MoSGen
A Visual Programming Tool for Mobile Services

Ruchika Singh

Luleå University of Technology
Master Thesis, Continuation Courses
Computer Science and Engineering
Department of Computer Science and Electrical Engineering
Division of Information and Communication Technology
MASTERS THESIS

MoSGen:
A Visual Programming Tool for Mobile Services

RUCHIKA SINGH

Luleå University of Technology
Department of Computer Science and Electrical Engineering
To my parents and family for their patience and support …
ABSTRACT

With the growth of mobile industry, a wide range of mobile services have been developed. But most of them are not adopted in our daily lives because they do not fulfil the needs and requirements of common users and are difficult to use. Programming of mobile services is also very complex and requires specialised knowledge. A literature study was carried out to evaluate all the environments that facilitate visual programming. Based on this study a suitable visual programming tool for generating mobile services was required to be suggested. But none of them fully matched with our requirements. Therefore, in this research a visual programming tool has been developed which could be used by novice end users to generate mobile services according to their needs. The developed prototype is referred to as Mobile Service Generator (MoSGen) in this thesis.

MoSGen, allows creating and triggering multiple mobile services simultaneously. For the present research a location-based service type called as Buddy Tracking Service has been considered. Two different services have been developed as examples to show the working principle of MoSGen. These services can be visually programmed and triggered with ease by the end users.

The target mobile phone considered suitable for this research is Sony Ericsson’s P1i. MoSGen can be successfully installed and executed on P1i. For location based services Global Positioning System (GPS) technology has been used to provide position information of mobile users. Since the phone does not have an inbuilt GPS, a GPS application has been developed that could fetch data from an external GPS receiver via Bluetooth.

In short, the thesis discusses the application of mobile services and visual programming with a detailed overview of the development process and evaluation of MoSGen prototype.

Keywords: MoSGen, mobile services, visual programming, J2ME, Bluetooth, GPS, location based services, buddy tracking.
PREFACE

The study and research work presented in this thesis has been carried out from February to September, 2008 at the Division of Information and Communication Technology, Department of Computer Science and Electrical Engineering and Centre for Distance-Spanning Technology (CDT), Luleå University of Technology, Luleå, Sweden.

I would like to express sincere thanks to my supervisor Dr. Kåre Synnes, Assistant Professor, Department of Computer Science and Electrical Engineering at Luleå University of Technology for his invaluable guidance and support during the research.

I would like to thank Mr. Mikael Börjeson and Mr. Jan-Olov Johansson of CDT for providing the resources required for this research. I would also like to thank Ericsson AB, Luleå for providing the project resources for research.

I would like to thank Mr. Ramin Karim, Division of Operation and Maintenance Engineering, LTU for fruitful discussions on Mobile Networking, Bluetooth and GPS technologies.

I would also like to thank all those who participated in the evaluation of MoSGen prototype. Their feedbacks were extremely important and useful.

I am indebted to my parents for always believing in me. My special thanks to my husband, Saurabh, for his support and useful suggestions during my research work.

Ruchika Singh
August, 2008,
Luleå, Sweden
# TABLE OF CONTENTS

*PREFACE* ...................................................................................................................... v  
*ABSTRACT* ..................................................................................................................... v  
*LIST OF ACRONYMS* ................................................................................................... xi

1. Introduction ............................................................................................................. 1  
   1.1. Background and Problem Description ........................................................... 1  
   1.2. State-of-the-Art .............................................................................................. 3  
   1.2.1 Evaluation of Visual Programming Tools ............................................. 13  
   1.3. Research Purpose ......................................................................................... 14  
   1.4. Research Question ....................................................................................... 14  
   1.5. Scope and Limitations .................................................................................. 14  
   1.6. Structure of the Thesis ................................................................................. 14  

2. Theoretical Framework .......................................................................................... 15  
   2.1 Java 2 Micro Edition (J2ME) ....................................................................... 15  
   2.2 Global Positioning System (GPS) ................................................................ 17  
   2.3 Bluetooth ...................................................................................................... 17  
   2.4 Mobile Services ........................................................................................... 18  

3. Software Development .......................................................................................... 21  
   3.1. Methodology ................................................................................................ 21  
   3.2. Software Requirements Specification ........................................................ 21  
   3.2.1. Functional Requirements ..................................................................... 21  
   3.2.2. Non-Functional Requirements ............................................................. 22  
   3.3. Software Design ........................................................................................... 24  
   3.3.1 System Architecture ............................................................................. 24  
   3.3.2 Design Considerations ......................................................................... 25  
   3.3.3 Data Model ........................................................................................... 26  
   3.3.4 User Interface and Interactions ............................................................ 27  
   3.4. Implementation ............................................................................................ 38  
   3.5. Software Validation ..................................................................................... 39  

4. Software Evaluation .............................................................................................. 41  
   4.1 Approach ...................................................................................................... 41  
   4.2 Results .......................................................................................................... 42  

5. Discussion and Conclusions .................................................................................. 43  
   5.1 Discussion .................................................................................................... 43  
   5.2 Conclusions .................................................................................................. 44  
   5.3 Scope For Further Research ......................................................................... 44  

*REFERENCES* ............................................................................................................. 47
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>AWT</td>
<td>Abstract Window Toolkit</td>
</tr>
<tr>
<td>BACCII</td>
<td>Ben A. Calloni Code for Information Interchange</td>
</tr>
<tr>
<td>BTSPP</td>
<td>Bluetooth Serial Port Profile</td>
</tr>
<tr>
<td>CDC</td>
<td>Connected Device Configuration</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
</tr>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rates for GSM Evolution</td>
</tr>
<tr>
<td>FLINT</td>
<td>FLowchart INTerpreter</td>
</tr>
<tr>
<td>FP</td>
<td>Foundation Profile</td>
</tr>
<tr>
<td>GPGGA</td>
<td>Global Positioning System Fix Data</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile</td>
</tr>
<tr>
<td>HTC</td>
<td>High Tech Computer</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Enterprise Edition</td>
</tr>
<tr>
<td>J2ME</td>
<td>Java 2 Micro Edition</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Standard Edition</td>
</tr>
<tr>
<td>JABWT</td>
<td>Java APIs for Bluetooth Wireless Technology</td>
</tr>
<tr>
<td>JCP</td>
<td>Java Community Process</td>
</tr>
<tr>
<td>JSR</td>
<td>Java Specification Requests</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Base Connectivity</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LBS</td>
<td>Location-based Services</td>
</tr>
<tr>
<td>MIDI</td>
<td>Musical Instrument Digital Interface</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Message Service</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MoSGen</td>
<td>Mobile Service Generator</td>
</tr>
<tr>
<td>NMEA</td>
<td>National Marine Electronics Association</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>PBP</td>
<td>Personal Basis Profile</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PDAP</td>
<td>Personal Digital Assistant Profile</td>
</tr>
<tr>
<td>PP</td>
<td>Personal Profile</td>
</tr>
<tr>
<td>RAPTOR</td>
<td>Rapid Algorithmic Prototyping Tool for Ordered Reasoning</td>
</tr>
<tr>
<td>RFCOMM</td>
<td>Radio Frequency Communication</td>
</tr>
<tr>
<td>SIVIL</td>
<td>Simple Visual Language</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UTC</td>
<td>Universal Time Coordinate</td>
</tr>
<tr>
<td>VAP</td>
<td>Visual Agent Programming</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>WCDMA</td>
<td>Wideband Code Division Multiple Access</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless LAN</td>
</tr>
</tbody>
</table>
Chapter 1

1. Introduction

This chapter presents the underlying background and problem areas of the research study. It discusses state-of-the-art concerning the different visual programming tools existing for novice end users. It also discusses the research purpose, research question and the scope of the research. Lastly, the structure of the thesis is outlined.

1.1. Background and Problem Description

The mobile telecommunications industry is one the fastest growing industry in the world. It provides a wide range of services to use, from delivery of voice over phones to the benefits of new content services such as Internet, email, messaging services, java games and location based services. Due to the applicability of these different services in our day to day life, mobiles have become a basic amenity rather than a status symbol. The growing availability of memory resources at mobile nodes and wireless communication technologies motivate the development of larger and larger set of mobile services.

Even though there is a huge number of mobile services commercially available in the market, but surveys have shown that only basic mobile services for example SMS, ring tones, icons and logos have been adopted or at least been tried by a majority of users (see Carlsson et al., 2005). The more advanced services have not yet found their ways into the everyday lives of consumers as they do not fulfil the needs and requirements of the users (see Kargin and Basoglu, 2007). Due to lack of expertise many users find the current services difficult to use and hesitate to trust them. Often they are unaware of existing services (see Pittet, 2004). It has been realised that the mobile services would be effective and widely adopted if they are simple to find, simple to develop, simple to trust, simple to use and appropriate to the needs and requirements of individuals (see Bartolomeo et al., 2006).

Nowadays mobile devices (e.g., mobile phone, Personal Digital Assistant (PDA)) are widely used by people living in different economic strata and having different professional background. They might be professionals in specific application domains but they might not be experts in programming. We will use the word ‘novice’ for such people who have no programming experience but are genuine users of computer and mobile phones. As already highlighted, one the critical problems of current mobile services is that they are unable to meet the needs and expectations of most of the users. If it becomes possible for the novice mobile users to themselves develop their own services, this could highly increase their trust and flexibility to use mobile services according to their own needs and desires. But the design and development of mobile services itself is complex, time consuming and requires a certain level of programming skills. The knowledge of server implementations, database and networking are usually required in order to produce a reasonable mobile service.
Computer professionals can certainly find their way out due to their technical skills and specialized knowledge about programming. But the majority of mobile users are novice and learning programming is not easy for them. It takes time to get familiar with programming and then gaining further knowledge to develop advanced programming skills. Therefore advanced applications like mobile services are extremely difficult to be designed and developed by the novice users. This restrict them to the current set of services, many of which they find complex and difficult to use and might not satisfy individual requirements and expectations.

The above limitations of novice users can be overcome by providing a visual programming tool that enables development of own mobile services by performing visual programming. One of the most-valued advantages of visual programming tools is that one does not require programming experience to learn and perform visual programming. It would certainly make a good impact if they are easily able to create a wide range of mobile services on an “as-needed” basis. This could perhaps motivate the users to adopt more mobile services into their everyday lives. This also increases their general awareness of the current mobile services. Even professional programmers would certainly find this environment as a more convenient, fast, and friendly way of mobile computing. The capability of the tool to support a wide range of users increases its applicability and demand.

The use of visual expressions in programming is called visual programming, where the program specification is carried out by spatial manipulation of visual elements on the display screen. Examples of visual expressions include icons, forms and graphs. Visual programming provides direct interaction between a person and machine which makes it easier to create programs (Hirakawa and Ichikawa, 1990). However, visual programming systems also have certain criticisms in terms of high memory consumption to meet their dynamic needs and wasteful usage of screen-space by visual expressions such as icons (see Brad, 1983). Both memory and screen-space are more limited in mobile phones, but we can argue that the current mobile phones already have sufficient memory to support so many advanced features like games, touch screen, internet access, digital cameras, music players, Global Positioning System (GPS) and customised software. As the mobile industry will grow further, the availability of memory for these devices will also increase. So it is believed that from time to time, there will be a lot of mobile phones available that could meet the memory requirements for the interactive and smooth execution of visual programs. For the case of screen space usage, visual expressions are more useful than just spatial reasoning because they enable us to define high level of operations. Other than this iconic interfaces have the advantage of requiring a user to recognise and point rather than remember and type as with text-based languages (Smith et al., 1982). Hence, if novice users are able to develop simple as well as advanced applications just by using set of visual expressions then the amount of space they occupy is worth their utility. Nevertheless, the screen-space utilisation could be improved through scrolling and providing higher abstraction.
Visual programming is one of the successful approaches of software development. There are numerous computer-based visual programming tools which have been developed by programmers and researchers over the years. But to date no programming tool exists for mobile phones or PDAs. An evaluation of different visual programming tools is discussed in the next section.

1.2. State-of-the-Art

A review of the different computer based visual programming tools that have been developed primarily to support novice users is presented in this section with an aim to adapt some of their features into the proposed visual programming tool.

BACCII (Ben A. Calloni Code for Information Interchange)

Visual programming tools where primarily icons are used to perform visual programming are called as iconic programming systems (see Ichikawa and Hirakawa, 1990). The BACCII is a Microsoft Windows based iconic programming tool that is designed to teach procedural programming to novice programmers (see Calloni and Bagert, 1994; Calloni and Bagert, 1995; Calloni and Bagert, 1997; Calloni et al., 1997). This is achieved by learning algorithm development. BACCII provides a user-friendly and syntax-directed environment where the programs are built as flowchart by inserting icons. The icons represent the syntactic structures of procedural programming for example variables, arrays, input-output, if-else, while-do and for-loop. When the flowchart is complete, the users can view the syntactically correct generated source code in American Standard Code for Information Interchange (ASCII) format for PASCAL, C, FORTRAN, or BASIC. Figure 1.1 shows the interface of BACCII while implementing an algorithm.

![Figure 1.1: BACCII coding screen](image-url)
Over the last few years, object-oriented constructs have been added to BACCII. The resulting environment is called BACCII++ which allows implementation of object-oriented programs by using iconic-flowchart. With BACCII++, the code can be translated into C++ programming language. Generating syntactically correct source codes is a good way of exposing novice programmers to high-level languages. But the generated source codes need to be separately compiled and executed to check the correctness of program’s logic.

**FLINT (FLowchart INTerpreter)**

FLINT is designed to teach novice programmers how to design and write programs (see Crews and Ziegler, 1998; Ziegler and Crews, 1999). It provides an environment to develop problem solving, design and logical thinking strategies. The tool supports several activities such as designing top-down structure charts, implementing algorithms using flowcharts, testing and debugging the implemented flowchart using the interpreter. It provides an iconic interface for developing flowcharts. The structure chart is a division of a problem into its major steps. Before constructing flowchart for a problem, the user should design a structure chart for better and deeper understanding of the problem. Each of the major steps in the structure chart can be implemented by a separate flowchart. Figure 1.2 shows the interface of FLINT while flowcharting one of the major steps from the structure chart. FLINT eliminates any focus on syntactical details and enables conceptual learning of introductory programming. It provides a very suitable programming environment for beginners but has a very limited scope of programming functionality.

![Figure 1.2: Screen Shot of FLINT](image)
SIVIL (SImple VIual Language)

SIVIL is designed to teach novice programmers the concepts of programming and develop logical thinking (see Masterson and Meyer, 2001). It uses icons and pictures as visual elements to symbolize all of its structures. The programs are created by connecting the visual elements. These programs can be compiled, tested and debugged, and viewed in a graphically rich environment. The interface has the capability to include comments close to the appropriate icon or picture. The brief comments can be displayed in the coding window and the detailed descriptions can be hidden behind the code. This helps students to realize the importance of adding comments to their programs. The tool aims to make programming easier for novices. Figure 1.3 shows the user interface of SIVIL modelling a numeric guessing game. SIVIL provides a natural and practical guide towards helping students understand and enjoy programming but starting to learn languages like Java or C++ after learning only SIVIL would be an entirely different experience.

Figure 1.3: Screen shot of SIVIL
RAPTOR (Rapid Algorithmic Prototyping Tool for Ordered Reasoning)

The RAPTOR is designed to teach algorithm development to novice programmers. It provides a flowchart-based environment for algorithm development (see Carlisle et al., 2005; Carlisle, et al., 2004; Giordano and Carlisle, 2006). It encourages improving problem solving skills while reducing focus on syntactical details. RAPTOR is equipped with more sophisticated functions, procedures, data structures and graphics libraries to create more interesting programs. It also allows inclusions of comments, testing and debugging their flowchart within the environment. Figure 1.4 shows the user interface of RAPTOR while developing a flowchart-based algorithm. It provides a very practical way of algorithmic programming, but does not completely remove focus on syntax. It enforces syntactical Java-like entries into the flowcharting symbols due to which a novice programmer would need enough guidance and instructions to learn creating programs.

Figure 1.4: RAPOR Screen Shot
B#

B# is designed to help novice programmers develop problem solving skills. It is an attempt to solve the difficulties experienced by the students in introductory programming courses. With B#, programs can be created by using iconic flowcharts (see Greyling et al., 2006). As the flowchart icons are drawn on the screen, the equivalent source code in Pascal is also generated which can be simultaneously viewed by the user. Figure 1.5 shows the user interface of B#. The flowchart can be executed at any time, allowing the users to test and debug their flowchart. While viewing the generated source code simultaneously, the users gain familiarity and knowledge in textual programming. It is an effective tool for introducing the novice students to programming but has very limited scope of programming functionality.

![Figure 1.5: B# Screen](image)
Visual Programming using Flowchart

This tool is designed to assist the novice programmers learn algorithmic design and development (see Charntaweekhun and Wangsiripitak, 2006 for details). The algorithms are developed by constructing iconic-flowcharts. While the flowchart is built, the translation module of the tool translates the flowchart into C source code. During the time the programmer compiles the flowchart, the C compiler and interpreter embedded within the tool compiles the C source code to create an EXE file. This file is then invoked to print the output onto the display screen of the visual editor. In case of any missing inputs, the tool acts as an interface to receive and pass the required inputs to the EXE program for execution. Figure 1.6 shows the user interface of the tool during the construction of a flowchart. The tool is practical and useful for teaching computer programming to beginners, but the existence of the C compiler and linker is usually hidden from users.

Figure 1.6: Visual Programming Using Flowchart
Computer-Mediated Collaborative Visual Learning System

The Computer-Mediated Collaborative Visual Learning System is designed for nursing science undergraduates to enable a visual representation of the nursing care plans (see Paik and Ham, 2006 for details). To provide quality patient care, they need to generate a quality nursing care plan which consists of a number of sequential steps. The nurses learn to do so by studying the existing cases and also by observing what other nurses do. These tasks can be efficiently done and in much less time by using this centralised visual learning system. The visual representation is based on MindMap which is an alternative way to organize information about patients and enhances learning experiences of nursing students. The system also keeps track of evidences which are accumulated from previous nursing care plans generated. This promotes evidence-based nursing care to enable generation of more accurate and reliable nursing care plans. Figure 1.7 shows visual representation of the nursing care plan for one of the patients.

Figure 1.7: MindMap for Patient HS
VAP (Visual Agent Programming)

The VAP is designed to help novice users to script and deploy life-like characters, also called as animated agents. It provides an easy to use interface for programming the agents. The agents are programmed by creating or selecting commands for them individually. While the user creates a new command or modifies an existing command for any of the agents, the program code for the command is automatically generated (see Khowaja and Guha, 2007). VAP supports both JavaScript and VBScript and the user can select to generate the program code in any of these scripting languages. VAP not only allows programming of agents graphically but also enables users to program these agents manually by referring the generated program script. Figure 1.8 shows the user interface of the system while adding commands for one of the agents. Users can select the agents by simple mouse click. VAP is available online for free. It provides other useful features like constant guidance by giving detailed error descriptions with solutions and also provides automatic generation of comments with every single command. However, users cannot create their own comments.

Figure 1.8: Screen Shot of VAP
**HomeCI**

This tool is designed to help novice healthcare professionals in the management and deployment of technology within the home environment. The purpose is to support independent living for elderly and disabled people. This tool provides an environment to monitor the operation of technology within the person’s home by establishing event-based rules. These rules are patient-specific and are created by the operator using visual expressions. Each rule represents a chain of sequences of events. If the sequence is broken, it is considered as abnormal behaviour and would require attention from the healthcare professionals. Like that, a number of rules/conditions can be established to provide a complete home-based support (see Nugent, et al., 2007). Figure 1.9 shows the editor’s user interface while creating one of the healthcare rules. The tool provides an easy and practical way of managing and monitoring technology within the home environment. However, it can program only one rule at a time and the connectors (for example lines, arrows) available to connect the icons are also limited.

![Figure 1.9: Screen Shot of HomeCI visual editor](image)

**Progranimate**

Progranimate is designed to help novice programmers learn algorithmic programming (see Scott, et al., 2008). The algorithm is built by constructing flowcharts using icons as its components and structures. The flowchart can be executed any time, allowing
users to test and debug their flowchart. As the flowchart is constructed, the equivalent source code is also generated in a selection of languages that currently includes Java and two versions of Visual Basic. By observing how the flowchart design maps to the code solution, gives exposure to the syntax and logics of text-based languages. Progranimate is integrated with the web and can be launched online after installing java runtime environments, Java 1.5 or later. Figure 1.10 shows the interface of the tool while constructing an algorithm. It provides a rich error feedback environment and real-time guidance but has a very limited scope of programming functionality.

**Figure 1.10: The Progranimate Environment**
1.2.1 Evaluation of Visual Programming Tools

As discussed above, there are several visual programming tools which have been developed primarily for novice users. However, all the discussed tools are computer based and are used for specific purpose. But all of these tools possess features which might be of use for the development of the proposed tool. Table 1.1 shows the features required for the proposed tool and their availability in the existing tools.

Table 1.1: Evaluation of the availability of the desired features in the existing tools

<table>
<thead>
<tr>
<th>Features</th>
<th>Tools</th>
<th>Point &amp; Click</th>
<th>Connectivity between Icons</th>
<th>Complete Iconic representation (no textual feeding)</th>
<th>Rule Based Iconic Flow</th>
<th>Real-time Guidance</th>
<th>Error Description</th>
<th>Multiple Problems Execution</th>
<th>Service Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BACCII</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>FLINT</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>SIVIL</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>RAPTOR</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>B#</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>VPF*</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>CMCVLS**</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>VAP</td>
<td>Absent</td>
<td>Absent</td>
<td>Absent</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
<tr>
<td></td>
<td>HomeCI</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
</tr>
<tr>
<td></td>
<td>Progranimate</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Absent</td>
<td>Absent</td>
</tr>
</tbody>
</table>

* VPF: Visual Programming using Flowchart  
** CMCVLS: Computer-Mediated Collaborative Visual Learning System

All of these tools have some of the features which could be adapted in the proposed tool but none of the tools have all the required features. Therefore, instead of adapting a particular tool, the good features present in all these tools are adapted while designing the proposed tool, which we name as **MoSGen (Mobile Service Generator)**.

Among all the tools reviewed, HomeCI has most of the features required for the proposed tool. Apart from the other useful features, it also has a feature of service generation and complete iconic representation which is absent in all the other tools discussed. Therefore, the user interface of MoSGen is quite similar to HomeCI.

Based on the literature review, the purpose of the research and the research questions are described in the coming sections.
1.3. **Research Purpose**

The purpose of the research is to study and develop a suitable simple visual programming tool to help novice end-users develop mobile services.

1.4. **Research Question**

The research question that needs to be answered is:

1. How to design and develop an easy to use visual programming tool for simple mobile services?

1.5. **Scope and Limitations**

For the development of a visual programming tool, the current research has the following limitations:

The proposed visual programming tool has the capability to support programming of multiple mobile services at the same time. This research has a limitation of triggering up to two different kinds of services. But further programming can be done to enable more number of services with increasing complexity.

Presently the user interface of the tool has limited icons which are adequate for programming of up to two different simple mobile services. However, with increasing complexity of flow diagrams, more complex icons and connectors can be added into this visual programming tool.

1.6. **Structure of the Thesis**

The thesis consists of five chapters. Chapter 1 introduces the research, briefly giving the background and problem areas of the research study. The chapter also discusses state-of-the-art concerning the different visual programming tools existing for novice end users. Finally, it outlines the research purpose, research question and the scope of the research.

Chapter 2 presents the underlying concepts of technologies used in this research for the development of MoSGen.

Chapter 3 presents the various phases in the process of writing the prototype tool, MoSGen.

Chapter 4 presents user evaluation of MoSGen where the feedbacks on the overall behaviour and usability of the system have been analysed.

Chapter 5 presents the valuation of MoSGen and conclusions. Few suggestions for further research are also discussed.
Chapter 2

2. Theoretical Framework

This chapter presents the underlying concepts of technologies used in this research for the development of MoSGen.

2.1 Java 2 Micro Edition (J2ME)

J2ME is the java runtime environment for consumer and embedded devices such as PDAs, mobile phones and other small devices. Like its counterparts, that is, enterprise (Java 2 Enterprise Edition (J2EE)), desktop (Java 2 Standard Edition (J2SE)) and smart card (Java Card), the J2ME platform defines a set of standard Java Application Programming Interfaces (APIs). An overview of J2ME components and how it is related to other Java 2 Platform technologies is presented in Figure 2.1.

![Figure 2.1: The Java Platform](adopted from Sun Microsystems-Java ME, 2008)

J2ME is divided into configurations, profiles and optional APIs (Piroumian, 2002; Li and Knudsen, 2005). Mobile devices such as cell phones, pagers, PDAs etc., are diverse in form, features and functionalities. Therefore, configurations APIs have been introduced to capture the essential capabilities of each kind of device. A configuration is a set of horizontal APIs that define a minimum set of features for a group of devices having similar needs of processing power and memory. It specifies three basic elements:

- the Java programming language features
- the Java machine features, and
- the basic Java libraries with supported APIs

Currently, there are two standard J2ME configurations available. They are Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). The CLDC is specifically designed for devices having limited memory, processing power and graphical capabilities like mobile phones and PDAs. Whereas, the CDC is targeted towards larger devices with more capacity and having intermittent connection to network, such as set-top boxes, internet televisions, home appliances, and car navigation systems.

A profile is a set of vertical APIs that is layered on top of a specific configuration to provide domain-specific capabilities for a particular family of devices. It provides additional APIs for networking, creating user interfaces and handling user input. The Mobile Information Device Profile (MIDP) is a CLDC based profile for running applications on devices like mobile phones with small screen, wireless Hypertext Transfer Protocol (HTTP) connectivity and limited memory. Another CLDC based profile is Personal Digital Assistant Profile (PDAP) which is currently under development. It is a superset of MIDP and includes additional classes and features to target more powerful handheld devices. In terms of CDC based profiles, the Foundation Profile (FP) extends the CDC with additional J2SE classes, the Personal Basis Profile (PBP) extends the FP with Abstract Window Toolkit (AWT) user interface classes and the Personal Profile (PP) extends the FP with applet and full AWT support. The applications are then built on top of the configuration and profile and can only use the class libraries provided by these two lower-level specifications. A J2ME implementation can contain only one configuration but multiple profiles can be based on the same configuration.

Optional packages define a set of APIs that could be used with J2ME configurations and profiles to create a rich software stack. An example is Bluetooth API which adds Bluetooth support to CLDC-based profiles. A specification list of current configurations, profiles and optional packages can be accessed at the official website of Java Community Process (JCP) (see Java Community Process, 2008).

J2ME has become the most popular application platform for mobile devices (Sun Microsystems - Java ME at a glance, 2008). It includes flexible user interfaces, robust security and inbuilt network protocols. J2ME is widely used and has been deployed on billions of devices for manufacturers like Nokia, Sony-Ericsson, Blackberry, Motorola and many more. A real advantage with J2ME development is that it accommodates a wide range of mobile devices. In terms of its configurations and profiles, MIDP has been widely adopted as the platform of choice for mobile applications. CLDC combined with MIDP provide a complete Java application environment for mobile phones, PDAs’ and other devices with similar capabilities. If the developer is familiar with Java then building a prototype application is relatively simple and fast with MIDP/CLDC. In this research too, J2ME (CLDC+MIDP) has been chosen as the development platform for MoSGen.
2.2 Global Positioning System (GPS)

The GPS is a satellite-based navigation system that provides positioning and timing information. It was initially developed by U.S. Department of Defense (DoD) for military use, but later it was made available to civilian users as well. GPS consists of a constellation of at least 24 operational satellites which are placed in the six orbital planes of the earth. Each satellite makes two complete rotations around the earth everyday. The orbits are so arranged that at any point of time, there are at least four satellites visible in the sky.

Information about time, user position, and therefore velocity can be determined by GPS. The GPS satellites continuously transmit signals which contain data on the satellites location and time that can be detected by anyone with a GPS receiver. At least three satellites are needed in order to obtain position information. The receiver compares its own time with the time sent by each of the satellites and then multiplies each time interval with the speed of light to obtain its distance from the satellite. The receiver then uses triangulation method to determine the user’s position in terms of latitude and longitude. In order to determine position in terms of altitude as well, a fourth satellite will be required for the purpose (Johnson et al., 1992).

GPS works 24 hours per day, anywhere in the world and under all weather conditions. The accuracy of GPS receivers depends on the number of satellites visible. In order to receive sufficient signal, it needs a clear view of sky which can be obstructed by tall buildings, hills and trees. Most of the GPS receivers have an accuracy of 12 – 15 meters (Manson, 1997).

The GPS time information determined by the receiver is related to the worldwide time standard, Universal Time Coordinate (UTC). Each satellite contains atomic clocks which are continuously monitored against the atomic clocks on earth, and the whole system is calibrated against UTC (Institue Mihailo Pupin, 2008).

Today GPS is used in a wide range of applications which include location, navigation, tracking, mapping and timing. For the implementation of mobile services on MoSGen, GPS has been used as a tracking system to record the location of a mobile phone at regular intervals.

2.3 Bluetooth

Bluetooth is a standard for short range, low power, low cost wireless communication that allows devices to connect and interact with each other. Today, there are a number of different types of devices which can utilize the Bluetooth technology. They include intelligent devices (mobile phones, PDAs, personal computers), data peripherals (mice, keyboards, joysticks, cameras, digital, pens, printers, Local Area Network (LAN) access points), audio peripherals (headsets, speakers, stereo receivers), and
embedded applications (automobile power locks, grocery store updates, industrial systems, Musical Instrument Digital Interface (MIDI) musical instruments).

The connectivity range of Bluetooth devices is about 10 meters and data is exchanged at a rate of 1Mbps (megabyte per second). The Bluetooth is designed to connect up to eight different devices. Some devices can connect to only one device at a time while others allow multiple Bluetooth connections on the same device for example mobile phones and PDAs (Axim Site, 2008).

The popularity of Bluetooth technology is increasing rapidly. It is more ubiquitous among those who want to create personal area networks (PANs). Each PAN is a dynamically created network built around an individual that enables devices such as mobile phones and PDAs to connect automatically and share data instantly.

Java-based applications can access Bluetooth through the Java APIs for Bluetooth Wireless Technology (JABWT), or Java Specification Requests (JSR)-82 (Sun Microsystems - Using the Java APIs for Bluetooth Wireless Technology, 2008). It is an optional package that can be used with any J2ME configuration and profile.

In the prototype implementation, Bluetooth application programming using JABWT has been done for the purpose of establishing connection with a GPS receiver in order to retrieve the GPS data, such that both the mobile phone and the GPS receiver are Bluetooth-enabled. The data retrieved from the GPS receiver is then decoded to display the phone location.

### 2.4 Mobile Services

With the advancement of mobile technology, a wide range of mobile services have been developed, ranging from traditional voice services to mobile multimedia services which contain voice, text, graphics and video features. According to Vesa (2005), mobile services can be divided into three broad categories, which are

- **Person-person messaging** – includes Short Message Service (SMS), Multimedia Messaging Service (MMS), email, push-to-talk and other methods of personal non-voice communication.

- **Content services** – includes SMS-based content services, so called “2G content services” (Wireless Application Protocol (WAP) and MMS-based services and downloadable applications), etc.

- **Mobile data services** – allows data transfer over digital mobile networks by using technologies such as Global System for Mobile (GSM), General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), Universal Mobile Telecommunications System (UMTS) and Wireless LAN (WLAN)/Wi-Fi.
Location-based services (LBS) are part of mobile data services which considers the current location of the users. These services can be used to obtain useful information for example a list of nearby shops or presence of mobile buddies in close proximity. LBS can be categorised into reactive and proactive services. In reactive services, the location data is delivered to the user only on request whereas in proactive services, the location data is instantly and automatically delivered whenever a location event occurs, as for example, when the target user enters or leaves a city, building or any other geographic area. Proactive LBSs are more difficult to implement than reactive ones because the position of the target users has to be tracked periodically for checking the occurrences of location events.

Proximity detection is one of the applications of location-based services. It is defined as the capability of LBS to detect when two or more mobile phones approach each other closer than a predefined distance (Kupper and Treu, 2006). In MoSGen, mechanism of proximity detection has been considered for the development of mobile services. This could be called as a buddy tracking service which gets triggered when the target buddies are nearby within the range of predefined proximity distance. With the target mobile phones connected to GPS receiver, they deliver the GPS coordinates (location) information to a central location server. This transfer of data is also referred to as position updating and the server periodically checks for the target users to come within the proximity distance. In MoSGen, this is typically implemented as a dating service between mobile friends. A scenario of buddy tracking service is shown in Figure 2.2, where the tracker wants to receive an alert when the target mobile users are within the distance of ‘C’.
One of the advantages of the proposed mechanism is reduction of the number of messages that are exchanged over the air-interface as far as possible, thereby saving the network bandwidth and monetary costs. However, the other applications of location-based services, primarily proactive detection services are mobile gaming, instant messaging, fleet management and child tracking.

With the increasing demand of wireless technologies, a wide number of data services have been developed till date. A majority of data services are location-based which provide one of the most powerful ways to personalize mobile services as they are based on location.
Chapter 3

3. Software Development

This chapter presents the various phases during the process of development of the prototype tool, MoSGen.

3.1. Methodology

Software engineering practice incorporates the use of various methodologies for software development. In the present research, the Waterfall approach (see Sommerville, 2006) has been used for developing MoSGen. In the Waterfall approach, the development process is split into several discrete phases. Each phase should be completed before proceeding to the next phase. The approach is easy to implement but has less flexibility as overlapping between phases cannot be done. However, this approach seems to be suitable for prototype development where the requirements of the software are well-defined and known at an early stage. The various phases of development of MoSGen using the Waterfall approach are discussed sequentially in the following sections of this chapter.

3.2. Software Requirements Specification

A Software Requirements Specification gives the complete detail of the behaviour and environment for the software system under development. As such, it can be categorised into functional and non-functional requirements. The functional requirements define specific functions that the system must perform. On the other hand, the non-functional requirements describe how the system must behave, thereby imposing constraints on the design of the system.

3.2.1. Functional Requirements

The functional requirements of MoSGen have been laid as a result of discussions with CDT department at the university. The functional requirements of the system are as follows:

- The intended purpose of the system should be to allow the development and then triggering of simple mobile services.
- The development of mobile services on the system should be accomplished by performing visual programming.
- The mobile services which are to be currently supported by the system should be context triggered, for example services which are triggered by knowing the geographic locations of users.
• The intended target audience of the software would be general mobile users without any programming knowledge.

3.2.2. Non-Functional Requirements

The non-functional requirements of MoSGen have been established so as to meet its functional requirements. The non-functional requirements of the system include the following:

**Performance:** The system should enable development and triggering of at least one mobile service.

**Software Platform:** There are lot of choices of technologies for developing applications for mobile platform. They include C, C++, Java, Python, C# and many more. The author is more familiar with Java and this was one of the reasons for opting J2ME as the development platform for MoSGen. The choice of J2ME profile and configuration include MIDP and CLDC. The other good reasons for deciding upon a suitable platform have been discussed in the previous chapter (Please refer Chapter 2, Section 1).

**Hardware Platform:** Deciding upon a suitable phone for MoSGen was one of the most important decisions taken during the research. There had been a few mobile handsets available in the department for experimental use. In order to run MoSGen application tool, it was mandatory that the target device should have support for software based on J2ME platform and should also have a touch screen so as to enable visual programming. The only resource available that could match with these requirements was *High Tech Computer (HTC) TyTN II* having Windows Mobile 6.0 operating system. Though it is not Java enabled by default, International Business Machines (IBM) Corporation provides a Java Virtual Machine (JVM) that can run J2ME applications. One added advantage is that it has inbuilt GPS which simplifies the implementation of location based services for such PDAs. Experiments were carried out by students in other projects to see what type of MIDP applications can run on HTC TyTN II. While they were experimenting with a simple location based service application, they couldn’t retrieve and display the GPS data on the phone. So the idea of adopting HTC as the target device for this tool was rejected as this could obstruct successful running of location based service applications written in Java. It was then decided to find a phone or PDA that would be suitable for this research.

The target mobile phone must have the following hardware features:

- Should be Java enabled so that it could directly run J2ME applications without any problems. Further it should have support for both CDC and CLDC environments. Though CLDC environment is sufficient for this research, but in future if more advanced features need to be added then CDC would be a more preferable environment.
- Should have a good display size and touch screen to enable visual programming.
• Symbian operating system is suitable to create advanced Java-based applications. Also, it provides extensive connectivity options including GSM, GPRS, Wideband Code Division Multiple Access (WCDMA), WiFi and Bluetooth.

• In-built GPS is preferred for fast development of location based services. But even if GPS is not present in the phone, using an external GPS receiver can solve the purpose.

After sorting the available phones, Sony Ericsson P1i turned out to be the most suitable phone for this research. It meets all the above requirements except that it lacks inbuilt GPS. This limitation was overcome by using an external Wayfinder Bluetooth GPS receiver. Since P1i supports Bluetooth API and the GPS receiver is Bluetooth-enabled, it is possible to write MIDP applications to create a Bluetooth connection with the receiver to retrieve and read GPS data. The receiver is quite affordable, small, and handy and serves as a good solution for the purpose. Figure 3.1 shows P1i phone with Wayfinder Bluetooth GPS receiver.

![Figure 3.1: Sony Ericsson P1i and Wayfinder Bluetooth GPS receiver](image)

It was then required to assess the number of such resources which would be essential for this research. At this stage it was important to decide about the actual service that was required to be implemented. Some LBS need information about the geographic location of only the device user itself for example services which display ads specific to the region where the user is travelling in. But some other services need to know the locations of multiple users. One of such types is buddy tracking service where each user wishes to be informed whenever one in the list of other users enters in a region of close proximity to the user. Proximity detection is one of the most interesting and attractive applications of location based services which computes and maintains social networks. Hence, we decided to proceed for implementing one of these service types. More detailed information about location based services has been mentioned in the previous chapter (Please refer Chapter 2, Section 4). Two or more P1i and GPS receivers would be required for testing proximity between two or more mobile users. In this research two P1i and two GPS receivers have been used due to provision of limited resources for prototypes.
3.3. Software Design

This section presents an architectural overview of the entire system, design considerations, a generic data model of the system and lastly, the flow of graphical user-interfaces of the software, also called Storyboarding.

3.3.1 System Architecture

The MoSGen system consists of the three main components:

- Client – P1i and Bluetooth GPS receiver constitute the client of the system.
- Server – resides in a dedicated computer connected to the internet
- Database – resides local to the web server

The system uses Apache Tomcat as the web server and MySQL as the database. Both Tomcat and MySQL are widely used and provides a simple but powerful development environment. Figure 3.2 shows the basic architecture of the system.

![Figure 3.2: MoSGen Software Architecture](image)

The working principle of each of the design components are explained in the following steps:

- Each of the GPS receivers calculate its position on receiving signals from at least three GPS satellites located high above the Earth. The GPS receivers need a clear view of sky for their best performance. An overview of GPS and its working mechanisms has been discussed in Chapter 2 (Please refer Chapter 2, Section 2 for details).
The MoSGen application running in P1i connects with a GPS receiver to read the location coordinates. The connection is established via Bluetooth and the receiver should not be beyond 10 meters range from the mobile phone (Please refer Chapter 2, Section 3 for details). The GPS receiver is handy and small in size that it could easily fit into pockets just as easily as a mobile phone. Hence, wherever the user goes, he can always take the GPS receiver along with his P1i phone.

The P1i client communicates with the server via internet. The server accesses the database using Java Database Base Connectivity (JDBC) driver.

Each P1i client sends the location information to the server. The server then stores the received coordinates into the database. For buddy tracking service the clients need to send GPS coordinates to the server at periodic intervals so that the database is always updated with their current positions.

If a P1i client invokes the buddy tracking service, it needs to send a list of target buddies or friends to the server. The server then checks if the database contains the coordinates of the target users. If present, the server computes distance between the client and all of its target users on the basis of their individual coordinates. As the database gets periodically refreshed with new coordinates, the computation of proximity distance is also performed at regular intervals.

If any of the computed distances are within a specific range of values, the client and the tracked user receive an SMS regarding their proximity.

If any user in the buddy list is not present in the database, the server performs no computation with respect to that user. The rest of the users in the list remain unaffected.

Thus we see that the web server plays an important role in keeping track of GPS coordinates of all the connected clients and simultaneously tracking proximity detection if such a service is requested by any of the connected clients.

### 3.3.2 Design Considerations

Before stepping into the actual implementation, certain aspects of the solution need to be considered. Each aspect reflects the goals the software is trying to achieve. Those aspects are:

**Usability:** The user interface of the software should be interactive and friendly so that it can be easily used by novice users.

**Extensibility:** There should be a possibility to add new components to the system without major modifications to the existing system.

**Modularity:** The system should consist of independent well-defined modules. These modules are developed and tested independently and then integrated to the system.
**Reusability:** The modular components of the system should be reusable as far as possible.

The extensibility, modularity and reusability features of design are achieved by adopting object-oriented software technique throughout the development process. The usability aspect of design is taken care of while presenting the user interface (see Section 3.3.4).

**3.3.3 Data Model**

The MoSGen architecture can be defined by a data model. A data model is a logical organization of data objects and relationships among them. The modular components or classes that serve as building blocks of the final system are identified and decided upon this model. Figure 3.3 shows the data model of the system.

*Figure 3.3: MoSGen Data Model*

The diagram shows three architectural components of MoSGen – the Client, the Server and the Database. Components like Cell ID, Bluetooth GPS and Draw Canvas comprises the modular components of the client.
3.3.4 User Interface and Interactions

The workflow of MoSGen prototype tool is divided into three sequential operations:

- Providing the Cell ID information – Mandatory to identify the tool user to the server
- Retrieving the phone GPS coordinates – Mandatory for location based services
- Creating and Triggering Mobile Services – The main purpose of the tool

The user interface design, interactions and working mechanism of MoSGen is presented in the following steps:

**Step 1:** The tool starts with a screen asking the user to manually provide the phone number information. Due to security reasons J2ME does not allow access to the cell contact id information. To overcome this limitation the user is asked itself to manually select the correct contact id for the phone.
Step 2: The user clicks on *OK* button and gets a list of contact names from the phone book where it selects its phone name. In this example we would consider that the tracker is “Ruchika”. Hence, the name *Ruchika* is selected from the list.

Step 3: The application confirms that the user has selected the correct option. If the user clicks the *OK* button, the phone number is sent to the server which stores the information in the database. Otherwise, the user could go back to again select from the list.
**Step 4:** Now, that the user has successfully completed providing the Cell ID information, he/she is presented with a screen which shows the GPS information of the user’s mobile phone. Initially, the values are null which indicates that the user has to perform manual configurations to get the GPS coordinates. If the user wants to create location based services then it is mandatory to get the GPS coordinates, otherwise the user could directly jump to visual programming screen by clicking the DrawTool button. Since the current prototype supports only location based services, the user has to initially get the GPS coordinates which can be obtained by switching ON the external GPS receiver and then clicking the Search button. This could also be interpreted as - if a user wants to keep track of any individual, then the system should make sure that the user could also be tracked by other such users.
**Step 5:** On clicking the *Search* button, the system searches and enlists the Bluetooth devices in range.

The user can then select his/her Bluetooth GPS receiver name in the list and click on the *Select* button to get the GPS coordinates. If in case it does not find the device in the list, the user could go back by clicking the *Back* button for another fresh search.

**Step 6:** When the mobile successfully finds and selects the GPS device, a screen appears which shows the GPS coordinates retrieved from the GPS device.
Here, the user must wait for a few seconds until the Status ConnectedQuality is more than zero or the Satellite value is at least 3. The GPS parameters are updated at every 10 seconds so as to keep track of the current position of the user. Every time the values are refreshed, new location values are automatically sent to the server, so that, the server keeps track of the position of the user at regular intervals.

**Step 7:** The user then clicks on the DrawTool button to switch to the screen where he/she can create services by performing visual programming. The methodology of visual programming is based on the concept of flowchart which gives a step-by-step solution of a problem. Flowcharting concept is quite popular today and known to most of the novice users. Also, from the literature study done in this research, it can be concluded as one of the most suitable and convenient methodology of visual programming for novice end users.

On MoSGen, the flowcharts are constructed by drawing predefined icons. These icons are situated in a single row on the top panel of the screen. The icons can be drawn only within the blue area of the screen which is considered as the drawing panel. A small portion of the drawing panel is reserved to display errors and confirmation messages that we would see later. The icons drawn on the panel can be easily dragged and moved around. It is also possible to resize the arrow icons by dragging their tips. The drawn icons can be deleted by dragging them to the rightmost bottom area of the screen represented by a trash icon. All the icons drawn on the panel can be removed at once on clicking More -> ClearScreen.
Step 8: To draw any icon, the user first clicks at the icon from the top panel and then clicks on *More ->Actions->Draw* and clicks anywhere in the drawing panel wherever the user wants to draw the icon.

![Image of drawing tool with icons and actions]

Step 9: The icons can be connected with each other by drawing horizontal arrows between them. Whenever any arrow intersects with any icon, an *Intersection* message pops up. Whenever any two icons are connected, the colour of the arrow turns to green and remains so until both the icons remain intersected with the arrow. So the green colour of any arrow confirms that icons are connected at its ends.

![Image of connected icons with intersection message]
Step 10: The two mobile services currently supported by MoSGen are Coffee and Movie services. Both these services are types of buddy tracking service (Please refer Chapter 2, Section 4). When the coffee or movie service is triggered, the system starts tracking if the user is in the proximity range of any of its friends. The two services have their own list of friends. If any friend comes in the proximity range, then an SMS is sent to both the user and his friend. In case of Coffee service the SMS contains a message saying that since the two friends are nearby, they can meet and have coffee. Whereas, in case of Movie service, the SMS contains a message saying that since the two friends are nearby, they can meet and go for a movie. The support for multiple services adds flexibility and extensibility to the system design.

Step 11: The Coffee service can be created by following a simple rule. Me (user) icon is connected with Friends icon which is then connected with Coffee icon. This logically means that the phone user wants to track his friends for coffee. Similarly, the Movie service can be created by connecting the icons in the following order: Me -> Friends -> Movie. This logically means that the phone user wants to track his friends for movie.
Step 12: The server already has information about the phone user and now it is time to add friends for the services created. Friends can be added to Coffee service by pressing any key on the mobile phone and then while holding it and clicking the Friends icon connected to Coffee icon on the screen. The user is instantly presented with a screen which allows it to add friends from the phonebook and delete the already added ones.

In the phonebook displayed on selecting Add Friends, it is possible to make multiple selections of friends. But since the number of resources (i.e., mobile phone and GPS receiver) used in this research are only two, it is currently possible to test the service for only one friend at a time. If more resources are available, then accordingly more friends can be tested for the service. Selecting Delete Friends shows a list of already added users. Here also, it is possible to delete as many users by making multiple selections. The user can go back to the Drawing Tool screen by clicking the Back button.

The friends for Movie service can be added or deleted by following the same steps after holding down any of the phone keys and clicking the Friends icon connected to Movie.

In the current scenario, the user “Ruchika” has added “Leena” from the phone book as its Coffee and Movie friend. In this case, it is assumed that “Leena” is also a user of MoSGen and her mobile has its GPS coordinates on the server.
**Step 13:** Now the user is finished with providing the required information for the services created. These services can now be triggered on clicking *Done* button. A message is then displayed on the topmost area of drawing panel confirming what services have been triggered.

In case the user did not add friend(s) to any of the services and clicks on the *Done* button, an alert is generated asking the user to add friend(s) for the service. If the user has any dummy or unconnected icons on the drawing panel then clicking *Done* button would display an informative error message. If the user connects icons out of rule order, for example *Friends -> Me -> Coffee*, then neither would it be able to add friends for this, nor would the Coffee service get triggered. But at the same time, if the user has created Movie service correctly and added friends for it, then only the Movie service would get triggered. Hence the execution of the other service(s) which has been correctly programmed is not affected if the user has programmed one of the services incorrectly.
Step 14: At any time on the Drawing Tool screen the user can click on the Back button to see his current GPS coordinates. The Drawing Tool has some more options that could be useful for the user. After triggering the services, the user can click on More -> Check Movie Friends to directly see the list of friends being tracked for Movie. Similarly, he can click on More -> Check Coffee Friends to directly see the list of friends being tracked for Coffee. Since currently only one friend is added for services, the user can see only one friend in the list. Also, the user could anytime check what services are currently running on the system by clicking More -> Check Services.

Step 15: The user can check the proximity distance with its Coffee and Movie friend on clicking More -> Proximity Distances.
As the users are mobile, the proximity distance also keeps changing accordingly. The proximity distance gets updated at every 10 secs.

**Step 16:** As soon as the proximity is detected between the current tracking user “Ruchika” and friend “Leena”, an appropriate message is displayed on the Proximity Distance screen.

Also, an informative SMS is automatically sent to both from the tracker’s phone (here, “Ruchika”). Since “Leena” had been added as the friend for both Coffee and Movie service, two different SMS (for Coffee and Movie) would be received by “Leena” and “Ruchika” as well.

The two SMS received by “Ruchika” for Coffee and Movie:
38

The two SMS received by “Leena” for Coffee and Movie:

<table>
<thead>
<tr>
<th>Ruchika</th>
<th>16/08/2008 07:38</th>
<th>Ruchika</th>
<th>16/08/2008 07:37</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi, Ruchika +46702071036 is nearby... Would you like to have COFFEE with your friend!</td>
<td>Hi, Ruchika +46702071036 is nearby... Would you like to goto MOVIE with your friend!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Step 17:** If in-case any of the users closes the MoSGen application, his/her phone identification and GPS coordinates entries would be removed from the server.

### 3.4. Implementation

The client component of MoSGen comprises the *Draw Canvas* application that allows users to create and trigger mobile services according to a set of visual programming rules. In order to support the client with GPS data, the implementation also includes *Bluetooth GPS* application, written using MIDP APIs. The phone identification number is retrieved by the *Cell ID* application where the user manually needs to select the correct id for the phone. These three high-level components are integrated to build a complete client solution.

The implementation of *Bluetooth GPS* application is done using an optional package, JSR-82 (or, Bluetooth API). The application first searches for all Bluetooth devices in its vicinity. The device discovery process provides information about the Bluetooth Uniform Resource Locator (URL) and the name of the Bluetooth device. The connection on the URL is then established with the GPS device using Bluetooth Serial Port Profile (btspp://) protocol. If the device is a GPS receiver, the GPS data is read from the wireless serial port. The JSR-82 API defines a communication protocol, Radio Frequency Communication (RFCOMM) which is used to read GPS data from the receiver. GPS receivers use the National Marine Electronics Association (NMEA) standard protocol to transmit data (see NMEA 0183, 2008). The data output of the receiver is a set of NMEA sentences. In this research, since we are only concerned with the geographical coordinates i.e., latitude and longitude, hence while reading the
GPS data, the sentence that starts with "$GPGGA" (Global Positioning System Fix Data) has only been considered.

The client on the other hand, communicates with the server using HTTP protocol. The server components are written using Java Servlets and it is able to connect to MySQL database using JDBC driver for MySQL.

The computation of proximity distance is performed on the server. The calculation is done using a standard mathematical approach. For the current research, the proximity calculation is simple as limited factors have been considered for the computation. Hence the proximity distance calculation is not very accurate.

The GPGGA sentence includes longitude and latitude information which are in degrees and minutes (see NMEA 0183, 2008). For example longitude data 081.51.6838, W is read as Longitude 81 deg 51.6838’ W and latitude data 41.24.8963, N is read as Latitude 41 deg 24.8963’ N. The first three digits in longitude and the first two digits in latitude are represented in degrees and the remaining digits are in minutes. The proximity distance between two points can be calculated by determining the (X, Y) coordinates of the two points, where X is longitude and Y is latitude. In this research, the unit for proximity distance has been considered as meters. Hence, it is required that the (X, Y) coordinates should also be expressed in meters. Assuming the two users are positioned at points P1 and P2, the GPS coordinates of point P1 and point P2, has been used to calculate the proximity distance between the two points as shown in the following steps:

1. The (X1, Y1) coordinate of point P1 is first determined by a simple procedure. The longitude data is converted into meters to get X1. A degree of longitude at the equator is 111200 meters and a minute is 1853 meters (see Dutch, 2008). Using these conversion factors the degrees and minutes part of the data is converted into meters and then added together to get X1. Similarly, the latitude data is converted into meters to get Y1.

2. After determining (X1, Y1) coordinate of point P1, the next step is to determine (X2, Y2) coordinate of point P2. This is done by repeating the same steps as above but considering the longitude and latitude data of point P2.

3. Now that the points P1 and P2 have been derived in meters, the distance between the two points is calculated by using the distance formula (the formula for calculation of geometric distance between two points), that is,

\[ C_1 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \]

where, C is the proximity distance between points P1 and P2 in meters.

3.5. Software Validation

Software Validation determines if the system meets its requirements specification and is free of defects. This involves inspection and reviews at each stage of development process from requirements specification to implementation.
The prototype system has been tested at three levels which involves testing the components of the system, testing the integrated system and finally, testing the system with customer’s data (see Sommerville, 2006). Whenever defects occur at any level, debugging is immediately performed which may require other levels of testing to be repeated. Since the implementation followed object-oriented approach, debugging was easy. The three levels of testing are described as below.

**Component (unit) testing:** Each unit or basic component of the system is tested to ensure that the component operates correctly. The components are tested independently.

**Integration testing:** The individual components are combined and tested as a group. This is concerned with finding errors in the interfaces and interaction between the integrated components.

**System testing:** The testing is performed on the integrated system to ensure that the system conforms to its specifications.

All these tests performed during the development process allowed discovering and eliminating problems in the system.
Chapter 4

4. Software Evaluation

This chapter presents user evaluation of MoSGen where the feedbacks on the overall behaviour and usability of the system have been analysed.

4.1 Approach

The evaluation of MoSGen was performed by three novice users. All users were professionals in their work domains but had no prior learning and experience of programming. However, the users were well aware of generic functional aspects of computer, mobile phone and PDA.

Prior to using the system, the users were explained about the purpose and characteristic features of the system. This was followed by giving a small demo of the sequence of operations required to be performed until the user successfully gets connected and acquires the GPS coordinates. After this, they were explained about the constructs visible on the visual programming interface and how to use them to create one of the mobile services. For the other service, they were given the required explanation and were asked to program themselves in the evaluation period. Buddy tracking services need at least two users to see if the SMS could be sent and received on proximity detection. Therefore for evaluation purpose, the author enacted as the buddy for Coffee and Movie service for each of the three users. But, no other support or suggestion was provided during the evaluation period. A wireless network was used during the evaluation. Therefore, the proximity distance was limited to the range of wireless network of the location and was kept up to 20 meters. However, by using a mobile network, the proximity range could be largely increased.

The three users were asked to keep record of the time duration they consumed to complete the evaluation. During the evaluation they were able to:

- Send their phone id to the server.
- Get GPS coordinates for their phone.
- Create and trigger both Coffee and Movie services in parallel.

As an output, the tracker and the author received an SMS for both the services when the author was in the proximity range of the tracker.

When the evaluation was done, the users were asked to fill up a small evaluation form.
4.2 Results

The results of the evaluation show how the users felt about the functionality and usability of the system. Each response was scored on a scale of 1 to 5 (1 – very poor, 2 – poor, 3 - satisfactory, 4 – good, 5 - excellent). The evaluation results are shown in Table 4.1.

Table 4.1: User evaluation results

<table>
<thead>
<tr>
<th></th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease to get GPS coordinates</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Ease to create mobile services</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Ease of use of icons and connectors on visual programming screen</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Overall experience</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Usefulness of the system</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Evaluation time (in minutes)</td>
<td>26</td>
<td>30</td>
<td>25</td>
<td>27</td>
</tr>
</tbody>
</table>
Chapter 5

5. Discussion and Conclusions

This chapter presents the discussion on evaluation of results and conclusions. Finally, some suggestions for further research are also presented.

5.1 Discussion

The evaluation of MoSGen was performed by three novice users and all of them gave positive results. They were quite satisfied with the process of getting GPS coordinates which they found simple and straightforward. The strength of the wireless network and the quality of satellite connectivity was good enough and they were able to get the GPS coordinates in very less time. But since the position accuracy of GPS receivers is not 100 percent, it often gives multiple values for the same position. Hence, it may be possible that two users are within a distance of 20 meters but the receivers show inaccurate position values because of which the proximity distance calculated by the server is more than 20 meters. But, it was observed that as the distance between users increases, the accuracy of results also increases. Thus, the system would become more reliable if larger proximity distances are considered, using mobile network over wireless network.

The users enjoyed playing with the icons and connectors to create mobile services which could be done using a small set of these visual notations. The logical rules were easy to follow. However, they found one feature a bit uncomfortable while performing visual programming. One has to first click on the icon and then a command from the menu to draw icons on the display and could not directly drag and drop the icons on the display. Due to some limitations of J2ME which is not as powerful as Java, direct drag-and-drop feature could not be implemented for the tool. The ease of performing visual programming also depends on individual skills to some extent. One user could do it very fast while the other one could take more time to get used to it. Otherwise, the overall experience was good in all the cases.

The usefulness of the system was considered satisfactory but good enough for a prototype solution. A lot more work needs to be done to make it more professional, usable and have support for more interesting mobile services which could be simple or advanced. The current version of MoSGen prototype showed how it is possible for novice end users to create simple mobile services by using a set of visual notations.

The time difference between users at triggering and testing the services was mainly due to their ability to use icons and connectors on small screens like that of P1i.
5.2 Conclusions

In the current research, a literature review was done to study the different visual programming tools developed for novice end users. In relation to the research question of the thesis, a user friendly visual programming tool, MoSGen, has been designed and developed for novice end users to develop simple mobile services by performing visual programming. Two mobile services have been incorporated into MoSGen to demonstrate the tool’s applicability. A GPS application has been developed to implement buddy tracking service. The tool evaluation was performed by three novice users who provided highly positive feedbacks. With this, the author believes that the initial expectations of the software have been successfully fulfilled. The system has been so designed that adding new features and functionalities can be easily done if the developer has a good understanding of the working principles of the underlying source code. Further developments of MoSGen prototype can provide interesting and extremely useful software which would help wide range of mobile users adopt and use mobile services in their daily lives.

5.3 Scope For Further Research

The current research showed how simple mobile services can be created by performing visual programming. The same tool can be extended to program advanced services by adding more features and functionalities to the system.

MoSGen has the capability to run multiple services. Currently only two mobile services are supported, and were developed for the purpose of prototype evaluation. But the tool can be extended to add support for more number of services. With the growth of mobile industry, the memory capacity of mobile phones would also be increasing. Increased memory would make it possible to run more services in parallel.

It could also be noted that since the current research supports only two mobile services, the size of the drawing panel (Please refer Section 3.3.4 for details) has been kept up to the screen size which is sufficient for creating two services. But in future if the system supports more number of services then the size of the panel can also be increased to as much length required, just by adding a vertical scrollbar to the panel.

Location based services usually require knowledge of user current position in terms of coordinates. In future it is most likely that technologies such as GPS would be inbuilt in most of the touch screen phones. The availability of inbuilt GPS in phones suitable for MoSGen would eliminate manual configurations the end user needs to perform to get GPS coordinates from an external GPS receiver.

In this research we faced problems in deciding a suitable phone for MoSGen. Turning this prototype solution into the actual product would considerably take some time. Hopefully, it may be possible that by then, the system would be portable on many platforms and essential technical features like touch screen etc., would be available in
more number of mobile phones so that most of the customers would be able to use it on their phones or PDAs.

The implementation of MoSGen is currently based on CLDC MIDP configuration. In further research to develop the actual software, the user interface could be made more appealing and professional. This can be achieved by using CDC PP configuration instead as it supports Java AWT classes. Also, CDC is a superset of CLDC and contains all the libraries supported by CLDC. Therefore, it could be a development choice for J2ME applications in future research.
REFERENCES

Axim Site, TED’s Bluetooth Guide,  
Link: http://www.aximsite.com/bluetooth/index.htm#section2,  


Dutch, S. (2008) *Converting UTM to Latitude and Longitude (Or Vice Versa)*, University of Wisconsin Green Bay,
Link: http://www.uwgb.edu/dutchs/UsefulData/UTMFormulas.htm


Institue Mihailo Pupin, *GPS-Based Applications*,
Link: http://www.imp.bg.ac.yu/Products/gps.htm,


Sun Microsystems, *Java ME at a Glance*,
Link: http://java.sun.com/javame/index.jsp,
Last accessed: 2\textsuperscript{nd} July, 2008.

Sun Microsystems, *Using the Java APIs for Bluetooth Wireless Technology*,
Link: http://developers.sun.com/mobility/apis/articles/bluetoothintro/index.html,
Last accessed: 2\textsuperscript{nd} July, 2008.
