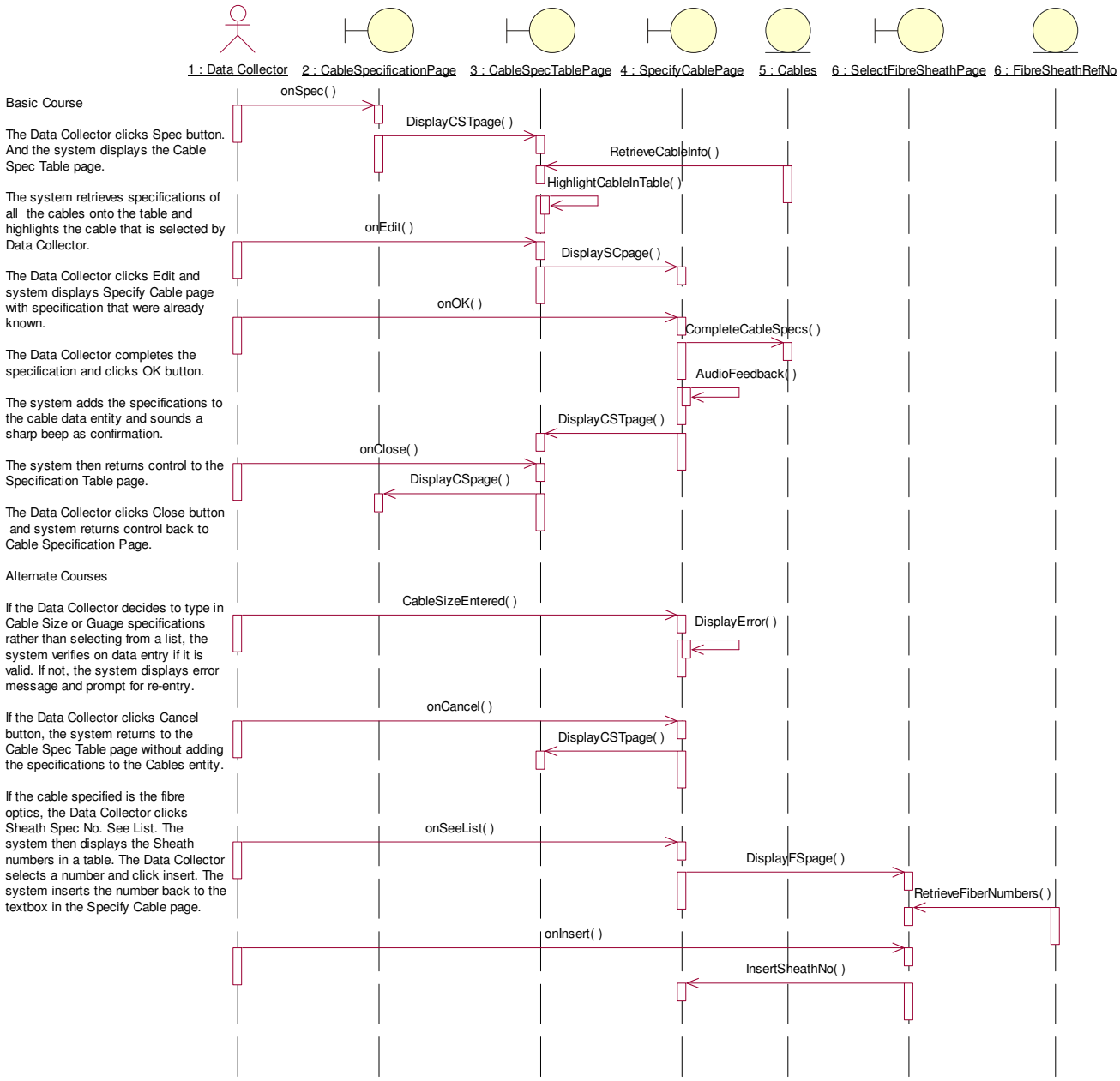



**Figure 83: Sequence Diagram for Specify Cable Use Case**

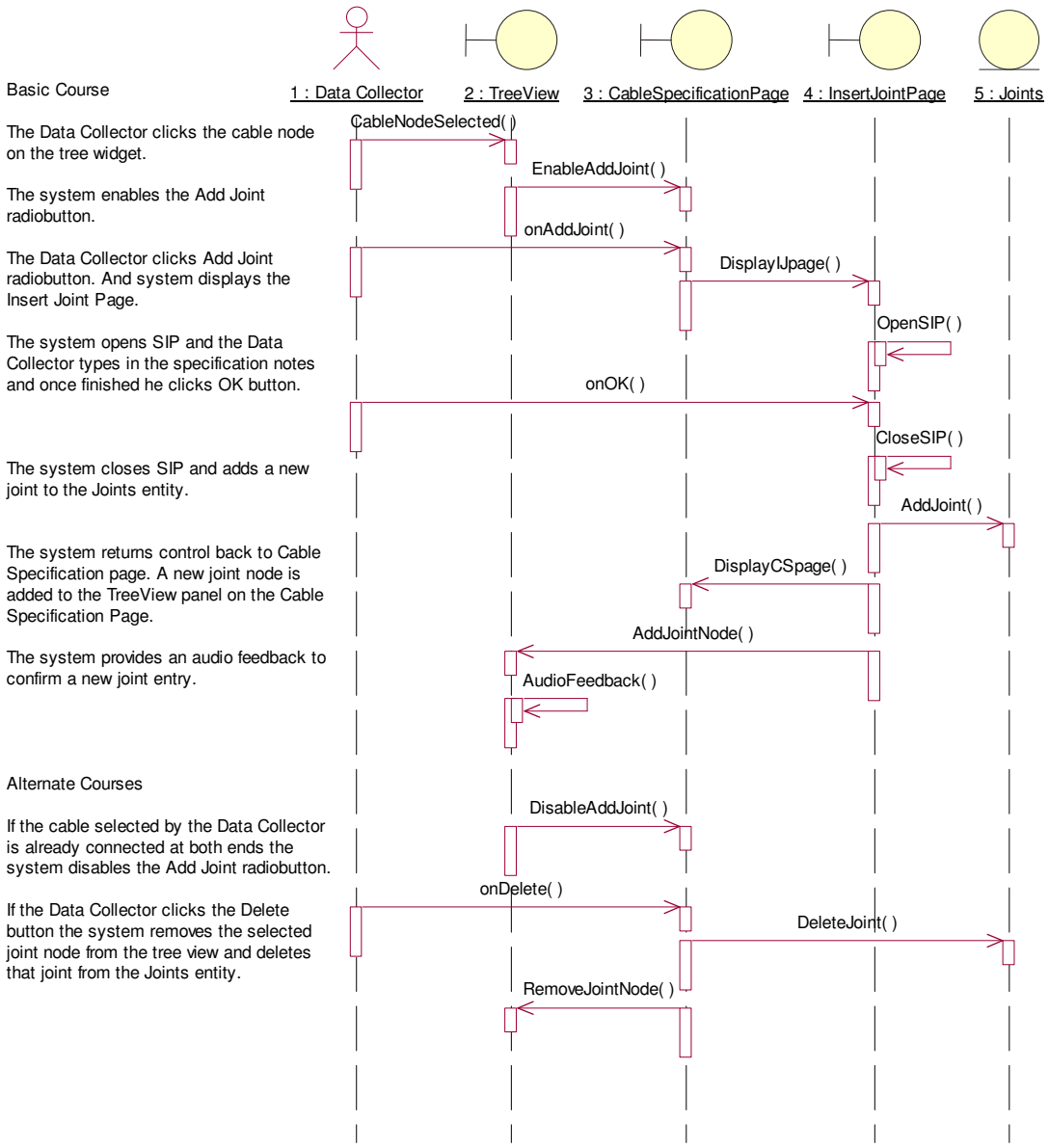



**Figure 84: Sequence Diagram for Place Joint Use Case**

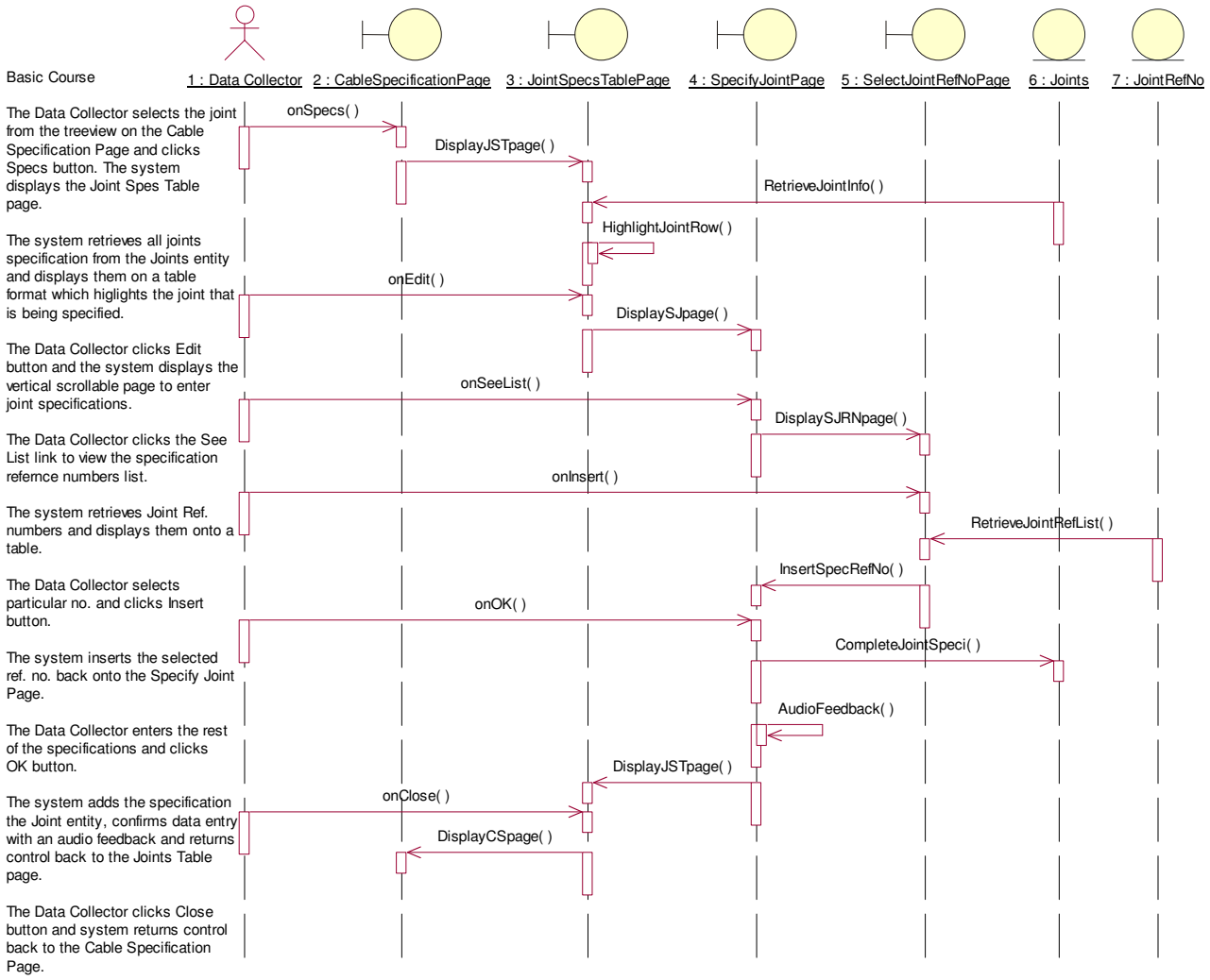



**Figure 85: Sequence Diagram for Specify Joint Use Case**

# 6 Class Diagram

The class diagram (see section 2.2.5) for the field data collection application is presented below. The functions (or methods) and attributes of the classes are derived from the sequence diagram in section 5.

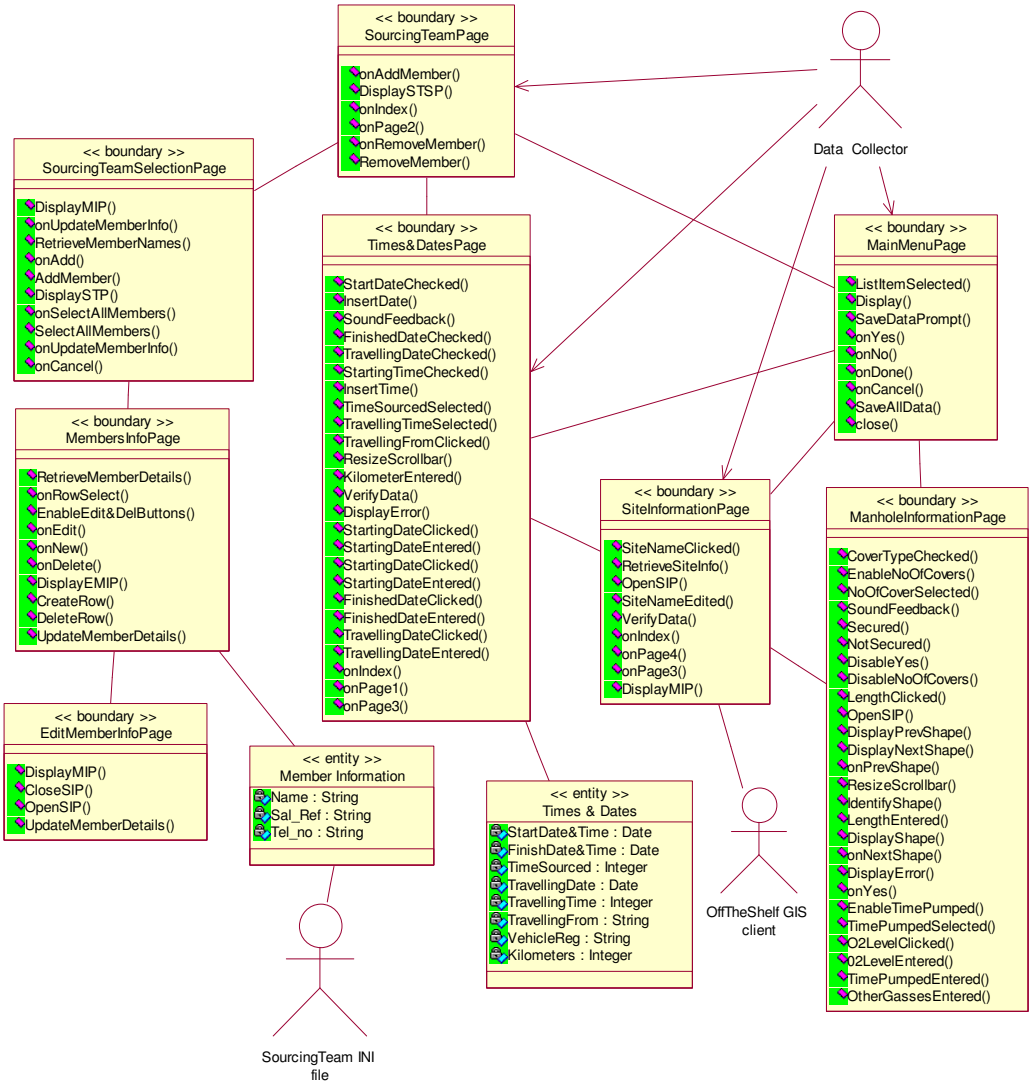



**Figure 86: Class Diagram (part 1) for Field Data Collection Application**

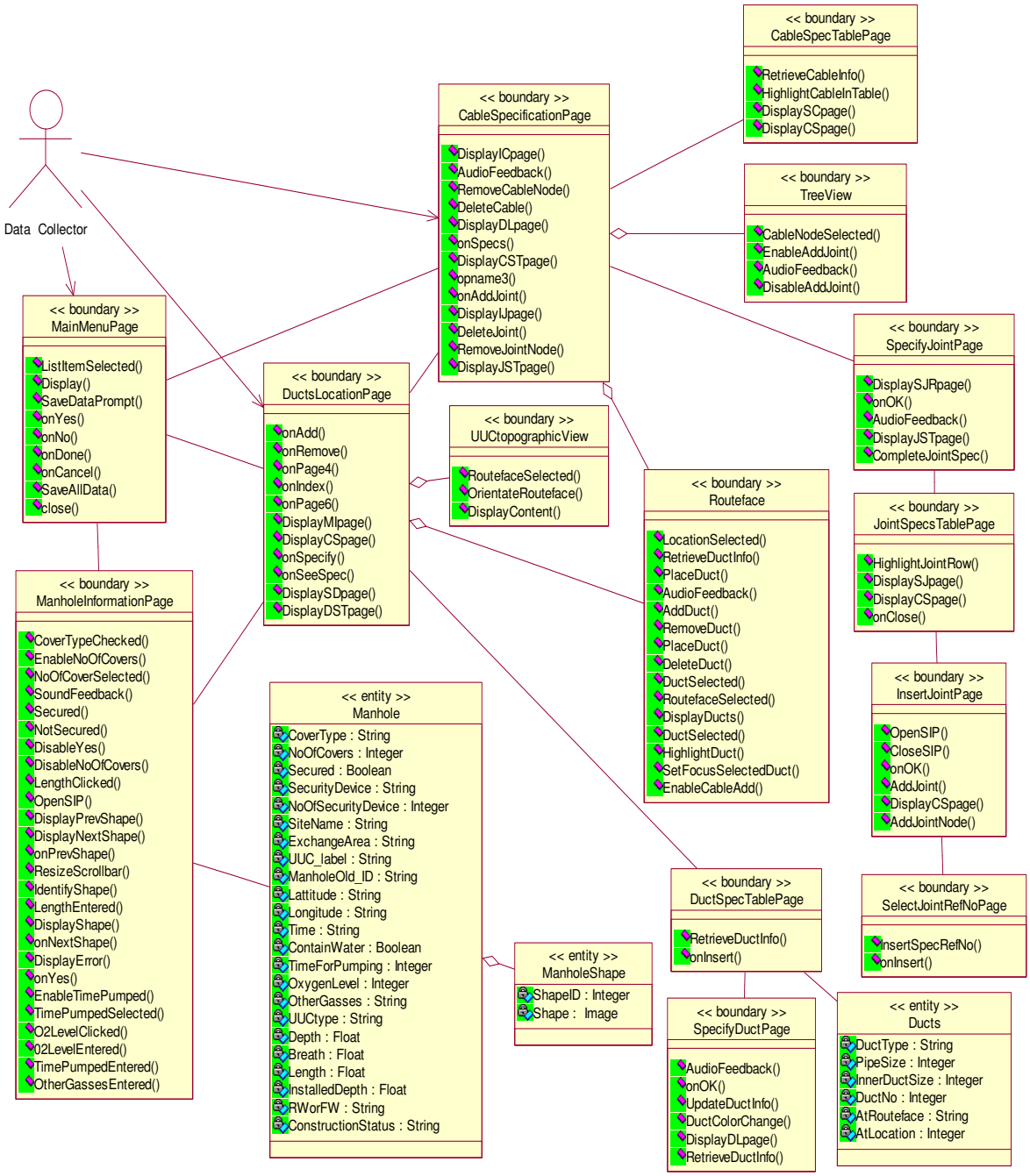



**Figure 87: Class Diagram (part 2) for Field Data Collection Application**

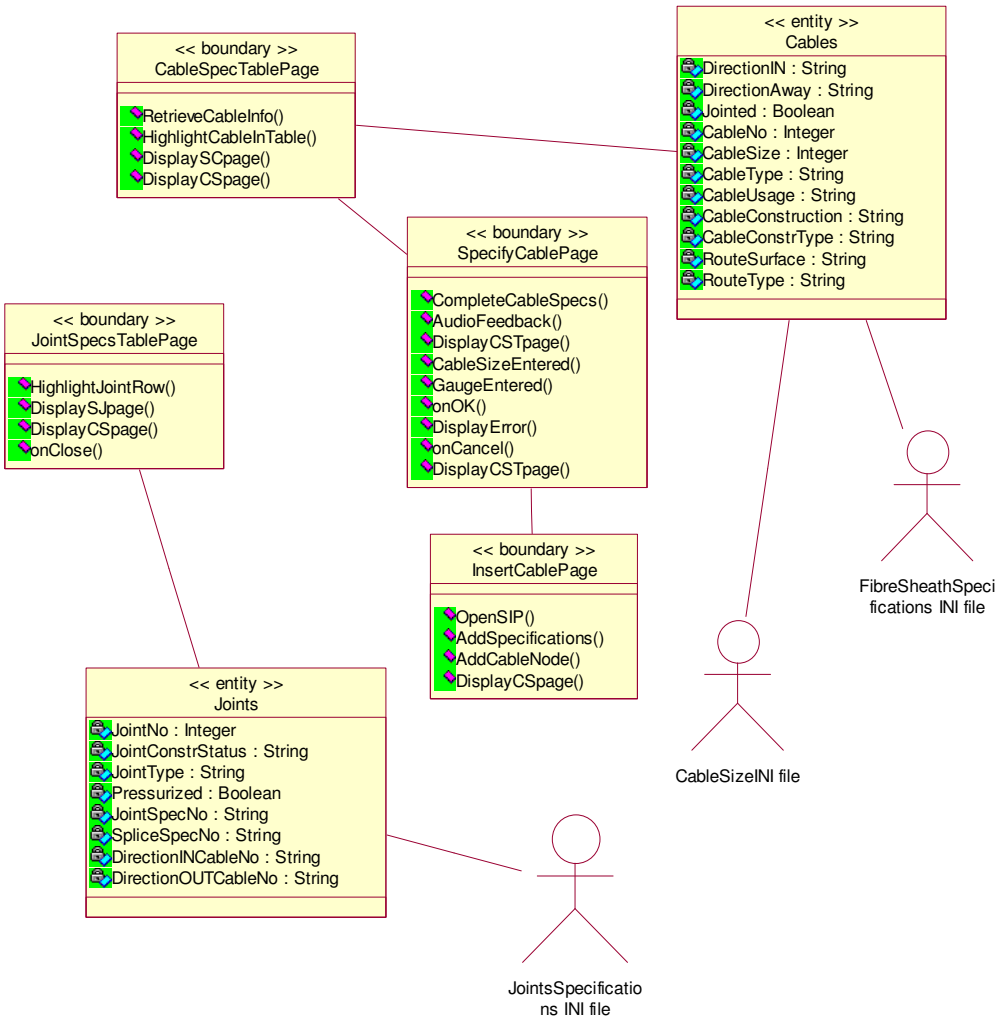
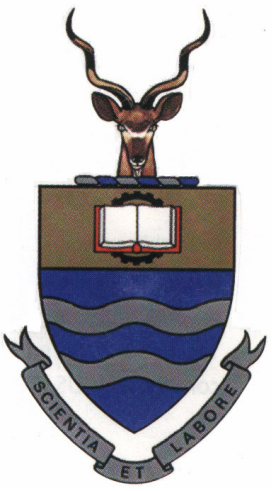



**Figure 88: Class Diagram (part 3) for Field Data Collection Application.**

# 7 Conclusion

The software usability is not merely how the user interface looks. To a large extent, it concerns the behavior of software particularly how it interacts with the user. With this in mind, the functionality of field data collection application is presented in the software design. The software design process used, namely the ICONIX provides a framework to comprehend not only the behavior but also the structure and complexity of software as well. It is felt that the ICONIX design process is able to do this sufficiently without the complexity of a much larger object-oriented design process like Rational Unified Process (RUP).

Some of the contents of software design such as the class diagrams do not really affect the software usability. However, these models allow the software developers to extend the existing functionality and even to continue with the implementation of the prototype to fully functional software. This project, however, has not implemented the software design into a complete, fully functional software system since it is deemed unnecessary in terms of the objective of this research. Nevertheless, object-oriented design will enable easier reuse and maintenance of software components especially when sourcing templates are modified regularly. Overall, the software design process used has adequately explained the behavior and functionality of field data collection application. However, it is felt that a superior design could be achieved if more time was spent on improving the design in an iterative manner.




# Usability Evaluation

Doc. No. 5

# CONTENTS

CONTENTS I

LIST OF FIGURES II

LIST OF TABLES III

1 SCOPE 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

2 USABILITY EVALUATION METHOD 2

3 TEST PROCEDURE 4

3.1 Testing Technique 4

3.2 Test Participants 6

3.3 Test Cases 6

3.4 Test Environment 7

3.5 Test Equipment 8

3.6 Test Data 8

4 TEST RESULTS 10

4.1 Performance Data 10

4.2 Subjective Data 15

5 TEST LIMITATIONS 16

6 DISCUSSION OF TEST RESULTS AND RECOMMENDATIONS 18

7 CONCLUSION 24

# LIST OF FIGURES

*Figure 1: Usability evaluation in a field setting* 8
*Figure 2: Times taken to complete data sourcing tasks using prototype versus paper sourcing template.* 11
*Figure 3: Subjective viewpoints on the difficulty of test cases* 15
*Figure 4: "Triangulation" – using multiple sources of data to find usability problems (Source: Dumas J et al., 1999b)* 19
*Figure 5: SIP concealing the textfields* 20

# LIST OF TABLES

*Table 1: Test cases* _____ *10*
*Table 2: Errors reported from each test case and their recovery times* _____ *12*

# 1 Scope

## 1.12 Introduction

The field data collection prototype is evaluated for usability with the intended users in a field setting. Although the usability test alone does not solve usability problems, uncovering them during the usability test is an important feedback to improve the prototype before final implementation. Effectively, the test results are then used to fine tune critical features of the usability design. This document describes the usability evaluation techniques, test environment and participants used for the field data collection prototype. In addition, the test results are explained and appropriate recommendations are made for further development of the prototype.

## 1.13 Purpose

The purpose of this document is to present the reader with the findings and recommendations of usability evaluation of the field data collection prototype.

## 1.14 Audience

The intended readers of this document include researchers involved in the areas of software usability, mobile computing, as well as software designers, developers, the external examiner and other interested parties.

# 2 Usability Evaluation Method

The two most common usability evaluation techniques (see section 4.4, Doc. No. 1) are considered for the field data collection prototype. They are as follows:

- Usability inspection or walkthrough methods including heuristic evaluation and expert evaluations.

- Empirical usability testing in laboratory or field setting.

The following questions (Karat C *et al*., 1992) are addressed in selecting a suitable usability evaluation technique:

a) *Usability problems*: Is it better than others in identifying usability problems?
b) *Cost-effectiveness:* Is it a relatively cost-effective technique?
c) *Individuals versus teams:* Does it require individual or groups of evaluators?
d) *Evaluator expertise:* Who are the evaluators? Can they be software engineers or should they be exclusively human factors experts?

A number of studies have compared different types of usability evaluation techniques to determine which are suitable for a particular situation. For example, Jeffries (1991) used UI experts to evaluate GUI based systems using walkthrough and empirical usability testing method. He found that heuristic evaluation identified more usability problems and more of the serious problem than the empirical usability testing. In an another study (Karat C *et al*., 1992), a group of non-UI experts and usability engineers were asked to evaluate usability using heuristic and empirical usability testing methods respectively. The results show that empirical testing condition identified the largest number of usability problems, and identified a significant number of relatively severe problems that were missed by the walkthrough method. Although it may be obvious, the findings suggest that the more

skilled and experienced the evaluator is, the more likely the usability problems that will be identified.

The walkthrough methods such as the heuristic evaluation achieve better results when performed by a group of evaluators. Each evaluator typically identifies between 20-30% of the usability defects (Nielsen J *et al*., 1990), but pooling the results of evaluations improves the overall percentages. Nielsen (1990) also found that individual evaluators who are not human factors experts are mostly quite bad at doing heuristic evaluation and he recommended that this method be done with between three to five evaluators and supplement this method with other evaluation methods to increase the total number of usability problems found. He also highlighted a few shortcomings with the heuristic evaluation method. In particular, it identifies usability problems without providing direct suggestions for how to solve them. A major advantage of reviews, inspection and walkthrough methods is that they can be done relatively cheaply and quickly. In contrast, usability laboratory and testing equipments used in empirical usability tests are expensive and not always available. In addition, it takes longer to execute the empirical usability test because of the time required to set up test equipments and arrange appointments with the participants. However, a field test is usually carried out without expensive testing equipments and hence it too can be done relatively cheaply.

Having mentioned some of the advantages and disadvantages of usability evaluation techniques, a field test is considered to be the most suitable technique for field data collection application. Unfortunately, the number (five or more people) of UI experts needed to effectively evaluate the prototype was not available. The use of usability laboratory was also not an option because of the high cost. A field test, on the other hand, is a more convenient choice because the fieldworkers are already available as test participants. More importantly, testing in a field setting provides a greater realism and comfort for the fieldworkers in their familiar environment. Some usability practitioners (Thomas P, 2002) go as far as saying traditional usability testing in a laboratory setting to be meaningless in a context of mobile usage.

# 3 Test Procedure

Now that the empirical usability testing in the field is selected as the most appropriate usability evaluation method, this section describes how the test was actually carried out. The following steps summarize the usability testing process:

1. Prepare tasks, questionnaires and data logging tools to be used during testing.
2. Select and train participants (represent intended users) to do tasks.
3. Observe and record what they do and say.
4. Analyze the data, diagnose usability problems and recommend changes to fix those problems.

## 3.1 Testing Technique

The nature of testing techniques used could be described as an *active intervention* (Dumas J *et al*., 1999a) where a test conductor takes an active role during the test. In other words, the test participant is actively probed for his understanding of whatever is being tested on by asking questions throughout the test. There are a number of reasons why this particular technique is useful. Firstly, by asking questions during the test rather than in the end, one is able to get insights into participant' state of mind at the moment when a particular task is performed. Otherwise, the participant forgets or is overwhelm by tasks that are more prevalent than others. Secondly, it is an excellent technique to use with the working prototypes because it provides a wealth of diagnostic information (Dumas J *et al*., 1999a). And finally, the fieldworkers are new to the usability testing process (any software testing for that matter) and hence close interaction with them during the test provides reassurance, improves their cooperation and hence increases chances of finding usability problems. Before the usability test could begin, test participants were informed about the test and what they were required to do. In addition, they

were given a demonstration on how to use the PDA and the field data collection prototype. Some kind of training is necessary because it gives participants a chance to orientate and familiarize with the prototype especially when they haven't used a PDA before.

During the test, the participants were asked to "think out loud" as they were doing the tasks. In other words, the participant speaks out whatever he is thinking (intentions, frustrations and comments etc). However, some people find it awkward and have difficulty sustaining this kind of commentary. The participant may be embarrassed to express his mind-set and often, there is a tendency to "censor", mentally correcting mistakes of logic or reasoning without reporting them (Constantine L *et al*., 1999f). Hence, a paired-subject strategy (also called co-discovery) is used where the participants are assigned into two-person teams and they complete the tasks together. Some researchers agree that the co-discovery often yields more information about what the users are thinking and what their strategies are in solving their problems than by asking individual participants (Hackman G *et al*., 1992). Although two people are involved in the test only one person is actually doing the tasks while his companion keeps the commentary running sharing ideas and making comments. This open approach worked well for the fieldworkers who are accustomed to working collaboratively in a team.

The participants were allowed to ask for help if they find that they cannot complete an assigned task. These helps were then documented as part of usability shortcomings of the prototype. This approach serves as a kind of helpdesk and not to be confused with influencing participants or giving hints in doing the tasks. In a traditional usability testing, some kind of help system is used to similar effect whether it is a help manual or some on-line help system. The participants were interviewed after completion of each task as well as after the test using a set of pre-test and post-test questionnaires (for the type of data collected, see section 4).

## 3.2 Test Participants

The same telecommunication fieldworkers from the Telkom data sourcing team participated in the usability test. The data sourcing team included three fieldworkers that participated in the usability test. The question of how many participants to be included in a usability test is one of intense discussions by the usability community. In general, usability tests do not require a large number of test participants. In fact, most major usability problems in a usability test can be found with relatively few participants. For example, Virzi (1992) found that 80% of the usability problems were detected with between 4 and 5 participants and 90% were detected with 10 participants. Additional participants were less and less likely to reveal new information. He also suggested that a usability test should have at least two or three participants so that test results do not reflect some idiosyncratic behaviors.

Unlike some statistical analysis, one doesn't need a random sample of possible end users as participants in a usability test. Instead, a *convenient* sample is used where people from appropriate population whom happen to be available. This was particularly the case with the telecommunication fieldworkers from Telkom who were able to participate in the usability test.

## 3.3 Test Cases

One of the essential requirements of a usability test is to determine what to test, more specifically, which tasks the users should do. Unfortunately, there are more tasks than there is time available to test them. Determining the appropriate task scope is not always simple. If the tasks scope is too narrow, the users feel that they are in an artificial situation and they are not likely to approach the tasks in the same way as they normally would. Making the task too broad can also become difficult to keep the user focused for the duration of the test. The right approach, therefore, is to concentrate on the main tasks that probe the most potential usability problems. These main tasks are readily derived from the *essential* use cases (described in section 5, Doc. No. 2) in the task model. They are as follows:

1. Enter sourcing team information
2. Enter sourcing time
3. Enter site information
4. Specify manhole covers
5. Specify UUB content
6. Specify UUB dimensions
7. Place duct
8. Specify duct
9. Place cable
10. Place joint
11. Specify cable
12. Specify joint

The test participants attempted the tasks in the order that they would complete in a typical sourcing work. The name of the task actually describes what the task is about. Usually, scenarios are used to describe the tasks so that it takes some of the artificiality out of the test. Bear in mind that a traditional usability test uses participants that come from different backgrounds and hence scenarios help them understand about the tasks and why they are doing them. For this project, however, use of detailed and descriptive scenarios is really unnecessary because the test participants are all fieldworkers and they are already familiar with these tasks.

## 3.4 Test Environment

The usability test was carried out at the actual data sourcing site inside the UUC (see Figure 1). A fairly typical UUC large enough to fit five people was chosen. Although the test coordinator had very little control over test conditions, the field setting provided the participants with more realistic test environment. This is particularly important because people tend to think and act differently when removed from their natural work setting. In addition, a conventional usability test

would not be able to simulate all the environmental factors (noise, light conditions etc.) inside the UUC and collaborative activities between the fieldworkers.




**Figure 89: Usability evaluation in a field setting**

## 3.5 Test Equipment

No sophisticated recording equipment or data capturing software was used in the usability test. Instead, a paper data-logging form (see Doc. No. 7) and a stop watch were used to record the test data. A minimal use of test equipment allowed the testing to take place in a confined space of UUC. It also kept the usability evaluation within the realistic environment of the UUC without turning to some kind of laboratory.

## 3.6 Test Data

The data collected from the usability test includes:

1. Subjective measures – the participant's perceptions, opinions and judgments.

2. Performance measures – counts of actions and behaviors observed during the test.

Comments from the participants were obtained during and after the test. During the test, spontaneous comments from the participants and their facial expressions (frustration, confusion and delight etc) were noted. In addition, the help questions asked by the participants were recorded. The participant was given one task at a time and at the end of each task, the participant was asked to comment on the difficulty of that task. Although this is a subjective measure, it is rated quantitatively. For example, a “very easy” and “very difficult” is rated as “1” and “5” respectively on a 5-point scale. The test coordinator also recorded own comments into the data-logging form during and after the task is performed. The performance data includes the time spent on:

- completing a task
- asking for help
- recovering from errors

A post-test questionnaire was used to gather the participant’s judgment and comments addressing overall usability of the prototype. The participants were also asked to comment on a number of screenshots of user interface that are linked to specific usability design features. (The data-logging forms and post-test questionnaire are shown in Doc. No. 7.)

# 4 Test Results

## 4.1 Performance Data

The performance data collected during the usability test includes the times taken to complete test cases (see Table 1). However, these performance data are rather meaningless unless we can compare them to something. Hence, the times taken to complete the data sourcing tasks using the paper sourcing forms were also recorded for comparison. In the first part of the test, the participants were asked to source the data as they would normally do using their paper sourcing forms. The one participant entered the information while the other two assisted him in finding the asset information. The data-entry participant was given a set of tasks or test cases that he must complete. These test cases came out slightly different to the ones given in **section 3.3** due to circumstances surfaced during the test. Some of the tasks were done so quickly that it was not practical to record the times. For example, to place a single duct takes less than two seconds to complete. Hence, a number of "small" tasks are combined into a single test case (see Table 1).

**Table 2: Test cases**

| Test Case No. | Description |
|---|---|
| 1 | enter sourcing team information |
| 2 | enter sourcing time |
| 3 | enter site information |
| 4 | specify manhole covers, UUB content, UUB dimensions |
| 5 | place 37 ducts |
| 6 | specify 37 ducts |
| 7 | place a single copper cable |
| 8 | place 5 cables with 2 joints |
| 9 | place a single fibre optic cable |
| 10 | place 2 cables with 1 joint |
| 11 | specify a single copper cable |
| 12 | specify a joint |
| 13 | specify a single fibre optic cable |

The time that took to complete the test cases using the paper sourcing forms and the field data collection prototype are compared in Figure 2. Before analyzing the plot, bear in mind that the participants have been using the paper sourcing forms for many months whereas the prototype was introduced for the first time (the training aside). Hence, one would expect the fieldworker to complete the test cases quicker using a paper sourcing form than the prototype. The graph in Figure 2 confirms this supposition where most of the tasks are completed quicker using the sourcing form. And then there are a few tasks that are done quicker using the prototype. The question that needs to be answered here is that what make some tasks quicker than others. In addition, the magnitude of speed difference needs to be explained as well.

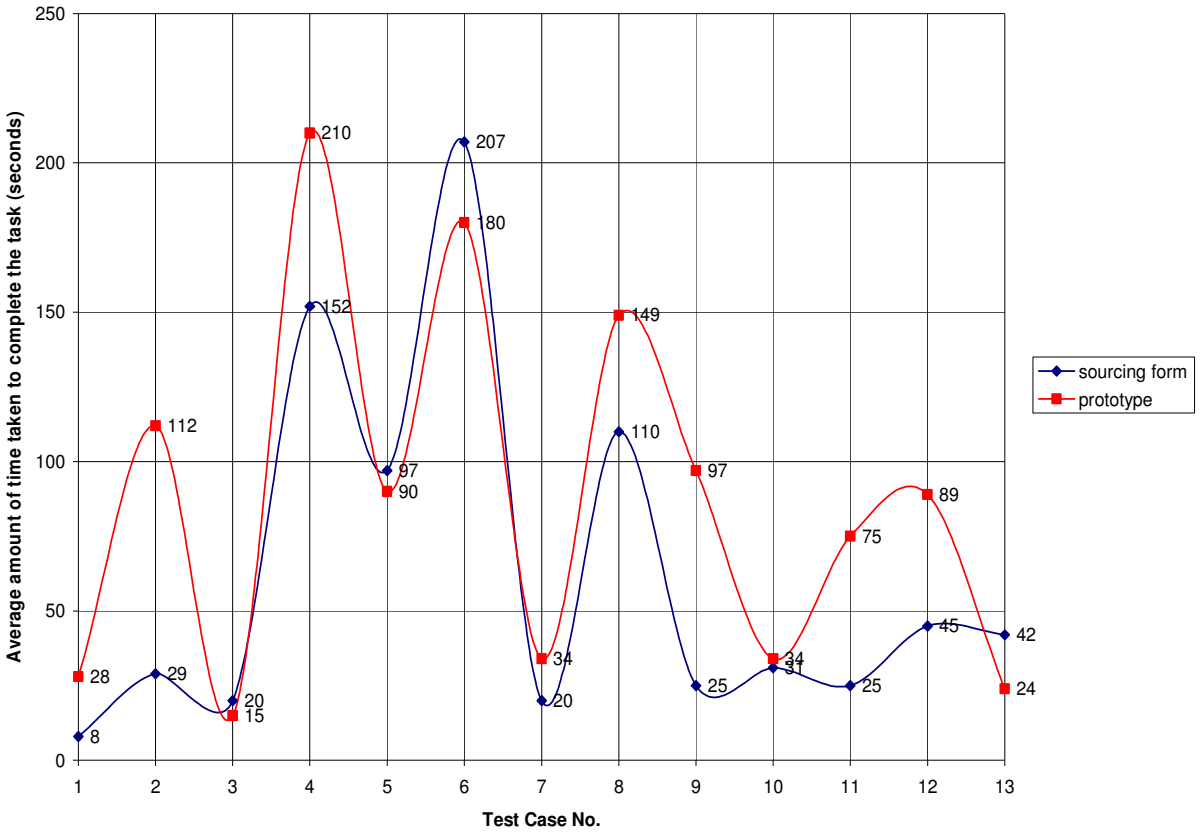



**Figure 90: Times taken to complete data sourcing tasks using prototype versus paper sourcing template.**

The approach adopted in this project is to analyze the test data from several sources at the same time (Dumas J, 1999b). In other words, one looks at all the data together to see how each set of data supports the other. The test data in Figure 2 is analyzed together with the usability problems (shown in Table 2) the users encountered when doing these tasks. They provide important clues in explaining the cause of usability problems, for example – why there are delays in completing a task. As shown in Table 2, these problems are very specific. This is because the main objective of a usability test is to find the "real problems" as opposed to finding existence of some phenomenon (Dumas J, 1999a).

**Table 3: Errors reported from each test case and their recovery times**

| Test Case No. | Errors | Average recovery time (s) |
|---|---|---|
| 1 | none | 0 |
| 2 | SIP covering textbox | 10 |
| 3 | none | 0 |
| 4 | scrollbar swinging off | 5 |
| 5 | delay with background colour changes | 0 |
| 6 | none | 0 |
| 7 | none | 0 |
| 8 | none | 0 |
| 9 | none | 0 |
| 10 | textbox confusion | 3 |
| 11 | scrollbar swinging off | 5 |
| 12 | scrollbar swinging off | 5 |
| 13 | none | 0 |

From looking at Figure 2, the test case no. 2 has the largest time difference. The participants had a problem with the SIP because when it opened it concealed the "Kilometers" textbox (see Figure 13 in Doc. No. 3). The user then had to scroll down a bit more and this created the delay. But this glitch does not explain relatively large time difference. On closer examination, the test case no. 2 required the user to type in the information for textboxes such as "Travelling From", "Vehicle Reg." and "Kilometers". The participants were also unfamiliar with the use of SIP and this

created further delays. They spent a large amount of time locating the letters on a soft-keyboard. To a similar effect, hand-writing recognition software did not work well for them. One participant, for example, wrote the letters too small that the system failed to recognize any of his inputs. Also, the participants found different "regions" of the SIP over which to write capital letters and numbers confusing.

The same trend is observed with the test case no. 4 where the user needs to type in the UUC dimensions such as height, width and depth. The test case no. 2 and no. 4 are very few tasks that require the user to type in the information. The rest of the tasks rely more on selections from pre-defined lists and menus. Interestingly, the test case no. 5 and no. 6 are quicker to complete using the prototype than the paper sourcing form. The automated labeling of ducts and use of combo-boxes made sure that these tasks are done quickly. However, there is a slight delay with the colour change for the ducts placement especially when the user taps the screen very quickly and repetitively.

Noticeably, the most prevalent usability problem is with the scrollbar. The participants prefer to use the scrollbar over navigation buttons to scroll up and down the screen. They often scrolled too fast that they missed the data fields that they were looking for. They became confused and felt that they were lost. This problem is particularly evident with the Manhole Information Page (see Figure 15-18, Doc. No. 3) which contains a relatively large number of data containers.

In order to quantify and summarize the overall performance of the prototype, a set of performance metrics is calculated. The five measures of performance are defined as follows (Constantine L *et al.*, 1999f):

1. **Completeness** – is the percentage of total assigned work completed within the allocated time. The time taken to complete the test cases using the data sourcing form is used as the allocated time.

2. **Correctness** – is the percentage of completed work that is correct. The completed work is the total number of data items collected.

3. **Effectiveness** – is correctly completed work as a percentage of total work.

4. **Efficiency** – is Effectiveness per unit time. In other words, the percentage of work correctly completed per unit time.

5. **Productiveness** – is the percentage of total test participant time spent productively. The unproductive time includes the time spent seeking help from the test conductor.

The performance measures are calculated as follows:

**Completeness** = $(T_f / T_p) \times 100$
= (811s/1137s) × 100 = 71.33 %

**Correctness** = ((correct data items)/ (total data items)) ×100
= (82/85) × 100 = 96.47 %

**Effectiveness** = (Correctness × Completeness) / 100
= (96.47 × 71.33) / 100 = 68.81 %

**Efficiency** = $(1/ T_p)$ × Effectiveness
= (1/1137s) × 68.81 = 0.061 % per second

**Productivity** = $((T_p - T_h)/ T_p) \times 100$
= ((1137s – 33s)/1137s) × 100 = 97.10 %

The $T_p$ and $T_f$ refers to an average amount of time taken to complete all the data sourcing tasks using the prototype and data sourcing form respectively. And $T_h$

refers an average amount of time taken to seek help from test conductor (i.e. unproductive time).

## 4.2 Subjective Data

The subjective data collected from the usability test is both quantitative and qualitative in nature. The former measures the "level" of difficulty of test cases from the test participant viewpoint. The participants were asked to rate the level of the difficulty of each test case from the magnitude of 1 to 5 (see Figure 3 below).

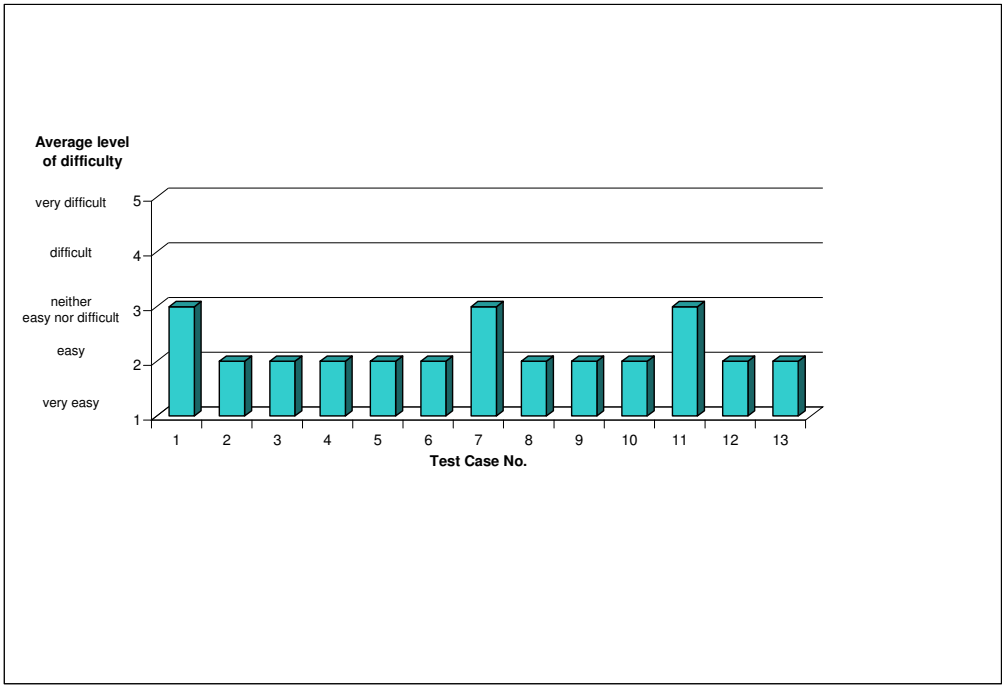



**Figure 91: Subjective viewpoints on the difficulty of test cases**

On average, the participants found 10 out of 13 tasks to be "*easy"* to do whereas the rest of the tasks were "*neither easy nor difficult"*. The questionnaires (see Doc. No. 7) addressing the specific usability features also indicated general opinion to be positive. On the contrary, the participants made comments that were not very specific. Instead, they highlighted only those usability problems that were most prevalent in their minds.

# 5 Test Limitations

The usability test has a number of limitations that need to be mentioned in order to determine the reliability and accuracy of test results. Firstly, the number of test participants used is insufficient. The test participants came from a small pool of specialized team and only a few of them were available at the time of the evaluation. The consequence is that a larger percentage of idiosyncratic behavior of participants could have been mistaken as the test data. In other words, the test results particularly the quantitative data could be skewed and unreliable. Secondly, the number of usability tests carried out are also insufficient. This again is due to a limited number of field visits constrained by busy schedule of fieldworkers. The number of test participants should be about 7 or 8 and usability testing carried out over a period of 3 or 4 days in order to obtain reasonably accurate test results.

Thirdly, a one-person team using a notepad is most likely to miss out many of the usability problems. Also, the test conductor does not have a human factors expertise. This lack of expertise and experience in a usability testing further reduced the number of usability problems found. Field testing is intended to provide realistic usage of the application but a lack of proper testing equipments like digital recorders to capture user behavior calls for concern. Ideally, a second person should assist the primary tester with a video recording of the tests and the data taken back to the laboratory for detailed analysis.

And finally, the test participants were unfamiliar with usability testing and they couldn't differentiate between a prototype and complete software. It is particularly difficult for them to evaluate the prototype or compare it to other similar systems because they have not used such systems before. Consequently, they are easily influenced by the tester and unable to provide any clear suggestions on how to improve the prototype (although they certainly like to use the application). Hence, it is felt that test participants should have been give an opportunity to test other mobile

applications of similar nature in order to make more informed assessment of the prototype.

# 6 Discussion of test results and Recommendations

This section attempts to explain the usability test results in Section 4, but also to provide suggestions and possible solutions for the usability problems. Beginning with the performance data (see Figure 2), the time taken to complete the tasks using the prototype is generally longer than the paper sourcing forms. A number of reasons for this outcome are proposed. Firstly, the test participants have significantly more experience using the paper templates than the prototype. Putting two hours long training aside, the test participants have not used a PDA before. Naturally, their proficiency will increase if they are given more time to experiment with the device. In order to obtain a fair comparison, they should have used the prototype to do the data sourcing for a few months and measure proficiency levels thereafter. Secondly, there are tasks that require the user to type in the data rather than selecting from a predefined list. These tasks take significantly longer to complete than their paper counterparts. For a foreseeable future, typing in the data using a SIP on a PDA (or any mobile device for that matter) will be slower than physically writing down on a paper document. It is difficult to explain the cause of usability problems by simply looking at the time it takes to complete them. One also needs to look at the nature of the task, the kind of errors and help sought by the test participants. Some usability professionals referred to this approach as the "*triangulation*" method (see Figure 4 below).

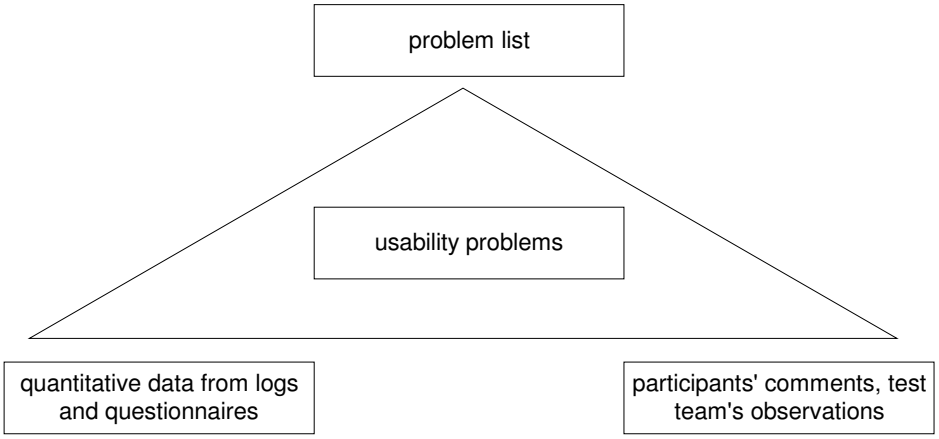



**Figure 92: "Triangulation" – using multiple sources of data to find usability problems (Source: Dumas J *et al*., 1999b)**

The number of usability problems found during the usability test is relatively low. This could mean a few things. First, the usability testing was not very successful possibly due to some of the limitations mentioned in Section 5. Or, the prototype may be well designed; hence there were not many usability shortcomings. It is most likely that it is contributed by a combination of these factors. Nevertheless, it is important to explain the cause of usability problems. Some of these usability problems (presented in Table 2) are explained using the Triangulation Method. In addition, possible solutions and recommendations to solve these problems are outlined.

**Problem 1: SIP conceals a textbox**

When the user clicks on the textbox at the bottom of the screen, the SIP is activated by a textbox event-listener. But the SIP appears on top of the textbox, concealing it completely (see Figure 5). The user is then required to scroll down to see the textbox again. This is rather a trivial problem but it occurs frequently with the pages that use vertical scrollbars. The test participants complained about it when they first use the prototype.

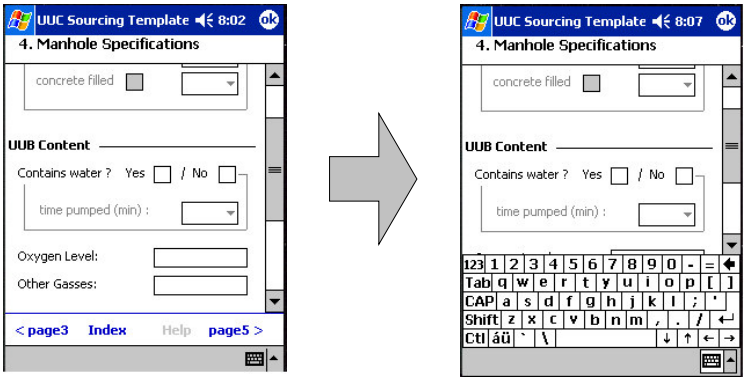



**Figure 93: SIP concealing the textfields**

The solution to this problem is fairly simple. The system must detect if the textbox that user clicked is placed or has moved (due to scrolling) to the bottom section of the page. If so, the page must shift up by a SIP length when the SIP is activated.

**Problem 2: Scrollbar swings off on Manhole Specification Page**

When the user tries to scroll a page by dragging the scrollbar he sometimes misses some of the data containers. This is because the distance over which the scrollbar moves is relatively small and scrollbar sensitivity is directly proportional to the length of the page that it scrolls. This was the case with the Manhole Specification Page which is longer than any other page. In other words, if the user drags the same distance on this page it would scroll a larger section of the page than its shorter counterpart. This inconsistency creates confusion for the users.

The scope of this problem is global (as opposed to local problem). Hence, the solution to the problem must be obtained from looking at other areas of the prototype. Ideally, one would leave out the scrollbars altogether. But if the application contains many data containers, it would require a few dozen screens without the scrollbar. The navigation becomes difficult in order to find the right data container. One could put these screens in a predefined sequence but this is

inflexible because the data entry personnel do not necessarily enter data in a particular sequence. By using scrollbar the number of screens required is significantly reduced. The Manhole Specification Page is longer than other pages because of a relatively large number of data containers dedicated to the manhole specification. As mentioned in the design phase, the purpose is to enable the user to navigate easier by placing together information that is closely related. It is recommended that the user use a navigation button for "small-steps" scrolling so that he can see the page by small section at a time. But for a quick scroll from top to the bottom of the page, dragging the scrollbar using the stylus is recommended. These techniques are simple to do and could be easily adopted by the users.

**Problem 3: Delay with background colour changes on Routeface panel on Ducts Location Page.**

There is a slight delay with the change of duct colour when the user taps the ducts extremely quickly. At such speed, the visual feedback mechanism failed to do what it was designed for. This problem is expected. Unfortunately, there is no clear solution to this problem. One would find this problem even with fastest desktop computers. The user can press a number of buttons and control items much quicker and repetitively using the stylus than a mouse or touchpad. The rate of user interaction is generally higher with the direct manipulation style of interaction than an indirect manipulation technique. The user is recommended to click at a quick but reasonable speed which is more than enough for the speed required for the placing ducts onto the routeface.

An alternative solution is to use interactive high quality graphics components in the place of standard visual component such as buttons, frames and labels etc. Importantly, it is necessary to use GUI components that are lightweight considering resource constraints of a PDA. A Scalable Vector Graphic Basic (SVGB) from W3C (2003) is designed to display vector based 2D graphics on a PDA. This XML-based language is lightweight, data-driven, and interactive –

allows a rich set of event handlers such as *onmouseover* and *onclick* to be assigned to any SVG graphical object. Unfortunately, the amount of delay resulting from clicking the SVGB objects with event-listeners is found to be greater than a standard eVB component. The former, however, is more suitable for maintenance and reuse of software which of course is the primary objective of object-oriented design. Once graphical templates are created using SVGB objects, they can easily be imported to other mobile applications.

In order to view the SVG files from the eVB application, one needs to write a separate program that will interpret Document Object Model (DOM) for that SVG standard. Alternatively, off-the-shelf reusable code modules in the form of ActiveX components for viewing (or editing) SVG files are available from the third-party vendors (e.g. eSVG, 2003 and PocketSVG, 2003).

Most of these usability problems are rather trivial. There were no severe usability problems that would require redesign of the prototype. Most of these usability problems can be resolved by minor modification to the existing prototype. The performance data metrics suggest that the intended users are able to use the prototype quite effectively. They are able to enter data not only accurately but efficiently as well. Their performance could be much greater if they used the prototype for the same period of time for which they have been using paper sourcing forms. Interestingly, the test participants make very few mistakes. Despite being new to the technology, the participants appeared to be confident and comfortable during the usability testing. Most promisingly, they look forward to use the prototype for their data collection work.

The subjective views of the test participants documented in the post-test questionnaires suggest that users are satisfied with the prototype. The user comments on the user interface screenshots confirmed this conclusion. On the contrary, they would like to see “more user-friendly” prototype but they failed to give any specific details. For example, they couldn’t come up with anything that they dislike about the prototype. Now that they have seen how much a usable software

system could make a difference, they wanted to make it “easier” without knowing what they really want. Unfortunately, this is one of the dangers of relying on the users to decide on what makes software usable.

# 7 Conclusion

The usability of field data collection prototype was evaluated using an empirical field test with the UUC data sourcing personnel. Field test was carried out at the actual data sourcing site where the test participants were given the prototype to do their work. The time taken to do the data sourcing tasks using the prototype was recorded and then compared with that of paper sourcing forms. The participants were then interviewed and asked to complete questionnaires that probe their subjective views on usability of the prototype. The number of usability problems discovered by the usability test is considered to be relatively low but nevertheless corresponding recommendations were made to correct these problems.

Overall, the qualitative test results are very encouraging. The majority of the fieldworkers found the prototype easy to use and keen to use the application in place of paper sourcing template that they are currently using. However, the accuracy of quantitative data is questionable due to a number of shortcomings in usability testing. Firstly, the number of test participants and the frequency of usability testing are insufficient. It is felt that the usability test could have been executed far better by increasing the number of test participants and frequency of tests over a longer period of time. Secondly, without proper data logging equipments (video camera, data logging software etc.), a one-person team has little control over the test conditions in the field. This could be one of the reasons why there were very few usability problems uncovered by the usability test. Another concern is that the test conductor has a limited expertise in conducting a usability test. Given these conditions many usability problems may have gone undetected. And finally, it is felt that test participants were easily influenced by the test conductor and did not have enough exposure to other mobile applications to make impartial evaluation of the prototype.

Ideally, the number of test participants should be increased to about 7 or 8 people and the usability testing is to be carried out over a period of 3 or 4 days. The main

purpose of usability evaluation is to fine tune critical design features and it is crucial that redesign takes place over a number of iterations.