

INTELLIGENT APPLICATION DEVELOPMENT FOR MOBILE DEVICES

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
ÇANKAYA UNIVERSITY

BY

UĞRAŞ ÖZKAN

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
COMPUTER ENGINEERING

JANUARY 2007

Title of the Thesis: **Intelligent Application Development for Mobile Devices**

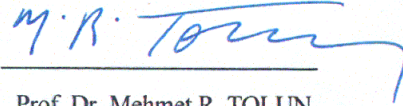
Submitted by **Uğraş Özkan**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University



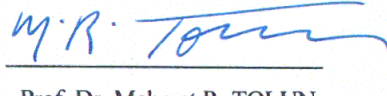
Prof. Dr. Yurdahan GÜLER  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Mehmet R. TOLUN  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

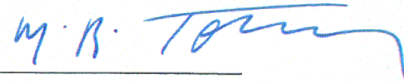


Prof. Dr. Mehmet R. TOLUN  
Supervisor

Examination Date: 18/01/2007

Examining Committee Members:

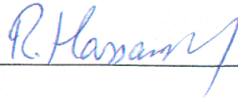
Prof. Dr. Mehmet R. TOLUN (Çankaya Univ.)



Assoc. Prof. Dr. Ferda ALPASLAN (METU)



Asst. Prof. Dr. Reza HASSANPOUR (Çankaya Univ.)

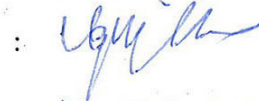


## STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Uğraş ÖZKAN

Signature :



Date :

18.01.2007

## ABSTRACT

### INTELLIGENT APPLICATION DEVELOPMENT FOR MOBILE DEVICES

ÖZKAN, Uğraş

M.Sc., Department of Computer Engineering

Supervisor : Prof. Dr. Mehmet R. TOLUN

January 2007, 82 pages

Mobile systems are becoming more dominant in usage according to their abilities and they are offering a considerable market opportunity for software. Artificial Intelligence technologies offer the potential to create compelling software for the mobile platform. However, few intelligent applications have been developed for mobile systems. The main reason for this lack of development is to mobile device limitations, such as memory and processing power. The main purpose of this thesis is to show that although some limitations, a mobile device should be connected to an expert system.

**Keywords:** Expert System, Mobile Applications, Intelligence Applications, Java, Jess

## ÖZ

### MOBİL AYGITLARDA AKILLI UYGULAMALAR GELİŞTİRMEK

ÖZKAN, Uğraş

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi : Prof. Dr. Mehmet R. TOLUN

Ocak 2007, 82 sayfa

Mobil sistemler kapasiteleri ölçüsünde kullanım olarak daha baskın hale gelmektedirler. Bu da mobil sistemler için yazılım alanında önemli bir pazar sağlamaktadır. Yapay zeka teknolojileri mobil sistemler için potansiyel olarak yeni fırsatlar sunmaktadır. Fakat mobil sistemler için çok az sayıda zeki uygulama geliştirilmektedir. Bunun en önemli sebebi olarak mobil sistemlerdeki hafıza ve işlemci gücündeki sınırlamalar gelmektedir. Bu tezin amacı mobil bir sistemle, uzman bir sistemin, mobil sistemlerdeki sınırlamalara karşın, birbiri arasında bağlantının yapılabildiğini göstermektir.

**Anahtar Kelimeler :** Uzman Sistem, Mobil Uygulamalar, Akıllı Sistemler, Java, Jess

## **ACKNOWLEDGEMENT**

I would like to express my special thanks to my supervisor Prof. Dr. Mehmet R. TOLUN for his supports and suggestions.

to my Sirinim for her patience.

## TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM PAGE.....	iii
ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGEMENT.....	vi
TABLE OF CONTENTS.....	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
LIST OF SYMBOLS.....	xii
<b>CHAPTERS:</b>	
<b>1. INTRODUCTION</b>	
1.1 Problem Statement.....	1
1.2 Research Objectives.....	2
1.3 Scope of Study.....	2
<b>2. EXPERT SYSTEMS AND DEVELOPMENT TOOLS</b>	
2.1 Expert Systems.....	3
2.1.1 Artificial Intelligence (AI).....	3
2.1.2 What is an Expert System.....	5
2.1.3 Concept of Inference Engine.....	8
2.1.3.1 Knowledge Base.....	8
2.1.3.2 Inference Engine.....	11
2.1.4 The Expert System Development Life Cycle.....	12
2.1.4.1 The Prototyping Approach.....	12
2.1.4.2 The Feasibility Study .....	15
2.1.4.4 Maintenance of Expert System.....	16
2.1.5. Expert System Shells.....	17

2.2	Jess Expert System Shell.....	17
2.2.1	General Information.....	17
2.2.2	Jess Performance.....	18
2.3	Applications and the Future.....	19
2.3.1	Applications of Expert System.....	19
2.3.2	Emerging Applications Expert Systems.....	19
2.3.3	The Future of Expert Systems.....	20
<b>3. INTELLIGENT SYSTEM DEVELOPMENT WITH MOBILE DEVICES</b>		
3.1	Mobile Device World.....	22
3.1.1	Mobile Commerce.....	22
3.1.2	Mobile Technology Adoption.....	23
3.1.3	Mobile Device Manufacturers.....	24
3.1.4	Mobile Software Platform Providers.....	25
3.2	Smart Mobile Devices.....	25
3.3	Application Development over Mobile Device with Java.....	26
3.3.1	Why Java.....	26
3.3.2	Java Micro Edition (J2ME) Architecture.....	28
3.3.3	Competing Technologies.....	30
3.4	Developing End-to-End Systems.....	30
3.4.1	Simple Object Access Protocol (SOAP) and Web Service....	31
3.4.2	kSOAP.....	33
3.4.3	Remote Procedure Call (RPC).....	34
3.4.3.1	Remote Method Invocation (RMI).....	36
3.4.3.2	Java Micro Edition RMI Optional Package.....	36
3.5	An Application: Chest Pain Mobile Expert System (MES).....	37
3.5.1	Introduction.....	37
3.5.2	What does MES.....	38
3.5.3	How does MES Works.....	38
3.5.4	The problem solving paradigm.....	39
3.5.4.1	The paradigm.....	39
3.5.4.2	What problems do MES handles.....	39
4.	<b>CONCLUSION.....</b>	<b>41</b>



**REFERENCES.....R1**

**APPENDICES:**

**A. Structure of SOAP Message.....A1**  
**B. Decision Rules to the Mobile Expert System.....B1**  
**C. Mobile Expert System Facts.....C1**  
**D. Sample Dialogue on the Mobile Expert System.....D1**  
**E. Flow Design of Mobile System.....E1**  
**F. Web Service Definition Language of Mobile Expert System.....F1**

## LIST OF TABLES

### TABLES

<b>Table 1</b>	: Conventional and Expert System Life Cycle Stages.....	12
<b>Table 2</b>	: Feasibility Assessment Issues.....	15
<b>Table 3</b>	: Operating Systems and Programming Languages.....	25

## LIST OF FIGURES

### FIGURES

<b>Figure 1</b>	: Modules in an Expert System.....	7
<b>Figure 2</b>	: Parts of Expert System.....	8
<b>Figure 3</b>	: An example of a Semantic Network.....	10
<b>Figure 4</b>	: Stage Based Approach to System Development.....	13
<b>Figure 5</b>	: Incremental Expert System Prototyping.....	14
<b>Figure 6</b>	: Technology Adoption Curve.....	23
<b>Figure 7</b>	: Overview of J2ME.....	29
<b>Figure 8</b>	: Basic Usage of Web Services.....	32
<b>Figure 9</b>	: Web Service Architecture.....	33
<b>Figure 10</b>	: The Principle of Middleware.....	35
<b>Figure 11</b>	: RMI Components.....	36
<b>Figure 12</b>	: End-to-End System Architecture.....	39

## LIST OF SYMBOLS

### SYMBOLS

<b>API</b>	: Application Programming Interface
<b>BREW</b>	: Binary Runtime Environment for Wireless
<b>CDC</b>	: Connected Device Configuration
<b>CLDC</b>	: Connected Limited Device Configuration
<b>CORBA</b>	: Common Object Request Broker Architecture
<b>CPU</b>	: Central Processing Unit
<b>GUI</b>	: Graphical User Interface
<b>HTTP</b>	: Hypertext Transfer Protocol
<b>J2SE</b>	: Java Standard Edition
<b>J2EE</b>	: Java Enterprise Edition
<b>J2ME</b>	: Java Micro Edition
<b>JAR</b>	: Java Archive
<b>JCP</b>	: Java Community Process
<b>JNI</b>	: Java Native Interface
<b>JSP</b>	: Java Community Process
<b>JSR</b>	: Java Specification Request
<b>JVM</b>	: Java Virtual Machine
<b>MIDP</b>	: Mobile Information Device Profile
<b>PBP</b>	: Personal Basis Profile
<b>RMI</b>	: Remote Method Invocation

- RPC** : Remote Procedure Call
- SAML** : Security Assertion Markup Language
- SDK** : Software Development Kit
- SOAP** : Simple Object Access Protocol
- UDDI** : Universal Description, Discovery, and Integration
- URL** : Uniform Resource Locator
- WAP** : Wireless Application Protocol
- WSDL** : Web Services Description Language
- XML** : Extensible Markup Language

## CHAPTER 1

### INTRODUCTION

#### 1.1 Problem Statement

The first generation of mobile devices has limited capabilities, their processor or memory size is very small and also their prices are very high to gain it. However today, mobile device technology has been improved much more and also a mobile device should be easily found by each person

After mobile system capabilities reach the necessary level (e.g. processor power, memory and screen size), and improvements on mobile operating systems started new era. This is meant that a mobile device is used as small computer. The new market for software industry is opened. Mobile device producers, telecommunication service providers, software houses and much more industry affected from this innovation.

The main purpose of mobile system is communication at the beginning. However, with the enormous progress in mobile world, people thought that they should be used other than communication. So, mobile devices have been used for entertainment (e.g. games, cameras), connecting to the internet via Wireless Application Protocol (WAP) and as a personal computer.

To develop an Intelligence application usually needs more powerful hardware. After the progress of mobile world, mobile systems have the ability to run simple intelligent applications especially expert systems. There are many different ways to run an Expert System with a mobile device. One of them is that developing an expert system on the mobile system which means that mobile expert system is created and deployed to mobile device, all the things occurs inside the mobile device. The other one is that developing thin client application which means that only the Graphical User Interface is located at the mobile system and an expert system that run on a different server. Mobile system and expert system connection is provided by some other protocols (e.g. Web Services, Remote Procedure Call etc.)

## **1.2 Research Objectives**

Intelligent Systems (classes of which include Expert Systems and Intelligent Software Agents) use an approach to problem solving which is different to conventional data and information processing algorithms. Intelligent Systems rely on knowledge and inference to solve complex problems, and have been applied to a huge range of problems in many domains and markets. Intelligent Systems effectively make specialist knowledge available for use by non-specialist users.

Intelligent Systems are commonly thought to require powerful computing environments with powerful processors and a large amount of memory in order to function effectively. In fact, this is only true during the development phase of Intelligent System implementation; once they are built they can be deployed on much less powerful devices. Specific objectives of this research that were realized are enumerated below.

- To show that Intelligence applications should be developed on mobile devices.
- To develop an expert system integrated with various mobile device

## **1.3 Scope of Study**

In this thesis, the main purpose is to create end-to-end mobile system. A mobile device in this thesis, Nokia 6630 with Symbian Operating System is used as client that is developed with Sun NetBeans. Symbian Operating system support Java applications. An expert system is developed with Jess Expert System Shell which is free for the academic purposes. Complicated expert system is not the purpose of this thesis so an expert system with about 20 rules and 50 facts is developed. A Java web service is created with Oracle JDeveloper and deployed to application server (Oracle Application Server). Due to Jess is created with java, the connection between Java Web Service and expert system should be done easily.

## CHAPTER 2

### EXPERT SYSTEMS AND DEVELOPMENT TOOLS

#### 2.1 Expert Systems

##### 2.1.1 Artificial Intelligence (AI)

AI is a result of the merge of philosophy, mathematics, psychology, neurology, linguistics, computer science, and many other fields. Furthermore, the application of AI relates to almost any fields. This variety gives AI an endless potential. A relatively young science, AI has made much progress in 50 years. Though fast-growing, AI has never actually caught up with all the expectation imposed on it. There are two reasons for public's over-confidence in AI. First, AI theories are often ingenious and subtle even fictional, implying much futuristic applications. Second, AI, being incorporated with computer technology, is often expected to progress as fast as the computer technology [1].

It wasn't until the postwar period (1945-1956) that Artificial Intelligence would emerge as a widely-discussed field. What propelled the birth of Artificial Intelligence were the arrival of modern computer technology and arise of a critical mass. Pioneers such as Marvin Minsky, John McCarthy, Allen Newell, and Herbert Simon led their students in defining the new and promising field. The development of the modern computer technology affected the AI research tremendously. Many pioneers of AI broke away from the traditional approach of artificial neurons and decided that the human thought could be more efficiently emulated with modern digital computer. Those who did not accept digital computers as the new approach stayed in the parallel field of neural network.

The Dartmouth Conference of 1956 brought AI to a new era. 1956-1963 represents the dawning of an intensive AI wave. During this period, major AI research centers such as Carnegie Mellon, MIT and its Lincoln Laboratory, Stanford, and IBM concentrated their work on two main themes. First, the attempt to limit the breadth of searches in trial-and-error problems led to the initiation of projects such as Logic Theorist (considered as the first AI program), Geometry Theorem Prove, and SAINT. Next, the study on computer learning includes projects on chess, checkers, and pattern recognition programs. Specialized list-processing AI languages such as LISP were also developed in MIT and other places in 1958.



By mid 60's, AI had become the common goal of thousands of different studies. AI researchers utilized their programming techniques and the improved computers in pursuing various projects. However, the memories of computers during these years were still very limited. Perception and knowledge representation in computers became the theme of many AI researches. One representative project was the Blocks Micro World project carried out in MIT. Facing a collection of pure geometric shapes, the robots looked through cameras and interpreted what they had seen. Then, they would move about, manipulate blocks and express their perceptions, activities, and motivations. With the booming of AI, the rival science of artificial neural network would face the downfall especially after the exposure of basic flaws in its research in "Perception" by Minsky and Papert.

Different AI-related studies had developed into recognizable specialties during the 70's. Edward Feigenbaum pioneered the research on expert systems; Roger Schank promoted language analysis with a new way of interpreting the meaning of words; Marvin Minsky propelled the field of knowledge representation a step further with his new structures for representing mental constructs; Douglas Lenat explored automatic learning and the nature of heuristics; David Marr improved computer vision; the authors of PROLOG language presented a convenient higher language for AI researches. The specialization of AI in the 70's greatly strengthened the backbone of AI theories. However, AI applications were still few and premature.

The 1980's were a period of roller coasting for AI. The anti-science tradition of the public was improved greatly following the appearance of Star Wars movies and the new popularity of the personal computers. XCON, the first expert system employed in industrial world, symbolized the budding of real AI application. Within four years, XCON had grown tenfold with an investment of fifty person-years in the program and an achievement of saving about forty million dollars in testing and manufacturing costs for the industrial clients. Following the brilliant success was the AI boom. The number of AI groups increased tremendously and in 1985, 150 companies spent about \$1 billion altogether on internal AI groups. However, the fundamental AI algorithm was still unsatisfying. These seemingly intelligent programs simply make dumb decisions faster. Indeed, the warning foreshadowed the downfall of AI industry in late 80's. The replacing of LISP machines by standard microcomputers with AI software in the popular C language in 1987 and the instability of expert systems caused a painful transition on expert system industry; the computer vision industry also suffered from a big setback when Machine Vision International crashed in 1988; one other major loss was the failure in Autonomous Land Vehicle project (AI drivers + Robotics). The AI industry started recovering at the end of the 80's but learning from the past experience, public assumed a much more conservative view on AI ever since. Another notable event is the revisiting of neural network with the work done by the Parallel Distributed Processing Study Group. In 1989, about three hundred companies were founded to compete for the predicted \$1 billion market for neural nets by the end of the century.

The Persian Gulf War in the early 90's proved the importance of AI research for military use. Tasks as simple as packing a transport plane and as complicated as the timing and coordination of Operation Desert Storm were assisted by AI-oriented expert systems. Advanced weapons were equipped with technologies previously studied in different AI-related fields such as Robotics and Machine Vision. Two projects succeeding the Automated Land Vehicle project were the Pilot's Associate project (electronic copilot) and the Battle Management System project (military expert systems).

The victory of Deep Blue over chess champion Kasparov in 1996 led to a new summit of AI gaming. A new branch of expert systems has been expected to prosper as Genetic Engineering matures. Manipulating such a gigantic knowledge base of human DNA map (Bioinformatics) will require very specialized algorithms and AI researches [1, 2, 3].

### **2.1.2 What is an Expert System?**

An expert system is a computer program that uses knowledge and inference techniques to solve problems. The expert system is an improved tool which is based on the decision-making property of a human expert. The most common form of an expert system is a program which is based on rules that analyze information about a specific class of problems. Expert system is a knowledge-based and also rule-based system.

Expert system is constructed to solve problems in the fields of accounting, medicine, process control, financial service, production, human resources and so on. Expert system concept is a branch of AI that performs deep usage of knowledge to solve problems with assistance of a human expert.

The basic distinction between expert systems and traditional problem solving programs is the coding types. In traditional programs, the problem expertise is encoded in both program and data structures. Traditional computer programs use conventional decision-making logic with a limited knowledge. This knowledge is embedded as a part of the programming code and when the knowledge changes, the program is reconstructed. In the expert system application, the problem expertise is encoded only in data structures.

Expert systems were born when it was realized that there was at least one aspect of intelligence that was not based on reasoning. An expert dealing with a problem in his field often uses very simple reasoning, relying more upon the knowledge gained from years of experience and training. This insight into the role played by knowledge in the cognitive process encouraged AI researchers to build systems that apply simple reasoning mechanisms to knowledge about a very specific area of expertise [4, 5, 6].

The term "expert system" refers to a system that uses contemporary computer technology to store and interpret the knowledge and experience of a human expert in a specific area of interest. By accessing this database of knowledge stored in a computer, a non-expert can get the benefit of expert advice in that area.

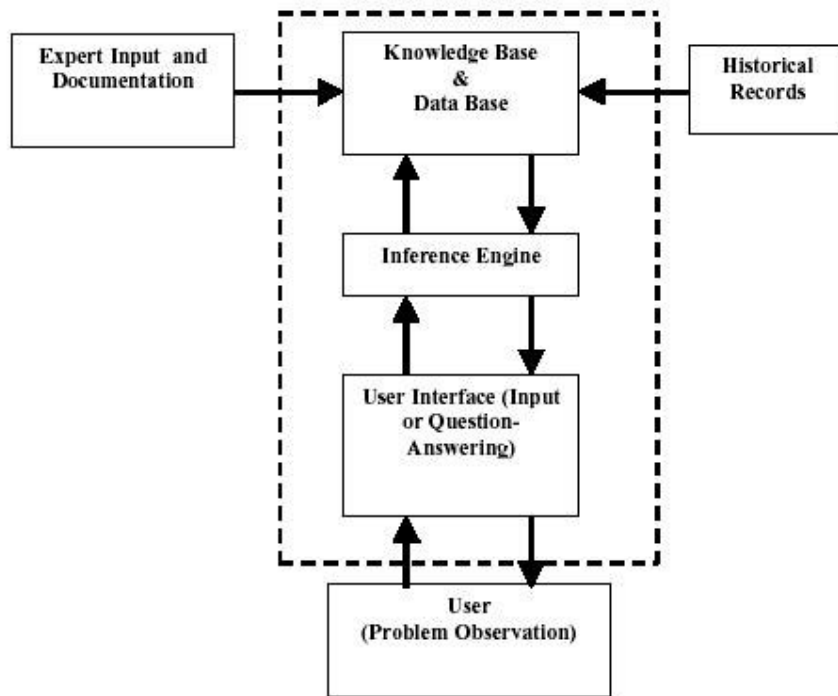
Expert system is most likely to be inferior to the individual whose expertise was used in developing it, or can at the most be as good as that individual. However, this may not always be the case. For instance, there are chess-playing systems that demonstrate a much higher proficiency in chess than the humans who helped design them. This example shows that it is not fair to characterize a computer-based expert system as necessarily being inferior to a human expert. Of course, expert systems do have some characteristics that distinguish them from other computer-based tools.

The most obvious feature of an expert system is that it operates as an interactive system that responds to questions, asks for clarifications, makes recommendations and generally aids the decision-making process. To a user, this interactive interface is what would distinguish an expert system from any ordinary computer tool. Behind this interface lie other characteristics that may not be immediately obvious to a person using the tool. Expert system tools have the ability to store and sift through significant amounts of knowledge. There are various mechanisms used in the storage and retrieval of knowledge. An expert system needs a large knowledge base in order to be able to tackle any kind of problem that may arise within its area of expertise. Not only must such a system be able to store the available knowledge, but it must also support mechanisms to expand and improve the knowledge base on a continuing basis. Every specialized field is always in a state of flux, with something new being discovered all the time. In order to keep the expert system up-to-date, it is necessary to leave the knowledge base open-ended so that new pieces of information can be added at any time, without need for significant changes in the structure of the system.

An expert system must have the capability to make logical inferences based on the knowledge stored. This is where the simple reasoning mechanisms used in expert systems come into play. This is what makes an expert system tick. A knowledge base, without any means of exploiting the knowledge stored, is useless. This would be analogous to learning all the words in a new language, without knowing how to combine those words to form a meaningful sentence.

Models can never be 100% accurate, since no expert is omniscient. It is important that users of expert systems exercise caution while interpreting the results produced by expert systems. An expert system, much like any computer program consists of a set of inputs, a set of outputs and a set of modules, which are designed to map the inputs into the desired outputs. The modules consist of the inference machine in the form of rule base. The inputs provided by the user, through user interface, are processed by the inference engine. The output is derived by the application through inductive reasoning that is based on expert knowledge embodied in the knowledge base.

A schematic representation of the modules in an expert system is shown in Figure 1. The components enclosed within the dashed lines knowledge base, inference engine and user interface, constitute the physical body of the expert system. The expert knowledge and facts from historical records are stored in the knowledge base in the form of rules. The user inputs and answers to questions that are posed by the system through the user interface. The inference engine examines the inputs with existing facts and rules and decides the order in which inferences are made. The final inferences and answers provided by the system are presented to the user, through the user interface [4].



**Figure 1:** Modules in an Expert System

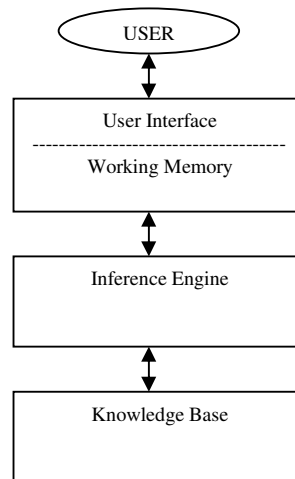
A feature somewhat unique to expert systems is that a particular system caters to a relatively narrow area of specialization. Expert systems are very domain-specific. A medical expert system cannot be used to find faults in the design of an electrical circuit. This focus on small domains is more a result of technological limitations than anything else. The quality of advice offered by an expert system is dependent on the amount of knowledge stored. As the scope of an expert system is widened, its knowledge base needs to be expanded. The methodologies available today limit the amount of knowledge that can be stored and retrieved in reasonable amounts of time. So, the constraints set by existing technology make it necessary to build expert systems that cater to relatively narrow domains.

The applications best suited for expert systems are those dealing with expert heuristics for solving problems. Any field, in which problems can be solved using purely numerical techniques, within reasonable periods of time, is not a suitable choice for the domain of an expert system. Building an expert system for such a field cannot be justified as there would be no advantage in doing so.

Expert systems have become increasingly popular because of their specialization, albeit in a narrow field. The small size of the domain makes encoding and storing the domain-specific knowledge an economic process. Also, as specialists in many areas are scarce, and the cost of consulting them is high, an expert system catering to any of those areas can be considered to be a useful and cost-effective alternative, in the long run [1, 7].

### 2.1.3 Concept of Inference Engine

In order to design an expert system, it is necessary to become familiar with the way such systems are structured. An expert system has three levels of organization knowledge base, working memory and inference engine. Apart from these, it has a user interface which permits the user to interact with the system [4, 6]. Figure 2 shows the relationship between these four parts of an expert system.



**Figure 2:** Parts of Expert System

#### 2.1.3.1 Knowledge Base

The knowledge base is the module around which the expert system is built. It contains the formal representation of the information provided by the domain expert. This information may be in the form of problem-solving rules, procedures, or data intrinsic to the domain. To incorporate this information into the system, it is necessary to make use of one or more knowledge representation methods.

The main bottleneck in the development of expert systems is the problem of knowledge acquisition. Transferring knowledge from the human expert to a computer is often the most difficult part of building an expert system. An expert's skill lies in performing a given task, not in explaining to others how it should be done. It could also happen that the expert may be reluctant to share his knowledge with others, fearing increased competition resulting in his becoming dispensable.

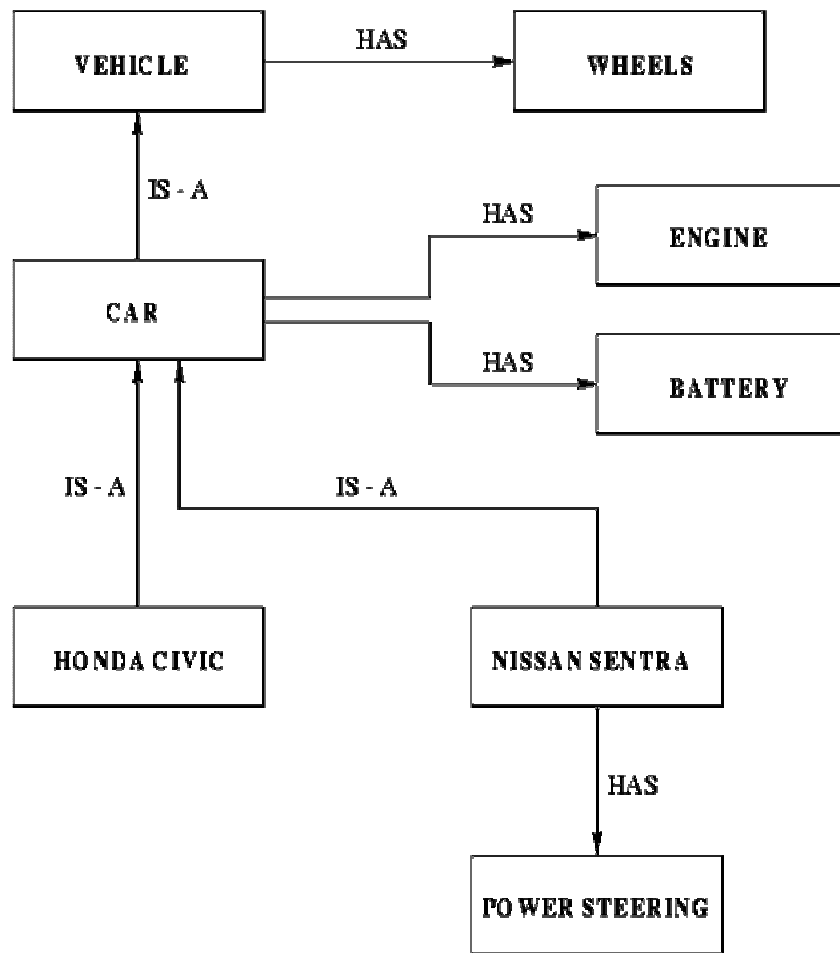
People have tried to automate the process of eliciting knowledge in an attempt to deal with this problem. The knowledge acquired from the human expert must be encoded in such a way that it remains a faithful representation of what the expert knows, and it can be manipulated by a computer. Three common methods of knowledge representation that have evolved over the years are IF-THEN rules, semantic networks and frames.

- **IF – THEN rules**

These are the form of IF A1 THEN B2 where each is a condition or situation, and each is an action or a conclusion. Because human experts usually tend to think along condition  $\Rightarrow$  action or situation  $\Rightarrow$  conclusion lines, IF-THEN rules are the predominant form of encoding knowledge in expert systems.

- **Semantic Networks**

In this scheme, knowledge is represented in terms of objects and relationships between objects. The objects are denoted as nodes of a graph, with the relationship between two objects being denoted as a link between the corresponding two nodes. The most common form of semantic networks uses the links between nodes to represent IS-A and HAS relationships between objects. In the example shown in Figure 3, a car IS-A vehicle; a vehicle HAS wheels. This kind of a relationship establishes inheritance hierarchies in the network, with objects lower down in the network inheriting properties from objects higher up.



**Figure 3:** An example of a Semantic Network

- **Frames**

In this technique, knowledge is decomposed into highly modular pieces called frames, which are generalized record structures. For instance, knowledge may consist of concepts and situations, attributes of concepts, relationships between concepts, and procedures to handle relationships as well as attribute values. In such a case, each concept may be represented as a separate frame. The attributes, the relationships between concepts and the procedures are allotted to slots in a frame. The contents of a slot may be of any data type - numbers, strings, functions or procedures and so on. Also, frames may be linked to other frames, providing the same kind of inheritance as that provided by a semantic network.

### 2.1.3.2 Inference Engine

The inference engine is the generic control mechanism that applies the axiomatic knowledge present in the knowledge base to the task-specific data to arrive at some conclusion. This is the second key component of all expert systems. Having a knowledge base alone is not of much use if there are no facilities for navigating through and manipulating the knowledge to deduce something from it.

As a knowledge base is usually very large, it is necessary to have inference mechanisms that search through the database and deduce results in an organized manner. A few techniques for drawing inferences from a knowledge base are described here. Consider the following set of rules:

**Rule 1:** IF A and C THEN F

**Rule 2:** IF A and E THEN G

**Rule 3:** IF B THEN E

**Rule 4:** IF G THEN D

- **Forward chaining**

Suppose it needs to be proved that D is true, given A and B are true. Start with Rule 1 and go on down till a rule that “fires” is found. In this case, Rule 3 is the only one that fires in the first iteration. At the end of the first iteration, it can be concluded that A, B and E are true. This information is used in the second iteration. This time Rule 2 fires adding the information that G is true. This extra information causes Rule 4 to fire, proving that D is true. This is the method of forward chaining, where one proceeds from a given situation toward a desired goal, adding new assertions along the way. In expert systems, this strategy is especially appropriate in situations where data are expensive to collect, but few in quantity.

- **Backward chaining**

In this method, one starts with the desired goal, and then attempts to find evidence for proving the goal. Returning to the previous example, the strategy to prove that D is true would be as follows. First, find a rule that proves D. Rule 4 does so. This provides a sub-goal to prove that G is true. Now Rule 2 comes into play, and as it is already known that A is true, the new sub-goal is to show that E is true. Here, Rule 3 provides the next sub-goal of proving that B is true. But the fact that B is true is one of the given assertions. Therefore, E is true, which implies that G is true, which in turn implies that D is true. Backward chaining is useful in situations where the quantity of data is potentially very large and where some specific characteristic of the system under consideration is of interest. Typical situations are various problems of diagnosis, such as medical diagnosis or fault-finding in electrical equipment.



## 2.1.4 The Expert System Development Life Cycle

### 2.1.4.1 The Prototyping Approach

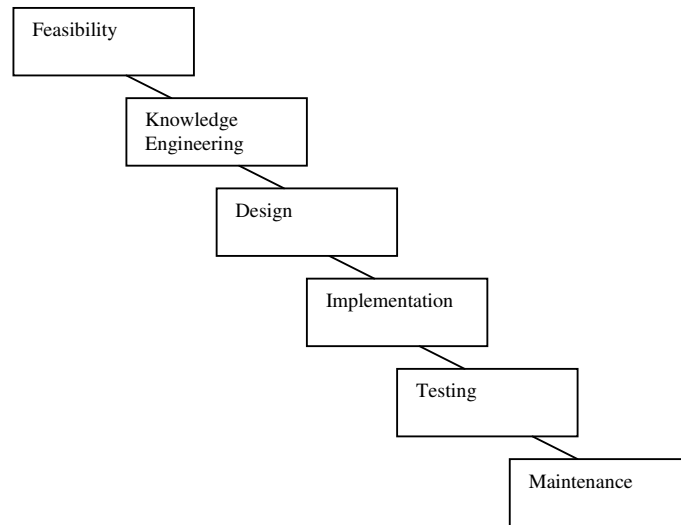
Conventional software development is often viewed in terms of applicable lifecycle paradigms. This is because software development goes through a number of stages from initial conception to the finished product. For example, a commercial software company that is considering writing an accounts package would probably undertake a feasibility stage initially to see if the project is likely to be commercially viable. If the feasibility study has given the project a green light then the next stage may involve a detailed analysis of the problem; the stage following problem analysis might be the design of the new system, and so on [4, 6, 8].

As it happens, the approach in developing expert system software has some similarities with conventional system. There are however, some differences as indicated Table 1. This figure depicts a typical breakdown of stages for both conventional and knowledge based systems.

**Table 1:** Conventional and Expert System Life Cycle Stages

<b>Conventional System</b>	<b>Knowledge-based System</b>
1. Feasibility Study	1. Feasibility Study
2. System Analysis	2. Knowledge Engineering
3. Design	3. Design
4. Implementation	4. Implementation
5. Testing	5. Testing
6. Maintenance	6. Maintenance

Despite many stages that appear to be similar, differences to emerge, particularly in the second phase. A number of methodologies have been adopted for expert system development. A methodology borrowed from conventional system development techniques is the stage-based approach which treats development as a sequential process of completed stages as shown in Figure 4.

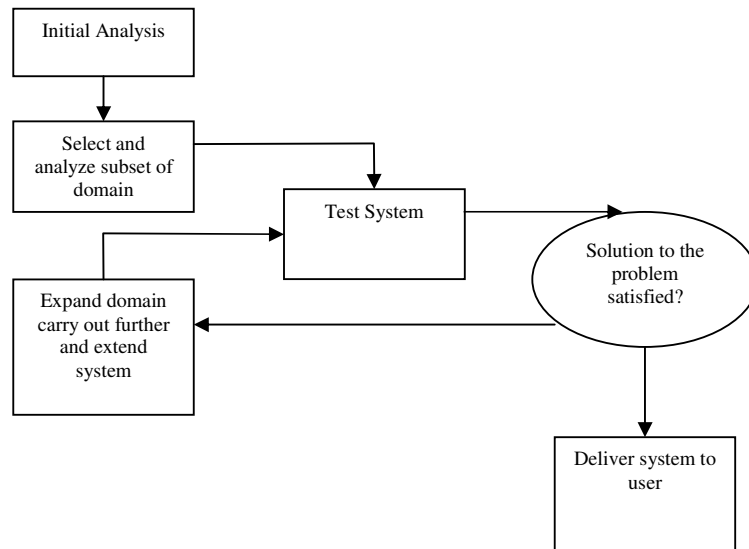


**Figure 4:** Stage Based Approaches to System Development

Conventional system analysis is well defined activity in the sense that the details and duration of each task can be documented, monitored, managed and sustained if the project has been scheduled with adequate resources. Knowledge engineering is very different; for one thing, it is frequently difficult to specify, with clarity, the requirements are an expert system because of the abstract nature of the knowledge. In expert systems, the end goals are typically not clearly defined. The goals are said to be “soft”. This reflects the general difference between an expert system and a conventional system in that the former is concerned with the encapsulation, representation and manipulation of knowledge, which is often abstract in nature, while the latter focuses on the processing of data, which is clearly defined. Also, even if the requirements are specified completely, experts frequently have difficulty articulating their knowledge, so modifications to the knowledge base. The iterative approach of development is by far the most successful paradigm used for expert systems development; it is called prototyping. That is the number of stages, could vary between the knowledge engineering and the testing state. In practice the prototyping stage will often involve the building of a rapid prototype. The idea is to provide a rapid springboard for discussion with users and experts and to demonstrate the system at an early stage to managers who may be skeptical about the use of expert system technology. Prototyping has been described as a “revolutionary change in development process” because it departs from the conventional software engineering approach. This prototype may be refined an unlimited number of times as a result of feedback and evaluation from users or experts [1, 4].

To understand how prototyping may work in practice consider first a conventional program development, for example a payroll system. The development of such system would involve a set of inputs: namely, employee number, pay rate, ever time rate, hours worked and so on. The outputs would typically be gross pay by multiplying hours worked by rate, and so on. Hence, the requirements for such a system can be clearly specified. A stage based approach may be appropriate.

The first prototype gives an indication of the likely look and feel of the end-product. It may well serve as a satisfactory basis for future development, in which case further refinements can proceed. Further refinements may involve adding manageable chunks to the existing prototype. This type of prototyping is called incremental prototyping since the remainder of the development will proceed with incremental advances on the first prototype. The iterative development life cycle is shown in Figure 5.



**Figure 5:** Incremental Expert System Prototyping

The prototype may also turn out to be unsatisfactory, in which case it might be discarded. This type of prototyping is called throwaway prototype [4]. There are some advantages of rapid prototyping:

- It gives project developers a clear idea of whether it is feasible to attempt to tackle the complete application using expert system technology.
- It provides a useful product for discussion at an early stage.
- It attracts management or potential project sponsors with scope for quick demonstration.
- It provides a means for experts to criticize the system or make amendments or exceptions to the rules.
- It helps sustain both the expert's and users' interest.
- It gives insight into effectiveness of the intended tools to be used and the intended knowledge representation formalism, and provides scope to make such changes if necessary at an early stage.

- It may disclose unforeseen shortcomings in the proposed systems that were unidentified during the feasibility stage.
- It might trigger suitability for expert systems in some relate domain.

#### 2.1.4.2 The Feasibility Study

The feasibility of an expert system has to be assessed against several criteria. However, a feasibility assessment strategy is necessary. One possible strategy derives a list of issues and then assigns each issue a weight that reflects its relative importance. A score is then assigned to each issue. Table 2 shows that as an example, a partial list of issues that may be considered. The total sum of these weights is calculated and used to ascribe a percentage value to the likely success of the expert system project.

**Table 2:** Feasibility Assessment Issues

Weight	Issue
2	Expert Knowledge Required
2	Problem is well defined
1	Problem requires uncertainty knowledge
2	Problem domain is well documented
7	Feasibility % = total score / total points

The three main general criteria to consider in feasibility study are;

- Cost
- Appropriateness
- Availability of expertise

#### Costs

A whole range of costs must be considered at the outset of an expert system development. Resources must be identified for analysis, design coding, implementation and maintenance, in addition to hardware and software costs. Costs of sources of knowledge must also be known, as well as the likely time spent for user during the design and analysis phases and for training in the use of the operational system.

#### Appropriateness

Expert systems have been successful in problems that require a heuristic approach to problem solving. However, expert systems are not suitable for all types of problem. Indeed, experience has shown that not all heuristic problems are amenable to expert

system solution. If a domain expert can solve the problem via a telephone conversation with the end-user, an expert system can probably be developed to solve the problem. The rationale underlying this test follows from the fact that the expert will have access to no additional information emanating from other sources, and the user will be able to describe the problem verbally. Conversely, if the user is unable to describe the problem verbally, or if the expert is unable, based on the telephone dialogue, to conclude a reasonable solution, then development of an expert system will be likely to fail. Ideally, for the telephone test to succeed, the time taken to solve the problem by a human expert should not exceed 30 minutes.

### **Availability of expertise**

Expert must be identified for the domain, and must be available to devote sufficient time to the project. Moreover, the expert must be co-operative and enthusiastic; otherwise the project will have little chance of succeeding. When the project has been identified and authorized the process of knowledge engineering can begin.

#### **2.1.4.4 Maintenance of Expert System**

Few expert systems will remain static for very long. For instance, a taxation expert system is likely to require annual changes to reflect budget changes. Similarly, a printer manufacturer using an expert system help desk to assist customer problems would carry out updating as a result of addition of new printers or removal of printers no longer supported by the company, and so on. Changes in company organization, or political, economic or cultural changes, need updating if the expert systems are to remain effective. Yet many expert systems are not maintained adequately.

The impetus for changes to an expert system can come from several sources. The expert, for example, may want to change the system to reflect changes in his or her knowledge about the domain. The user may also want to make changes, perhaps to modify the system to make rules work in a particular context, or make some other change to suit his or her needs. Changes could also be necessary as a result of alteration in company procedure. Other changes may be necessary as a result of upgrading the operating system to a newer version. These will possibly mean changes to the knowledge base. The knowledge engineer may, from time to time, need to make changes to link the system to external interfaces, such as databases or to the rule base to eliminate redundant rules etc. [4, 8]

There are two main factors that will influence the maintainability of an expert system:

- **Understandability:** if code can be easily understood it can be more easily maintained by maintenance engineers who were not members of the original development team.
- **Changeability:** This is the ease with which the expert system can be extended and changes incorporated.

### **2.1.5. Expert System Shells**

Shells provide an easy starting point for building an expert system because of their ease of use. They are expert systems that have been emptied of their rules. This means that developers can concentrate on entering the knowledge base without having to build everything, including the interface engine and user interface, from scratch. Even non-programming experts can familiarize themselves with shells fairly rapidly. Also, many expert system shells contain facilities that can simplify knowledge acquisition. Non-programming experts can acquire an understanding of shells without undertaking the lengthy learning process that programming other types of software development requires.

However, using a shell to build an expert system can seduce the builder into oversimplifying the application domain because shells are inflexible in that it is difficult to modify or change the way they work with regard to both representation of knowledge and the inference mechanism. It is therefore important not to let the shell dictate the representation of the domain, for the result will be reflected in the performance of the system [9].

## **2.2 Jess Expert System Shell**

### **2.2.1 General Information**

Jess is a rule engine and scripting environment written entirely in Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives access to all of Java's APIs.

Jess uses an enhanced version of the Rete algorithm to process rules. Rete is a very efficient mechanism for solving the difficult many-to-many matching problem. Jess has many unique features including backwards chaining and working memory queries, and of course Jess can directly manipulate and reason about Java objects. Jess is also powerful Java scripting environments, which create Java objects, call Java methods, and implement Java interfaces without compiling any Java code.

Jess is primarily intended as a library that can be embedded in other Java software. Jess comes complete with a simple command prompt. To run Jess as a standalone command-line application, execute the class "jess.Main" from the JAR file.

Programming with Jess occurs in two different but overlapping ways. First, Jess is used as a rule engine. A rule-based program can have hundreds or even thousands of rules, and Jess will continually apply them to the data. Often the rules represent the heuristic knowledge of a human expert in some domain, and the knowledge base represents the state of an evolving. In this case, the rules are said to constitute an expert system.

The Jess language is also a general-purpose programming language, and it can directly access all Java classes and libraries. For this reason, Jess is also frequently used as a

dynamic scripting or rapid application development environment. Whereas Java code generally must be compiled before it can be run, Jess interprets code and executes it immediately upon being typed. This allows experimenting with Java APIs interactively and build up large programs incrementally. It is also easy to extend the Jess language with new commands written in Java or in Jess itself, so the Jess language can be customized for specific applications. Jess is therefore useful in a wide range of situations [10, 11].

Given its flexibility, Jess can be used in command-line applications, GUI applications, servlets, and applets. Jess applications (with or without GUIs) are developed without compiling a single line of Java code and also writing Jess applications that are controlled entirely by Java code, with a minimum of Jess language code. Jess has been deployed in everything from enterprise applications using J2EE on mainframes to personal productivity applications on handheld devices. The most important step in developing a Jess application is to choose architecture from among the almost limitless range of possibilities. This choice should be done early in the development of the application. One way to organize the possibilities is to list them in increasing order of the amount of Java programming involved:

1. Pure Jess language, with no Java code.
2. Pure Jess language, but the program accesses Java APIs.
3. Mostly Jess language code, but with some custom Java code in the form of new Jess commands written in Java.
4. Half Jess language code, with a substantial amount of Java code providing custom commands and APIs. Jess provides the main () function (the entry point for the program).
5. Half Jess language code, with a substantial amount of Java code providing custom commands and APIs.
6. Mostly Java code, which loads Jess language code at runtime.
7. All Java code, which manipulates Jess entirely through its Java API.

### **2.2.2 Jess Performance**

Some people think that Java is slow. They're wrong. Modern Java virtual machines are extremely powerful and sophisticated. In many applications, Java is as fast as comparable C or C++ code. For Jess, being written in Java is not a liability. Jess is fast. Jess can plow through large piles of rules and facts in little time. Using Sun's Hotspot JVM on an 800 MHz Pentium III, Jess can fire more than 80,000 rules per second; it can perform almost 600,000 pattern- matching operations per second; it can add more than 100,000 facts to working memory per second; and a simple counting loop can do 400,000 iterations per second. Independent benchmarks have shown that Jess is significantly faster than many rule engines written in the faster C language. For example, on many problems, Jess outperforms CLIPS by a factor of 20 or more on the same hardware. Jess's rule engine uses an improved form of a well-known method called the Rete algorithm to match rules against the working memory. The Rete algorithm explicitly trades space for speed, so Jess can use a lot of memory. Jess does contain commands that sacrificing some performance to decrease memory usage. Nevertheless, Jess memory usage is not ridiculous, and fairly large programs will fit easily into Java's default heap size of 64 megabytes. Because Jess is a memory-

intensive application, its performance is sensitive to the behavior of the Java garbage collector. Recent Java Virtual Machines (JVM) from Sun features an advanced Java runtime called Hotspot, which includes a flexible, configurable garbage collection subsystem. The garbage collector is the part of the JVM that is responsible for finding and deleting unused objects. Although every Jess rule base is different, in general, Jess benefits with adjusting two parameters: the heap size and the object nursery size [11].

## **2.3 Applications and the Future**

### **2.3.1 Applications of Expert System**

Expert systems have found much use in industrial, commercial and financial application domains. Indeed, the range of expert system application areas is now diverse that they have been successful almost wherever human decision support is involved. Most application fall into the following task categories:

- **Diagnostic systems:** These infer systems malfunctions from observable situations, for example medical, engineering and software diagnosis.
- **Planning and scheduling systems:** These are systems that design actions, for example automatic programming, robot movement, military strategy and even timetabling.
- **Interpretation systems:** These are systems that infer descriptions from observables, for example surveillance systems or speech understanding systems.
- **Prediction systems:** These are systems that infer consequences from given situations; examples are traffic prediction or weather forecasting.

The purpose of an expert system is not only just to capture domain expertise, but to simulate a particular problem-solving task carried out within a domain.

### **2.3.2 Emerging Applications Expert Systems**

Today new application areas are emerging that lend themselves well to expert systems. They include; knowledge publishing, internet tools, configuration and intelligent front-ends.

#### **Knowledge Publishing**

Knowledge publishing is a growing application area of expert systems. The idea of knowledge publishing is encapsulated in the concept of a book. A book is passive object in that the onus is on the reader to find the passage of interest. Knowledge



publishing delivers knowledge to the user actively by providing what the user specifically requested. There are examples in common use that are disguised: that is, working within other systems. End-users may be unaware that they are using an expert system.

## **Intelligent Interfaces**

Intelligent interfaces offer the potential to automate many tasks that could normally only be performed by human user. Many computer users experience difficulty in accessing information required in computer systems. In an age of information overloading, users waste much time trying to find information.

## **Expert Systems and World Wide Web**

Intelligent interfacing covers a wide range of applications, and offers the potential of systems that deal with natural language, computer sound and vision, automated reasoning. Intelligent agents can launch themselves without any distraction to the user or, indeed, without the user even being aware of their existence. Practical examples exist in a variety of areas but accessing information is a very common application area for intelligent interfaces. Expert systems have been developed to assist users in accessing large, specific databases on the web. Typically, these expert systems would ask users the most relevant questions to elicit and identify the user problem. The expert system should then direct them to the information that offers the correct solution to the problem.

### **2.3.3 The Future of Expert Systems**

Many new application areas continue to surface. This interest has led to demand for better knowledge engineering tools that enable less experienced builders to develop systems. Advances in computer hardware have enabled expert systems to be built using a more sophisticated graphical user interface operating environment. Improvements in areas of expert system technology, such as knowledge representation, knowledge acquisition, development tools, expert system design, and the programming of expert systems, continue to be made.

## **User Interface Improvements**

Improvements in the quality of the user interface have been notable in recent years in both the content of the communication and the physical means of communication between the user and the expert system. In addition to high quality graphical user interfaces, many expert systems provide capabilities for natural language interaction, speech recognition and high-quality explanations. Improvements in this area are likely to continue in the future.

## **Knowledge Representation**

Tools that enable the integration of these schemas will continue to improve in the future. This will enable the knowledge engineer to combine models of domain knowledge, which provide a more realistic representation of the domain.

## **Knowledge Acquisition**

The knowledge acquisition stage of expert system development is well known for precipitating the bottleneck phase. Manual methods of knowledge acquisition often require interviews and lengthy verbal analysis. This is time consuming and expensive. The process of automating knowledge acquisition is therefore an issue of extreme importance.

## CHAPTER 3

### INTELLIGENT SYSTEM DEVELOPMENT WITH MOBILE DEVICES

#### 3.1 Mobile Device World

##### 3.1.1 Mobile Commerce

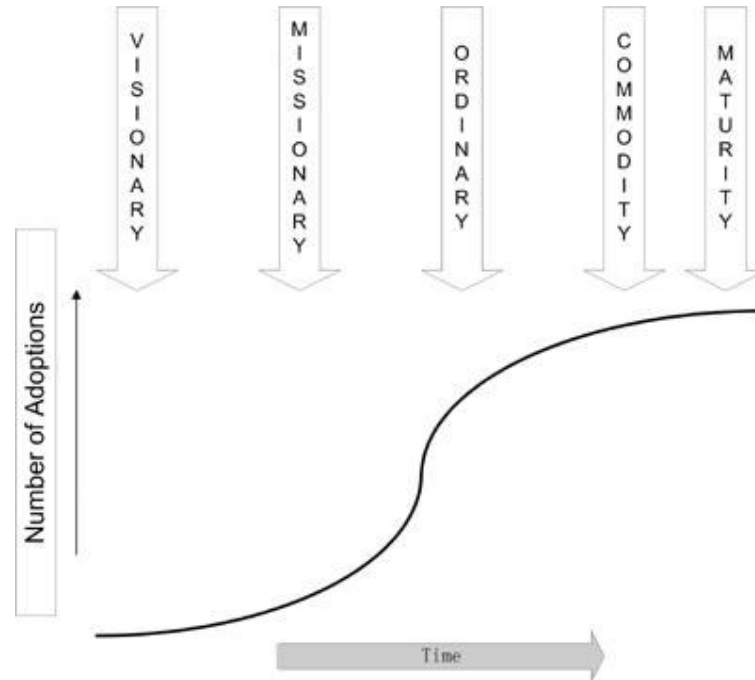
Moore's law states that computer capabilities double every 18 months. Meanwhile, prices for key computer components drop by halves [12]. When cheap personal computers became powerful enough for most common tasks and graphic user interfaces, they were massively adopted by average home users and enterprises. Together with Internet technology, the personal computer is widely credited as a key enabler of the electronic commerce New Economy revolution of the 1990s.

Moore's law still holds. Today, computer devices are everywhere in our lives. Those small and cheap devices have processing power beyond the high-end PCs of only a few years ago. When coupled with mobile communication devices and the mobile Internet, new-generation pervasive devices empower us to access information anywhere, anytime.

Unlike the PC-centric electronic commerce, mobile commerce is focused on personal experiences. A person carries a pervasive mobile device and gets information anytime, anywhere, from anyone. For the first time in history, a person's information access can be disassociated from her environment. For example, a traveler does not need to be in her office at a specific time to get ticket information. That unprecedented freedom of information could fundamentally affect all business categories [13].

### 3.1.2 Mobile Technology Adoption

To take advantage of mobile commerce, businesses and consumers must adopt state-of-the-art mobile technology. The diffusion of innovations usually goes through five stages: visionary, missionary, ordinary, commodity, and maturity. The characteristics of each stage are listed below in Figure 6.



**Figure 6:** Technology Adoption Curve

- **Visionary:** The technology has just come out. Few people see its business value. The technology proponents in the visionary stage base their arguments on advocacy rather than on solid value propositions.
- **Missionary:** Business practitioners start to see the value of the innovation. Pioneer companies or employees become early adopters of the new technology and start to profit from it.
- **Ordinary:** The value of the technology is well accepted by the main stream business executives, and most companies have developed plans to implement solutions based on the new technology. In this stage, the developers and enablers of the new technology make the bulk of profits.
- **Commodity:** In this stage, the adoption of the technology becomes common practice. The technology has started to generate profits industry wide. However, since implementations have been standardized, the barrier of market entry becomes substantially lower, which results in intense competition in the enablers sector.

- **Maturity:** In a mature market, most commodity technology suppliers are consolidated to a few dominant players.

On a typical innovation diffusion curve, the transition period between missionary and ordinary stages are associated with explosive growth of adoptions and a limited number of technology firms who have the expertise to implement viable solutions. The unbalanced demand and supply creates golden opportunities for developers and technology firms to make money. We have seen this pattern repeated throughout history.

The value of mobile commerce has been well accepted by industry leaders and business executives. Leading companies in financial services, information services, transportation, and manufacturing sectors have already started to implement their mobile commerce strategies. Other companies will soon follow suit. All these signs indicate that mobile commerce is currently at late missionary stage and is moving toward the ordinary stage.

Historically, technology adoption was never a smooth or linear process. As it has seen in the recent rise and fall of dotcoms, the expectations of technology adoption are often exaggerated; the relationships between the new and old business models are often distorted. Those unrealistic expectations have resulted in severe consequences for those failed dotcom companies and their employees. Nevertheless, adoption of e-commerce as a whole is steadily moving on [13, 14].

### **3.1.3 Mobile Device Manufacturers**

There is a wide range of mobile devices, including many kinds of cell phones, PDAs, consoles, and auto-mounted devices. Since mobile devices are to become pervasive personal belongings, they pose some unique design and technical challenges to manufacture. Successful mobile devices should have the following features:

- Small size
- Rich multimedia presentation capabilities
- Fast response time
- Large memory for data and applications
- Long battery life
- Fashionable and personalizable

Billions of dollars have been invested in mobile hardware research by leading companies such as Intel, Nokia, Motorola; they have produced many competing chipsets and handset designs [14].

### **3.1.4 Mobile Software Platform Providers**

Given the diversity of mobile hardware, there are many mobile device operation systems. Examples include PalmOS, Symbian OS, Windows CE, and Embedded Linux. The Operating System SDKs (Software Developer Kits) often lack advanced programming language support and important libraries for business functions.

#### **Symbian OS**

Symbian OS is an operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of common tools, produced by Symbian Ltd. Symbian is currently owned by Ericsson (15.6%), Nokia (47.9%), Panasonic (10.5%), Samsung (4.5%), Siemens AG (8.4%), and Sony Ericsson (13.1%). Symbian OS's major advantage is the fact that it was built for handheld devices, with limited resources, that may be running for months or years. There is a strong emphasis on conserving memory, using Symbian-specific programming idioms such as descriptors and a cleanup stack. Together with other techniques, these keep memory usage low and memory leaks rare. There are similar techniques for conserving disk. Furthermore, all Symbian OS programming is event-based, and the CPU is switched off when applications are not directly dealing with an event. This is achieved through a programming idiom called active objects. Correct use of these techniques helps ensure longer battery life [15].

#### **Windows Mobile**

Windows Mobile is a compact operating system combined with a suite of basic applications for mobile devices based on the Microsoft Win32 API. Devices which run Windows Mobile include Pocket PCs, Smartphones, and Portable Media Centers. It is designed to be somewhat similar to desktop versions of Windows [16].

#### **Palm OS**

Palm OS is a compact operating system developed and licensed by PalmSource, Inc. for personal digital assistants (PDAs) manufactured by various licensees. It is designed to be easy-to-use and similar to desktop operating systems such as Microsoft Windows. Palm OS is combined with a suite of basic applications including an address book, clock, note pad, sync, memo viewer and security software. Palm OS was originally released in 1996. Palm OS applications are primarily coded in C/C++. A Java Run time Environment is also available for the Palm OS platform, however it isn't shipped as standard on handhelds, and has to be obtained separately, as a result it is less popular with developers in general [17].

So, on top of operating systems, there are also application software platforms. Those platforms run on a variety of mobile devices and provide advanced sets of development tools and features. Examples of such platforms include WAP micro browsers, Java, and Microsoft .NET Compact Framework (Table 3). Java mobile application platform is one of the major focuses of this thesis.

**Table 3:** Operating Systems and Programming Languages

<b>Programming Language</b>	<b>Operating System</b>	<b>Example Device</b>
Java	Win32 WinCE Linux	Win 32 Desktop/ Laptop PCs HP Ipaq
J2ME	Symbian OS	Nokia 7650 Smart Phone Siemens M50
C++	Win32 Symbian OS Palm OS	Win 32 Desktop/ Laptop PCs PDAs from Palm

### 3.2 Smart Mobile Device

Intelligent Systems use an approach to problem solving which is different to conventional data and information processing algorithms. Intelligent Systems rely on knowledge and inference to solve complex problems, and have been applied to a huge range of problems in many domains and markets. Intelligent Systems effectively make specialist knowledge available for use by non-specialist users.

Intelligent Systems are commonly thought to require powerful computing environments with powerful processors and a large amount of memory in order to function effectively. In fact, this is only true during the development phase of Intelligent System implementation. Although powerful tools are required in order to construct Intelligent Systems, once they are built they can be deployed on much less powerful devices.

The requirement for a powerful computing environment means that mobile devices, which are typically limited in terms of memory and processing power are not the obvious candidates as target devices for Intelligent Systems. However, the requirement for a powerful computing environment for intelligent systems mostly applies to the development stage of the system's lifecycle. Once an intelligent system has been implemented and tested, all that is required is the knowledge base and the inference engine. Sophisticated tools for constructing, modifying and testing the system are no longer required; all that is required is that the system can be executed.

Mobile and wireless technologies are being used to increase revenue and decrease expenses across a wide range of functional areas including: invoice processing; field based workflows; and route management. Many of these lead to increasing customer satisfaction, and higher productivity. With mobile technologies changing rapidly and corporations constantly evolving business practices and plans, it's critical to have mobile technology solutions that go beyond simple flexibility and are capable of embracing change [12, 13, 18].

### 3.3 Application Development over Mobile Device with Java

#### 3.3.1 Why Java?

All major mobile device vendors, including Nokia, Motorola, Siemens, Samsung, Fujitsu, LG Electronics, Mitsubishi, NEC, Panasonic, Psion, RIM, Sharp, and Sony, have adopted Java as part of their core strategy for future smart devices. Major wireless carriers such as NexTel, SprintPCS, and AT&T have committed to support Java devices and applications on their networks.

Before Java was called Java, it was Oak—a programming language designed for TV set-top boxes and other devices. Considering its deep roots in devices, there is no surprise that many philosophies and designs behind the Java technology are perfectly suited for mobile applications [19]. The advantages of the Java technology are as follows.

- **Cross platform:** This is very important in the diversified mobile device market. In a heterogeneous enterprise environment, the ability to develop and maintain a single client for all devices results in huge savings.
- **Robust:** Since Java applications are completely managed, the byte code is verified before execution, and memory leaks are reclaimed by garbage collectors. Even if a Java application does crash, it is contained within the virtual machine. It will not affect other sensitive applications or data on the device.
- **Secure:** The Java runtime provides advanced security features through a domain-based security manager and standard security APIs.
- **Object oriented:** The Java language is a well-designed, object-oriented language with vast library support. There is a vast pool of existing Java developers.
- **Wide adoption at the back end:** It is relatively easy to make Java clients work with Java application servers and messaging servers. Due to the wide adoption of Java 2 Enterprise Edition (J2EE) on the server side, mobile Java is the leading candidate for enterprise front end applications.



However, technical merits are not the only factors that determine the success of a technology. The business values are just as important. In particular, the vendor's ability to address the concerns of the developer and user community key to the technology's adoption [20].

The concept of open interfaces is core to the Java technology. It works as follows: For a given computing task, a set of standard APIs is defined by a standards committee. Individual vendors then provide competing libraries that implement those APIs. The application code using the API is completely decoupled from the specific implementation provider. That approach minimizes the developer's learning cost and improves code portability. Yet, it also protects the freedom of choosing vendors. The Java Community Process (JCP) is an effort to develop standard Java API specifications. JCP Executive Committees (EC) consists of industry leading companies. Anyone in the general public can submit a new Java Specification Request (JSR) for a new API. The appropriate EC decides whether to accept this new JSR. Once approved, the JSR lead can recruit more companies or individuals to develop the API specification together. Every specification goes through multiple stages of community and public reviews before it becomes an official Java standard. The JCP ensures that all interested parties can express their concerns. As a result, the final APIs are supported by industry consensus. All standard mobile Java APIs are developed democratically through the JCP [19, 21].

Although cross-platform compatibility is a key concept behind the Java philosophy, Java designers also realize that portability has its limits. Portability is only meaningful among similar operating systems and hardware platforms. Today, the Java platform is partitioned into three editions, all of which have significant roles in mobility.

- **The Java 2 Standard Edition (J2SE)** which is the basis of the Java platform. It defines the Java Virtual Machines and libraries that run on standard PCs and workstations. In the mobility world, J2SE is the ideal choice for wireless laptop-based applications.
- **The Java 2 Enterprise Edition (J2EE)** which is the J2SE plus a large number of server side APIs. It aims to implement complex application servers. J2EE application servers can drive browser-based (e.g., WML and XHTML) mobile applications and become service end-points for smart mobile clients.
- **The Java 2 Micro Edition (J2ME)** which is a Java platform that is designed for small devices. It contains specially designed, lightweight virtual machines; a bare minimum of core-class libraries; and lightweight substitutes for standard Java libraries. J2ME is the ideal mobile client platform for wireless PDAs and enhanced mobile phones.

Applications for each of the Java editions can follow similar architectures. That allows companies to leverage existing developer talent, cut cost and build more maintainable products.

### 3.3.2 Java Micro Edition (J2ME) Architecture

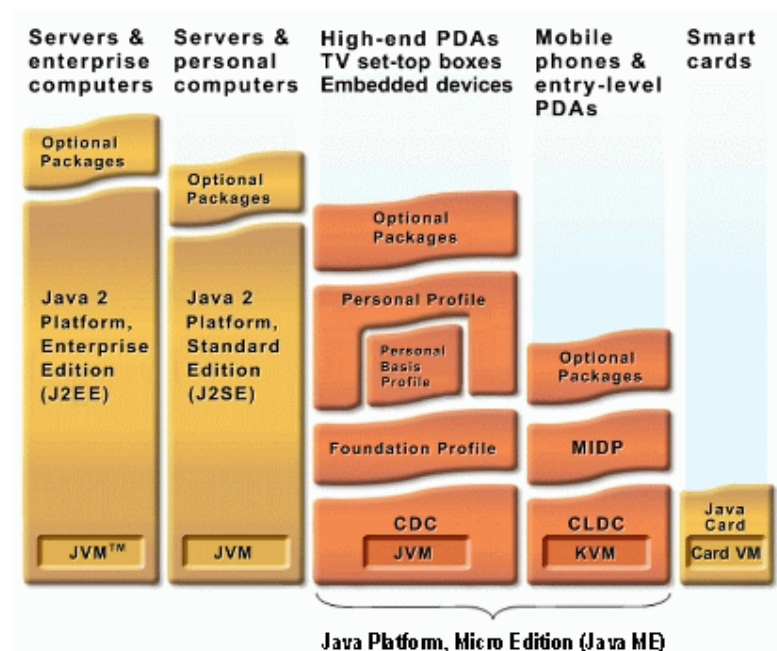
Java Micro Edition technology was originally created in order to deal with the constraints associated with building applications for small devices. For this purpose Sun defined the basics for J2ME technology to fit such a limited environment and make it possible to create Java applications running on small devices with limited memory, display and power capacity.

J2ME platform is a collection of technologies and specifications that can be combined to construct a complete Java runtime environment specifically to fit the requirements of a particular device or market. This offers a flexibility and co-existence for all the players in the eco-system to seamlessly cooperate to offer the most appealing experience for the end-user.

To balance portability with performance and feasibility in the real world, J2ME contains several components known as configurations, profiles, and optional packages. Each valid combination of a configuration and a profile targets a specific kind of device.

- The configuration provides the most basic and generic language functionalities.
- The profile sit on top of configurations and support more advanced APIs, such as a graphical user interface (GUI), persistent storage, security, and network connectivity.
- The optional packages can be bundled with standard profiles to support specific application needs. The two most important J2ME configurations are as follows.

The Figure 7 represents an overview of the components of J2ME technology and how it relates to the other Java Technologies [20, 21].



**Figure 7:** Overview of J2ME

Over time the J2ME platform has been divided into two base configurations, one to fit small mobile devices and one to be targeted towards more capable mobile devices like smart-phones and set top boxes. The Connected Limited Device Configuration (CLDC) is for the smallest wireless devices with 160 KB or more memory and slow 16/32-bit processors. The CLDC has limited math, string, and I/O functionalities and lacks features such as the JNI (Java Native Interface) and custom class loaders. Only a small subset of J2SE core libraries is supported by the CLDC virtual machines (Kilobyte Virtual Machines) [22].

The Connected Device Configuration (CDC) is for more capable wireless devices with at least 2 MB of memory and 32-bit processors. Unlike the CLDC, the CDC supports a fully featured Java 2 VM and therefore can take advantage of most J2SE libraries [23].

The most important and successful J2ME profile is the Mobile Information Device Profile (MIDP), based on the CLDC. The MIDP targets the smallest devices, such as cell phones. It is already deployed on millions of handsets. However, the MIDP v1.0 lacks some important features, such as security and advanced UI controls. As a result, device vendors often supply their own MIDP extensions to provide advanced custom features. Vendor-specific extensions undermine the portability of J2ME applications. Many problems with the MIDP v1.0 have been fixed in the MIDP v2.0.

With the configuration and profiles the actual application then resides, using the different available APIs in the profile. For a CLDC and MIDP environment, which is typically what most mobile devices today are implemented with, a MIDlet is then created. A MIDlet is the application created by a J2ME software developer, such as a game, a business application or other mobile features. These MIDlets can be written once and run on every available device conforming to the specifications for J2ME technology. The MIDlet can reside on a repository somewhere in the ecosystem and

the end user can search for a specific type of application and having it downloaded over the air to device.

The design and set up of the J2ME platform, covering everything from small limited devices with intermittent network connection, to capable on-line mobile devices, the design of the platform makes it flexible and capable to efficiently support the need for services covering all channels for mobility. The basic design makes services easily portable between the different configurations and profiles and the ambition to allow the same service being delivered via different channels can be achieved [20, 24, 25].

### 3.3.3 Competing Technologies

J2ME is not the only technology that enables mobile commerce. Leading competing technologies are as follows [12]:

- **WAP/WML:** WAP/WML is a platform for thin client applications (i.e. micro browser-based applications). The thin client paradigm is completely different from the smart client paradigm enabled by the J2ME. J2ME smart clients are likely to replace WAP/XML applications in the future.
- **BREW:** Qualcomm's Binary Runtime Environment for Wireless (BREW) is a technology that supports rich client development and provisioning. BREW applications are written in C/C++ and run natively on BREW-enabled phones. However, only a limited number of phones support BREW. Although BREW native applications can be heavily optimized, they are not executed in managed environments and therefore are prone to programming errors. J2ME applications could run on BREW devices through a J2ME runtime for BREW (e.g., the BREW MIDP runtimes from IBM and Insignia).
- **.NET Compact Framework (.NET CF):** Microsoft's .NET CF is the closest competition to the J2ME. Like J2ME, it targets smart-managed mobile clients development. It has a strong focus on enterprise applications. However, the .NET CF runs only on high-end Windows CE and Pocket PC devices.

## 3.4 Developing End-to-End Systems

### 3.4.1 SOAP and Web Services

SOAP is a language and platform-independent protocol for communication between applications. It is based on XML and is designed for communications via Internet, more accurately over HTTP. SOAP 1.2 has been published as an official standard by the World Wide Web Consortium. It is widely used today, especially as the payload protocol for Web Services.

Since SOAP is typically run over HTTP (or HTTPS), messages are normally not blocked by firewalls. This is not the case with Remote Procedure Calling(RPC), Common Object Request Broker Architecture(CORBA), and many other middleware technologies that have their proprietary communication mechanisms. By using SOAP messages an application can transfer textual and binary data and make method invocations, depending on the service the application is connecting to. SOAP defines a messaging envelope that contains a header part and a body part. The header is optional and can contain header blocks for control information, such as for addressing, security, and reliable messaging purposes. The body can contain one or more XML blocks, text, or no content at all. SOAP supports sending binary data as encoded text (such as Base64-encoded text) in the body, or as an attachment utilizing a Multipart-MIME structure. The structure of a SOAP message with a binary attachment is shown Appendix A.

The SOAP message contains a lot of overhead. A simple method invocation that in binary format takes maybe 100 bytes per message, can take around 1000 bytes per message when SOAP is used. Unfortunately this is the unavoidable cost of being interoperable across applications, platforms, and programming languages. SOAP specification itself does not contain any means to handle, for example, secure message exchange by design. However, SOAP is extensible and this and other kinds of properties can be added to it as extensions. Extensions are components that can inspect or modify SOAP messages before they are sent and upon reception.

SOAP is the most widely used protocol for XML-based object serialization. It is the technology of choice for future ubiquitous Web Services. Compared with competing technologies, SOAP has the following advantages:

- **Strong type support:** SOAP defines more than 40 standard data types through XML Schema and allows users to custom-define complex data types. Such sophisticated data-type support makes SOAP a powerful and rich language for exchanging information among today's widely deployed object-oriented systems.
- **Flexible and ubiquitous messaging:** In addition to strong data-type support, SOAP also supports various messaging schemes. Those schemes include synchronous RPC, asynchronous messaging, multicast messaging and complex message routes with multiple intermediaries.
- **Standardization:** Since SOAP has gained mainstream support as a Web Services messaging standard, most other Web Services protocols must interoperate or bind with SOAP. For example, WSDL (Web Services Description Language), UDDI (Universal Description, Discovery, and Integration), and most XML registries support SOAP; XML Digital Signature, XML Encryption, SAML (Security Assertion Markup Language), and other secure XML protocols all provide standard binding with SOAP. Each binding protocol provides syntax of its own special element inside SOAP messages. SOAP's full support for XML namespaces has made it easy to bind with other protocols.

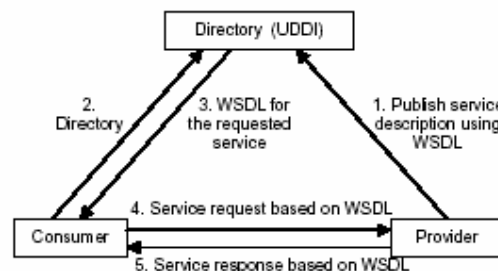
The World Wide Web Consortium defines a Web Service as a software system designed to support interoperable machine-to-machine interaction over a network. Web services are frequently just application programming interfaces (API) that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. Web service definition encompasses many different systems, but in common usage the term refers to those services that use SOAP-formatted XML envelopes and have their interfaces described by Web Service Definition Language (WSDL).

Web services have one big advantage that they are truly interoperable across vendors, platforms, and programming languages. This stems from the fact that they use XML-based SOAP as a payload protocol and universally accepted HTTP as a transport protocol. SOAP being a purely textual format is usable in any environment. HTTP has the advantage of typically having no trouble navigating through firewalls.

Unfortunately Web services' biggest advantage is also their biggest disadvantage especially SOAP adds a considerable amount of overhead to the messages sent across a network. In addition to HTTP being a fairly slow transport protocol, parsing the SOAP messages takes also a considerable amount of time. Thus Web services are slower than most of the other technologies and not very suitable for highperformance systems. Problems arise also, for example, on how to send binary data using a text-based payload protocol. Furthermore, Web services are inherently stateless, that is, a new object is created to service each request sent by the client. Callbacks are not directly supported at all.

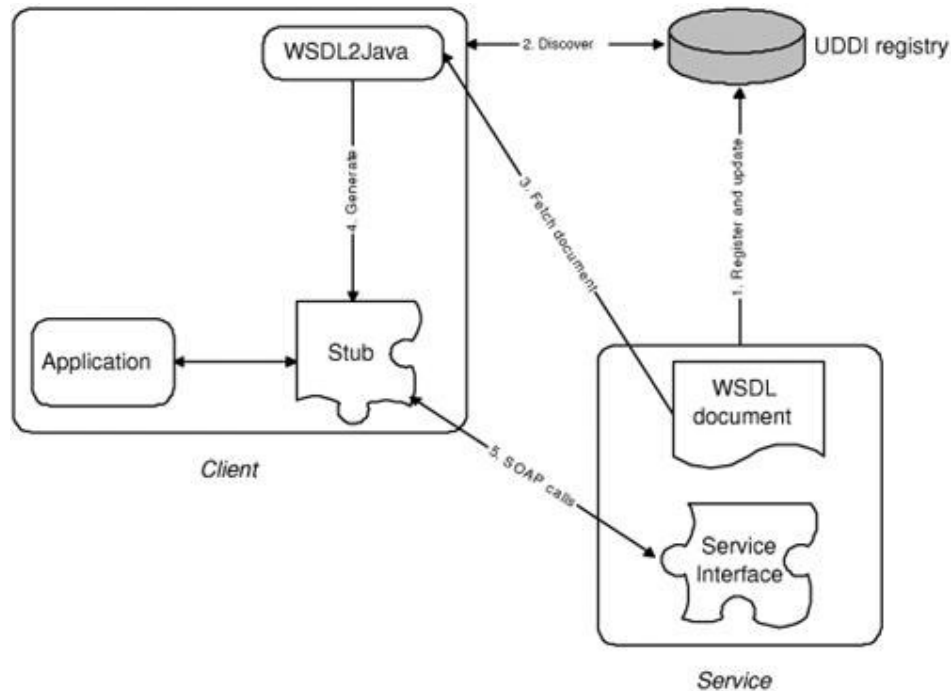
Web services can be discovered and invoked by several different applications alike, as well as by other Web services. Using Web services, businesses can share data among partners and across legacy applications.

In Web services, clients are called consumers and server applications providers of services. Providers publish their Web services described in Web Services Description Language by placing the WSDL documents into a special directory. The directories, in turn, use the Universal Description, Discovery and Integration (UDDI) technology that allows the directories to be searched for a particular Web Service. The basic usage of Web services is shown in Figure 8. The communication along each arrow in the figure is conducted using SOAP messages.



**Figure 8:** Basic Usage of Web Services

As an infrastructure solution, Web Services is touted as self-contained, automatically discovered, and automatically configured reusable software components. Web Services is much more than SOAP which only serves to provide a platform-independent transport layer. Figure 9 illustrates the overall architecture of Web Services.



**Figure 9:** Web Service Architecture

Each Web Service makes a description of its service available as a WSDL document. The WSDL describes technical details on how to access the service. Authorized remote clients can download the WSDL file and generate a stub that matches the SOAP service interface. Any RPC method in a Java stub can be called from client side Java applications as if it were a local method. All leading Web Services toolkits provide WSDL-to-stub code generators. Web Services register themselves with central registry databases such as the UDDI registry. The client searches the UDDI, finds out the service it needs, fetches the WSDL file, generates the stub, and starts calling remote methods [14, 20, 25, 26].

### 3.4.2 kSOAP

To build Web Services clients on J2ME devices, J2ME-compatible SOAP parser is needed. Most standard Java SOAP libraries (such as the Apache Axis and Java Web SDK) are too heavy for small devices. The open source kSOAP project runs on all J2SE and J2ME platforms, including the MIDP. Built on top of the kXML parser, the entire kSOAP library is only 42 KB. However, as a trade-off for the lightness, kSOAP does not support the entire SOAP specification. It supports the most commonly used

SOAP features and is sufficient for most Web Services that are currently available. Currently, almost all major kSOAP applications and tools are based on kSOAP release v1.2, which supports a core subset of SOAP 1.2 features. Every generic XML parser with namespace support understands SOAP messages and can extract information from them. In theory, it can always extract text information from a SOAP document using a generic XML parser and then convert those text strings to Java data objects. For example, the statement

```
int i = Integer.parseInt("123");
```

converts a text string "123" to an integer value 123. But such manual conversion burdens application programmers. Extracting Java data objects directly from a SOAP document provides a better approach.

A SOAP parser is built on a generic XML parser with special type-mapping and text data-marshaling mechanisms. A SOAP parser understands the data-type information in SOAP messages and automatically converts the SOAP elements to Java data objects. The parser's real value is that it makes SOAP serialization and deserialization and the entire wire protocol-transparent to object-oriented developers. The programmer just feeds Java objects into a SOAP writer, sends the message, waits for the server response, and then reads Java objects directly from the SOAP parser.

Using kSOAP to make simple RPC calls is very easy. The basic steps are the following:

1. Prepare the arguments to pass to the remote method. Instantiate a SoapObject object and add call arguments using the addProperty() method.
2. Prepare the call transport. Instantiate a HttpTransport object with the URL to the SOAP interface.
3. Make the remote method call. Pass the assembled SoapObject object (from step 1) to the HttpTransport object's call() method. The return value from the remote service is available as the return value of the call() method.

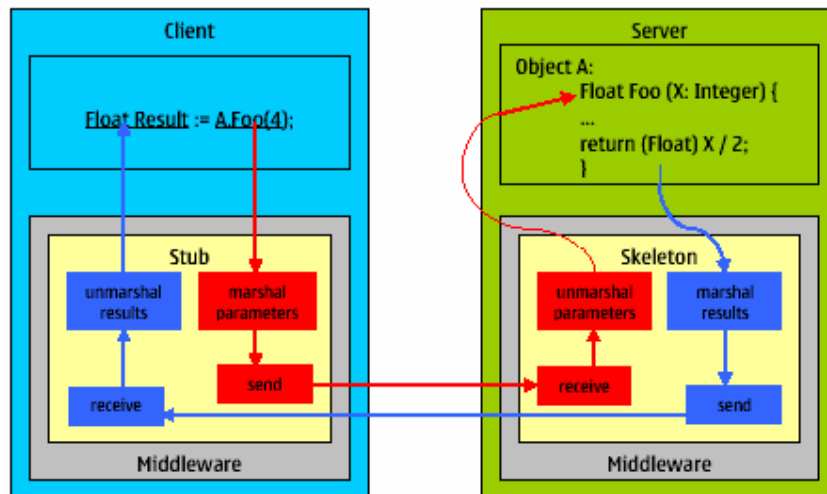
It is a difficult to sort through a long WSDL document and figure out the exact SOAP interface. For complex services, automatically generated client stubs from WSDL files prove useful. For example, both Apache Axis and Sun Java Web Services Developer Pack offer WSDL2Java tools. However, kSOAP is a small footprint library, not a complete Web Services toolkit. It lacks the tools to automatically generate client stubs. Fortunately, several J2ME IDEs provide such tools. kSOAP is adopted by IBM Web Sphere Studio Device Developer (WSDD, a J2ME IDE based on Eclipse), Sun ONE Studio and CodeWarrior Wireless Studio as their default mobile Web Services client library. All of them offer GUI wizards that ask the URL to the WSDL file and automatically generate the stubs into the current project source directory [12, 18, 27, 28, 29].



### 3.4.3 Remote Procedure Call (RPC)

Remote Procedure Call (RPC) is a technology that extends the conventional method calling mechanism to cover methods in other address spaces. In effect, this means making method calls to services that run on different processes or computers. With RPC, the application needs to know on which computer and on which port the method to be called is located at. Whether the method is local or remote, and which transport mechanism is used, are totally transparent from the calling application's point of view [30].

The basic idea of a remote method call using middleware is shown in Figure 10. In a typical scenario, the developer writes the server application with one or more classes that the client application should be able to instantiate and use. For each class, a separate interface definition also needs to be written by hand. The compiler uses this definition to create the client-side stub and server-side skeleton corresponding to the interface.



**Figure 10:** The Principle of Middleware

The client application cannot directly create or destroy objects of the class it wants to use; the client side stub represents the remote object to the client. The call goes through the stub, which merely marshals the parameters and passes the call to the server-side skeleton. The skeleton then unmarshals the parameters and makes the actual method call to the appropriate object. The return value is returned in the reverse order through the skeleton and the stub.

Although setting up a system like this might seem time-consuming and tedious at first, it usually is worth while if the application under development is not just a trivial one. The time spent when choosing a suitable middleware implementation and vendor will most likely be time saved later on in the development and maintenance phases. The structure of the application becomes simpler and application code is not overloaded

with details concerning the location of the server and the remote object. The client code is also not affected by the network, transport, and other communication details, such as the format of the messages passed between the client and the server.

Middleware in general provides several other benefits as well. Depending on the implementation, middleware can contain features for managing the CPU load and memory resources on the server. In many cases, it also provides various automatic services typically needed in distributed applications, such as naming, persistence, transaction, and connection security services. Now the application developer can concentrate on the logic of the application itself, rather than worrying about all the additional functionalities separately with every single application.

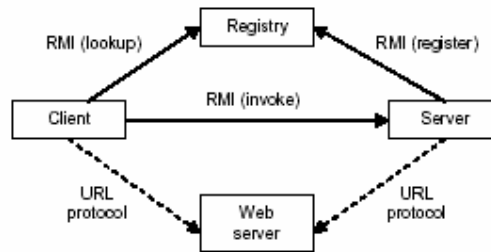
The downside of using commercial middleware is that it takes time and effort to learn to use all its features. And to really be able to make full use of all the capabilities, takes even longer. Furthermore, commercial products are rarely inexpensive (although there are exceptions to this, such as Java RMI), so in some cases it is not very logical to acquire such a product “just in case.” In the applications current and future needs should be determined very carefully before any decisions on purchasing a middleware product are made.

#### **3.4.3.1 Remote Method Invocation (RMI)**

Java Remote Method Invocation (Java RMI) technology, developed by Sun Microsystems, is the object oriented middleware equivalent of Remote Procedure Calls (RPC). In RPC, a client can only call procedures over a network, but in RMI the concept of classes and objects is strongly present. RMI bears many similarities to CORBA and can in many cases be used to solve similar programming problems.

The biggest difference between the two is that while CORBA is independent of the programming language, RMI works only with the Java programming language and cannot be used as a wrapper for legacy systems written in other programming languages. In RMI, client applications can use services provided by objects residing in another Java Virtual Machine using a remote reference to the remote object. The same remote object can also be used through a regular object reference by other objects in the same (server-side) JVM. The details of the communication between the client and the server are handled by RMI and the remote method call looks exactly like a standard Java method invocation, which makes RMI relatively easy to adapt.

The basic components of RMI are shown in Figure 11. The figure also shows which communication method is used between the entities. The server application registers its remote objects to the registry using RMI’s simple naming facility. The client application can then use the registry to locate the remote objects it needs to use. Remote references can also be passed or returned as part of the application’s normal operation.



**Figure 11: RMI Components**

If, for some reason, the class definition of the remote object is unknown to the client (or the calling client object is unknown to the remote object), RMI allows for the byte code and data of the unknown object to be loaded from an existing Web server. This is a very powerful feature, since it allows new types to be introduced in another JVM [20, 31].

### 3.4.3.2 Java Micro Edition RMI Optional Package (RMI OP)

The J2ME RMI Optional Package (RMI OP) is part of Java Micro Edition (J2ME) and is an implementation of JSR 066. This optional package allows remote method invocation and is based on Java Remote Method Invocation (RMI) technology found in Java Platform Standard Edition (J2SE).

RMI OP provides Java-to-Java remote method invocation for networked devices. It exposes distributed application protocols in terms of Java interfaces, classes and method invocations and shields the developer from the low-level details of network communications. The RMI OP reference implementation (RI) can be built with implementations of Connected Device Configuration/Foundation Profile (CDC/FP) based profiles such as CDC/FP 1.0.1, as well as with Personal Basis Profile (PBP) 1.0 when it is released [32].

## 3.5 An Application: Chest Pain Mobile Expert System (MES)

### 3.5.1 Introduction

MES is a rule-based system that helps a doctor determine what actions should be taken for a patient who comes to his office complaining of chest pain. For example, if the doctor feels that a patient is in danger of a heart attack, the doctor will send the patient to the emergency room (ER) by ambulance. If, on the other hand, the patient has symptoms suggesting esophageal disease, the doctor will evaluate and treat the patient. MES is programmed in Jess, a rule-based system that performs backward and forward chaining on rules.

MES is problematic because it prompts the user for lots of inputs. Chest pain diagnosis is a meaty domain with lots of factors to take into account. MES does not exhaustively handle all symptoms and/or causes of chest pain, but rather MES covers most of the common cases.

### **3.5.2 What does MES?**

MES determines between six possible actions to take on a patient who complains of chest pain:

1. Send to ER by ambulance
2. Send to ER unaided
3. Evaluate and treat
4. Treat only
5. Evaluate only
6. Send home

These actions on patients are generalized. For example, a patient with gastrointestinal problems may be treated with antacids and instructed to drink lots of water, but MES only specifies that the patient is to be treated. Similarly, how a patient is evaluated is ambiguous. The clearly specified actions are when patients are sent to the ER by ambulance or sent home. Patients sent to the ER unaided usually drive to the ER, but this does not rule out other means of reaching the ER such as being driven or taking public transportation.

The conversation in Appendix D was had with MES. A 30 year old man comes into the doctor's office complaining of sharp chest pain. The patient newly experienced a syncope (loss of consciousness), he has T-wave and ST inversion on his EKG, the pain radiates to the interscapular region (shoulders), and the pain lasts 15 minutes at a time. The patient has no other symptoms.

The program is attempting to answer what should be done to the patient by checking all possible actions that could be taken on the patient. This is why all base level inputs are asked of the user. Relevant inputs are highlighted in bold.

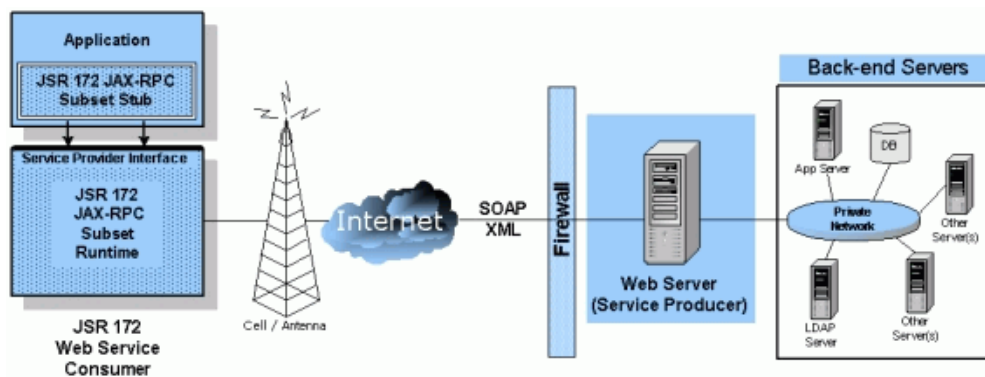
MES ultimately determines from the user input that patient has unstable angina, and therefore he should be sent to the ER by ambulance. MES determined that patient had angina from his abnormal EKG showings of ST and T-wave inversion, and used the recent syncopal episode to determine that patient has new symptoms. The presence of new symptoms tells MES that the angina is unstable, and that Patient should therefore be sent to the ER by ambulance.

### **3.5.3 How does MES Works?**

MES is an end-to-end mobile expert system. It includes a mobile system at the user side which is developed by Java Micro Edition on the Net Beans Integrated

Development Environment (Appendix E). This part only contains the graphical user interface and the connection information. There is no expert system code at this part. After compilation a JAR file is created which is deployed the mobile operating system. The main deployment environment is to Symbian Operating Systems.

Connection from mobile systems to the expert system is provided by the Web Services. K-SOAP is used for this purpose, k-SOAP is the lightweight environment and it is designed for mobile system (Appendix F). Signal comes from a mobile device web service client to connection provider, after that it's connected to internet with SOAP protocol as shown in Figure 12.



**Figure 12: End-to-End System Architecture**

At the backend an expert system is constructed to wait process. To develop an expert system, Jess expert system shell is used. Jess is completely developed with Java. Rules (Appendix B) and Facts (Appendix C) are created when developing MES. Web service server is located in front of the Jess, is deployed to Oracle Application Server (OC4J). The data which comes from a mobile system goes to web service then expert system shell.

### 3.5.4 The problem solving paradigm

#### 3.5.4.1 The paradigm

MES uses a rule-based system. Rules were well suited to the domain because they get right to the point. For example, if the patient's EKG has a loss of R wave, then it is immediately inferred that the patient has myocardial infarction (heart attack), and therefore should be sent to the ER by ambulance. Likewise, if a patient complaining of chest pain had a past heart attack, then MES knows to send the patient to the ER by ambulance.

#### **3.5.4.2 What problems do MES handles?**

MES was designed to handle as many cases of chest pain as possible, especially the common cases. Particularly, MES handles all serious causes of chest pain. The serious causes of chest pain are life-threatening. Even with only a few key symptoms, MES will know to send patients with such symptoms to the hospital by ambulance. The life-threatening cases are myocardial infarction, unstable angina, aortic aneurysm, aortic dissection, pulmonary embolism, pneumothorax, suicide, emergent headaches, extensive pneumonia, looks seriously ill, dehydration and acute abdomen. In addition, MES deals with patients with high risk, esophageal disease, esophageal cancer, musculoskeletal problems, gastrointestinal problems such as diarrhea, myocarditis, pleurisy, psychogenic problems such as depression and anxiety, headaches, costochondritis, pneumonia, herpes zoster, and functional disorders.

## **CHAPTER 4**

### **CONCLUSION**

There are several ways to develop mobile expert system. Developing on a mobile device is one of the ways. The other, which is proposed at this thesis, is end-to-end system. GUI is only developed on mobile device and the expert system which is developed by Jess Expert System Shell located at the remote server. The connection between the mobile system and expert system should be provided web services, remote method invocation etc. At this thesis Web Service implementation is done to connect those systems each other. K-SOAP is used for this purpose.

Although user have to pay some charge to service provider (e.g. Turkcell) to the connection due to web services; the system still has much more advantages. The main is that expert system is located on the different server so processor power or memory size of mobile device is not so considerable when developing the end-to-end system. Another advantage is that scalability of expert system. The rules and facts are developed only one expert system and all user are connected to system. As a result of this, expert system upgrade or maintenance is easily done.

## REFERENCES

- [1] **CREIVER D.** (1993), *AI: The Tumultuous History of the Search for Artificial Intelligence*, Basic Books.
  
- [2] **ROLSTON, D.** (1988), *Principles of Artificial intelligence and expert system developement*, McGraw Hill, New York.
  
- [3] **RUSSEL, S., NORVIG, P.** (2003), *Artificial Intelligence A modern Approach*, Prentice Hall, New Jersey.
  
- [4] **DARLINGTON, K.** (2000), *The Essence of Expert System*, Prentice Hall, Essex.
  
- [5] **DOLE J.** (1983), *A Society of Mind*, Carnegie Mellon University Department of Computer Science Tech. Report #127.
  
- [6] **GIARRATANO, S.C., RILEY, G.D.** (2005), *Expert Systems Principles and Programming*, Thomson Learning, Boston.
  
- [7] **KASHHAP N.** (1997), *An Expert Systems Application in Electromagnetic Compatibility*, University of Missouri-Rolla, Missouri.
  
- [8] **LAND, L.** (1995), *Knowledge Based Systems Usage*, McGraw-Hill, Berkshire.
  
- [9] **CARRICO M.A., GIRARD J.E.** (1989), *Building Knowledge Systems: Developing and Managing Rule Based Applications*, McGraw Hill, New York.
  
- [10] **FRIEDMAN E.** (2003), *Jess in Action*, Manning, Greenwich.



- [11] <http://www.jessrules.com/jess/index.shtml>
- [12] **KALAKOTA R., ROBINSON M.** (2001), *M-Business: The Race to Mobility*, McGraw-Hill, Osborne.
- [13] **MACKINTOSH R., KEEN G.W.** (2001), *The Freedom Economy: Gaining the mCommerce Edge in the Era of the Wireless Internet*, McGraw-Hill, Osborne.
- [14] **SCHILLER, J.** (2003), *Mobile Communications*, Addison Wesley, Harlow.
- [15] <http://symbian.org/developer/index.html>
- [16] <http://www.microsoft.com/windowsmobile/about/default.mspx>
- [17] <http://www.palm.com/us/index.html>
- [18] **MALLICK M.** (2003), *Mobile and Wireless Design Essentials*, John Wiley and Sons, New York.
- [19] <http://java.sun.com/features/1998/05/birthday.html>
- [20] **MUCHOW J.W.** (2002), *Core J2ME Technology and MIDP*, Prentice Hall, New York.
- [21] <http://java.sun.com/products/j2mewtoolkit/>
- [22] <http://www.jcp.org/en/jsr/detail?id=139>
- [23] <http://www.jcp.org/en/jsr/detail?id=36>
- [24] **SAMUEL A.L.** (1959), *Some studies in machine learning using the game of checkers*, IBM Journal of Research and Development, 210-219.
- [25] **TOPLEY K.** (2002), *J2ME in a Nutshell*, O'Reilly and Associates, New York.
- [26] <http://www.jcp.org/en/jsr/detail?id=172>

[27] <http://www.ksoap.org/>

[28] <http://ksoap.enhydra.org/>

[29] <http://ksoap2.enhydra.org/>

[30] **TANENBAUM, A.S., STEEN, M.** (2002), *Distributed System Principles and Paradigms*, Prentice Hall, New Jersey.

[31] **GROSSO W.** (2001), *Java RMI*, O'Reilly, New York.

[32] <http://java.sun.com/products/rmiop>

## APPENDIX A

### Structure of SOAP Message

```
<?xml version="1.0"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
  <soap:Header>  
    ...  
  </soap:Header>  
  <soap:Body>  
    ...  
  </soap:Body>  
</soap:Envelope>
```

```
--MIME_boundary  
Content-Type: image/tiff  
Content-Transfer-Encoding: binary  
Content-ID: <phone_model_6630.tiff@nokia.com>  
... binary tiff image ...  
--MIME_boundary—
```

## APPENDIX B

### Decision Rules to the Mobile Expert System

```
(import mes.model.*)  
(deftemplate Question (declare (from-class Question)))  
(deftemplate Answer (declare (from-class Answer)))  
(deftemplate Facts (declare (from-class Facts)))
```

```
(deftemplate node
```

```
  (slot name)  
  (slot type)  
  (slot question)  
  (slot yes-node)  
  (slot no-node)  
  (slot answer))
```

```
(defrule initialize-1  
  (not (node (name root))))
```

```
=>
```

```
(load-facts "mes.dat")  
(assert (current-node root)))
```

```
(defrule initialize-2  
  (declare (salience 100))  
  ?fact <- (next-gensym-idx ?idx)
```

```
=>
```

```
(retract ?fact)  
(setgen ?idx)
```

```
(defrule Start
  "Start it"
  (Answer (answer ?answer))
  (not(answer ?))
  (Facts {node == ?answer} (node ?node) (question ?question) (type ?type)
  (yes_node ?yes_node) (no_node ?no_node))
```

=>

```
(add (new Question ?node ?question ?type ?yes_node ?no_node)))
```

```
(defrule ask-decision-node-question
  ?node <- (current-node ?name)
  (node (name ?name)
    (type decision)
    (question ?question))
  (not (answer ?))
```

=>

```
(printout t ?question " (yes or no) ")
(assert (answer (read))))
```

```
(defrule bad-answer
  ?answer <- (answer ~yes&~no)
```

=>

```
(retract ?answer))
```

```
(defrule proceed-to-yes-branch
  ?node <- (current-node ?name)
  (node (name ?name)
        (type decision)
        (yes-node ?yes-branch))
  ?answer <- (answer yes)

=>

(retract ?node ?answer)
(assert (current-node ?yes-branch)))
```

```
(defrule proceed-to-no-branch
  ?node <- (current-node ?name)
  (node (name ?name)
        (type decision)
        (no-node ?no-branch))
  ?answer <- (answer no)

=>

(retract ?node ?answer)
(assert (current-node ?no-branch)))
```

```
(defrule ask-if-answer-node-is-correct
  ?node <- (current-node ?name)
  (node (name ?name) (type answer) (answer ?value))
  (not (answer ?)))
```

=>

```
(printout t "PATIENT should be " ?value crlf)
(assert (answer (read))))
```

```
(defrule answer-node-guess-is-incorrect
  ?node <- (current-node ?name)
  (node (name ?name) (type answer))
  ?answer <- (answer no)
```

=>

```
(assert (replace-answer-node ?name))
(retract ?answer ?node))
```

```
(defrule ask-try-again
  (ask-try-again)
  (not (answer ?))

=>

  (printout t "Try again? (yes or no) ")
  (assert (answer (read))))
```

```
(defrule one-more-time
  ?phase <- (ask-try-again)
  ?answer <- (answer yes)

=>

  (retract ?phase ?answer)
  (assert (current-node root)))
```



## APPENDIX C

### Mobile Expert System Facts

(MAIN::node (name root) (type decision) (question "Is it the case that PATIENT's pain lasts less than two seconds : ") (yes-node node5) (no-node node1) (answer nil))

(MAIN::node (name node1) (type decision) (question "Is it the case that PATIENT has too much diarrhea to keep up by drinking : ") (yes-node ans2) (no-node node2) (answer nil))

(MAIN::node (name node2) (type decision) (question "Is it the case that PATIENT's pain is relieved by antacids : ") (yes-node node5) (no-node node3) (answer nil))

(MAIN::node (name node3) (type decision) (question "Is it the case that PATIENT has excessive release of watery feces : ") (yes-node ans3) (no-node node4) (answer nil))

(MAIN::node (name node4) (type decision) (question "Is it the case that PATIENT has dysphagia : ") (yes-node node2) (no-node node5) (answer nil))

(MAIN::node (name node5) (type decision) (question "Is it the case that PATIENT's pain is aggravated by meals: ") (yes-node ans4) (no-node node6) (answer nil))

(MAIN::node (name node6) (type decision) (question "Is it the case that PATIENT does eat and taste again : ") (yes-node ans4) (no-node node7) (answer nil))

(MAIN::node (name node7) (type decision) (question "Is it the case that PATIENT's EKG has loss of R wave : ") (yes-node node12) (no-node node8) (answer nil))

(MAIN::node (name node8) (type decision) (question "Is it the case that PATIENT's EKG has ST inversion : ") (yes-node node9) (no-node node15) (answer nil))

(MAIN::node (name node9) (type decision) (question "Is it the case that PATIENT's EKG does have ST elevation : ") (yes-node ans2) (no-node node10) (answer nil))

(MAIN::node (name node10) (type decision) (question "Is it the case that PATIENT's EKG has T wave inversion : ") (yes-node node11) (no-node node15) (answer nil))

(MAIN::node (name node11) (type decision) (question "Is it the case that PATIENT has burning substernal pain : ") (yes-node ans4) (no-node node12) (answer nil))

(MAIN::node (name node12) (type decision) (question "Is it the case that PATIENT has belches : ") (yes-node node15) (no-node node13) (answer nil))

(MAIN::node (name node13) (type decision) (question "Is it the case that PATIENT's pain is aggravated by respiration : ") (yes-node ans4) (no-node node14) (answer nil))

(MAIN::node (name node14) (type decision) (question "Is it the case that PATIENT's pain is aggravated by lying down : ") (yes-node ans4) (no-node node15) (answer nil))

(MAIN::node (name node15) (type decision) (question "Is it the case that PATIENT has a headache : ") (yes-node node21) (no-node node16) (answer nil))

(MAIN::node (name node16) (type decision) (question "Is it the case that PATIENT has pounding on one side of head : ") (yes-node ans4) (no-node node17) (answer nil))

(MAIN::node (name node17) (type decision) (question "Is it the case that PATIENT has tightness in front of head or back of neck: ") (yes-node ans4) (no-node node18) (answer nil))

(MAIN::node (name node18) (type decision) (question "Is it the case that PATIENT's pain is relieved by medication : ") (yes-node node20) (no-node node19) (answer nil))

(MAIN::node (name node19) (type decision) (question "Is it the case that PATIENT's pain is relieved by sublingual nitroglycerine : ") (yes-node ans5) (no-node node20) (answer nil))

(MAIN::node (name node20) (type decision) (question "Is it the case that PATIENT's pain is relieved by nitrates : ") (yes-node ans5) (no-node node21) (answer nil))

(MAIN::node (name node21) (type decision) (question "Is it the case that PATIENT has night pain : ") (yes-node ans5) (no-node node22) (answer nil))

(MAIN::node (name node22) (type decision) (question "Is it the case that PATIENT has heartburn : ") (yes-node ans5) (no-node node23) (answer nil))

(MAIN::node (name node23) (type decision) (question "Is it the case that PATIENT coughs : ") (yes-node node26) (no-node node24) (answer nil))

(MAIN::node (name node24) (type decision) (question "Is it the case that PATIENT extensive pneumonia: ") (yes-node ans6) (no-node node25) (answer nil))

(MAIN::node (name node25) (type decision) (question "Is it the case that PATIENT coughs up mucus : ") (yes-node node29) (no-node node26) (answer nil))

(MAIN::node (name node26) (type decision) (question "Is it the case that PATIENT has fever : ") (yes-node ans5) (no-node node27) (answer nil))

(MAIN::node (name node27) (type decision) (question "Is it the case that PATIENT has abnormal breath sounds on stethoscope : ") (yes-node node29) (no-node node28) (answer nil))

(MAIN::node (name node28) (type decision) (question "Is it the case that PATIENT's pain is occurring with less provocation : ") (yes-node ans3) (no-node node29) (answer nil))

(MAIN::node (name node29) (type decision) (question "Is it the case that PATIENT's pain is increasing in intensity : ") (yes-node ans3) (no-node node30) (answer nil))

(MAIN::node (name node30) (type decision) (question "Is it the case that PATIENT's pain is lasting longer : ") (yes-node ans3) (no-node node31) (answer nil))

(MAIN::node (name node31) (type decision) (question "Is it the case that PATIENT' pain is increasing in frequency : ") (yes-node ans3) (no-node node32) (answer nil))

(MAIN::node (name node32) (type decision) (question "Is it the case that PATIENT's pain is aggravated by cough : ") (yes-node node36) (no-node node33) (answer nil))

(MAIN::node (name node33) (type decision) (question "Is it the case that PATIENT's pain is aggravated by inspiration : ") (yes-node ans3) (no-node node34) (answer nil))

(MAIN::node (name node34) (type decision) (question "Is it the case that PATIENT's pain is aggravated by movement : ") (yes-node ans4) (no-node node35) (answer nil))

(MAIN::node (name node35) (type decision) (question "Is it the case that PATIENT has panic episodes : ") (yes-node ans3) (no-node node36) (answer nil))

(MAIN::node (name node36) (type decision) (question "Is it the case that PATIENT's pain lasts hours to days : ") (yes-node node39) (no-node node37) (answer nil))

(MAIN::node (name node37) (type decision) (question "Is it the case that PATIENT's pain is unrelated by exertion : ") (yes-node ans3) (no-node node38) (answer nil))

(MAIN::node (name node38) (type decision) (question "Is it the case that PATIENT's pain is relieved by rest : ") (yes-node ans3) (no-node node39) (answer nil))

(MAIN::node (name node39) (type decision) (question "Is it the case that PATIENT has hypertension : ") (yes-node ans3) (no-node node40) (answer nil))

(MAIN::node (name node40) (type decision) (question "Is it the case that PATIENT smokes : ") (yes-node node48) (no-node node41) (answer nil))

(MAIN::node (name node41) (type decision) (question "Is it the case that PATIENT has hyperlipidemia : ") (yes-node ans5) (no-node node42) (answer nil))

(MAIN::node (name node42) (type decision) (question "Is it the case that PATIENT is obese : ") (yes-node ans5) (no-node node43) (answer nil))

(MAIN::node (name node43) (type decision) (question "Is it the case that PATIENT uses procainamide, hydralize, or isoniazid : ") (yes-node ans3) (no-node node44) (answer nil))

(MAIN::node (name node44) (type decision) (question "Is it the case that PATIENT has tenderness over joints where the ribs meet the breast bone: ") (yes-node ans3) (no-node node45) (answer nil))

(MAIN::node (name node45) (type decision) (question "Is it the case that PATIENT's pain lasts seconds to days : ") (yes-node node46) (no-node node49) (answer nil))

(MAIN::node (name node46) (type decision) (question "Is it the case that PATIENT's pain is sharp : ") (yes-node node47) (no-node node49) (answer nil))

(MAIN::node (name node47) (type decision) (question "Is it the case that PATIENT has tingling or electrical feeling : ") (yes-node ans3) (no-node node48) (answer nil))

(MAIN::node (name node48) (type decision) (question "Is it the case that PATIENT has numbness : ") (yes-node ans3) (no-node node49) (answer nil))

(MAIN::node (name node49) (type decision) (question "Is it the case that PATIENT has vibrating sensation : ") (yes-node ans3) (no-node node50) (answer nil))

(MAIN::node (name node50) (type decision) (question "Is it the case that PATIENT has heightened sensitivity near rash : ") (yes-node ans3) (no-node node51) (answer nil))

(MAIN::node (name node51) (type decision) (question "Is it the case that PATIENT has a rash : ") (yes-node ans3) (no-node node52) (answer nil))

(MAIN::node (name node52) (type decision) (question "Is it the case that PATIENT's pain radiates to interscapular region: ") (yes-node ans6) (no-node ans4) (answer nil))

(MAIN::node (name ans1) (type answer) (question nil) (yes-node nil) (no-node nil) (answer "Send Home"))

(MAIN::node (name ans2) (type answer) (question nil) (yes-node nil) (no-node nil) (answer "ER-Unaided"))

(MAIN::node (name ans3) (type answer) (question nil) (yes-node nil) (no-node nil) (answer "Evaluate and Treat"))

(MAIN::node (name ans4) (type answer) (question nil) (yes-node nil) (no-node nil)  
(answer "Evaluate"))

(MAIN::node (name ans5) (type answer) (question nil) (yes-node nil) (no-node nil)  
(answer "Treat"))

(MAIN::node (name ans6) (type answer) (question nil) (yes-node nil) (no-node nil)  
(answer "ER by Ambulance"))

## APPENDIX D

### Sample Dialogue on the Mobile Expert System

Is it the case that PATIENT's pain lasts less than two seconds : No

Is it the case that PATIENT has too much diarrhea to keep up by drinking : No

Is it the case that PATIENT's pain is relieved by antacids: No

Is it the case that PATIENT has excessive release of watery feces: No

Is it the case that PATIENT has dysphagia: No

Is it the case that PATIENT's pain is aggravated by meals: No

Is it the case that PATIENT does eat and taste again: No

Is it the case that PATIENT's EKG has loss of R wave: No

**Is it the case that PATIENT's EKG has ST inversion: Yes**

Is it the case that PATIENT's EKG does have ST elevation: No

**Is it the case that PATIENT's EKG has T wave inversion: Yes**



Is it the case that PATIENT has burning substernal pain: No

Is it the case that PATIENT has belches: No

Is it the case that PATIENT's pain is aggravated by respiration: No

Is it the case that PATIENT's pain is aggravated by lying down: No

Is it the case that PATIENT has a headache: No

Is it the case that PATIENT has pounding on one side of head: No

Is it the case that PATIENT has tightness in front of head or back of neck: No

Is it the case that PATIENT's pain is relieved by medication: No

Is it the case that PATIENT's pain is relieved by sublingual nitroglycerine: No

Is it the case that PATIENT's pain is relieved by nitrates: No

Is it the case that PATIENT has night pain: No

Is it the case that PATIENT has heartburn: No

Is it the case that PATIENT coughs : No

Is it the case that PATIENT extensive pneumonia : No

Is it the case that PATIENT coughs up mucus: No

Is it the case that PATIENT has fever: No

Is it the case that PATIENT has abnormal breath sounds on stethoscope: No

Is it the case that PATIENT's pain is occurring with less provocation: No

Is it the case that PATIENT's pain is increasing in intensity: No

Is it the case that PATIENT's pain is lasting longer: No

Is it the case that PATIENT' pain is increasing in frequency: No

Is it the case that PATIENT's pain is aggravated by cough: No

Is it the case that PATIENT's pain is aggravated by inspiration: No

Is it the case that PATIENT's pain is aggravated by movement: No

Is it the case that PATIENT has panic episodes: No

Is it the case that PATIENT's pain lasts hours to days: No

Is it the case that PATIENT's pain is unrelated by exertion: Yes

Is it the case that PATIENT's pain is relieved by rest: No

Is it the case that PATIENT has hypertension: No

Is it the case that PATIENT smokes : No

Is it the case that PATIENT has hyperlipidemia: No

Is it the case that PATIENT is obese: No

Is it the case that PATIENT uses procainamide, hydralize, or isoniazid: No

Is it the case that PATIENT has tenderness over joints where the ribs meet the breast bone: No

**Is it the case that PATIENT's pain lasts seconds to days: Yes**

**Is it the case that PATIENT's pain is sharp: Yes**

Is it the case that PATIENT has tingling or electrical feeling: No

Is it the case that PATIENT has numbness: No

Is it the case that PATIENT has vibrating sensation: No

Is it the case that PATIENT has heightened sensitivity near rash: No

Is it the case that PATIENT has a rash: No

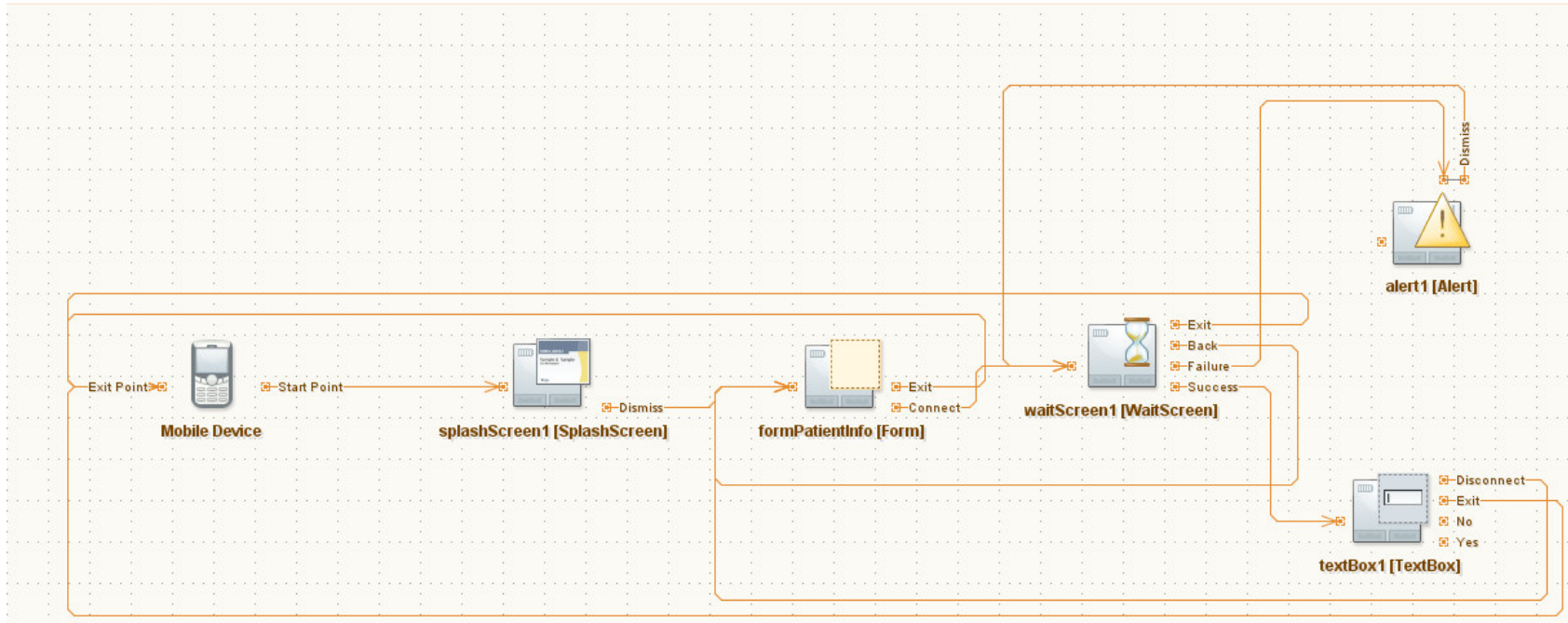
**Is it the case that PATIENT's pain radiates to interscapular region: Yes**

[WHAT-TO-DO PATIENT SEND-TO-ER-BY-AMBULANCE]

## **APPENDIX E**

### **Flow Design of Mobile System**

Flow Designer: MIDP-2.0



## APPENDIX F

### Web Service Definition Language of Mobile Expert System

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<definitions
  name="MESWS"
  targetNamespace="http://tempuri.org"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://tempuri.org"
  xmlns:ns1="http://mes.ws/IMESWS.xsd">
  <types>
    <schema
      targetNamespace="http://mes.ws/IMESWS.xsd"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
      <complexType name="mes_model_Question" jdev:packageName="mes.model"
        xmlns:jdev="http://xmlns.oracle.com/jdeveloper/webservices">
        <all>
          <element name="question" type="string"/>
          <element name="type" type="string"/>
          <element name="node" type="string"/>
          <element name="yes_node" type="string"/>
          <element name="no_node" type="string"/>
          <element name="answer" type="string"/>
        </all>
      </complexType>
    </schema>
  </types>
</definitions>
```

```

    </complexType>
    <complexType name="ArrayOfString"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <complexContent>
    <restriction base="SOAP-ENC:Array">
    <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="xsd:string[]"/>
    </restriction>
    </complexContent>
    </complexType>
</schema>

```

```

</types>
<message name="initExpertSystem0Request"/>
<message name="initExpertSystem0Response">
    <part name="return" type="ns1:mes_model_Question"/>
</message>
<message name="initExpertSystem21Request"/>
<message name="initExpertSystem21Response">
    <part name="return" type="ns1:ArrayOfString"/>
</message>
<message name="initExpertSystem32Request"/>
<message name="initExpertSystem32Response">
    <part name="return" type="xsd:string"/>
</message>
<message name="initExpertSystem43Request">
    <part name="ans" type="xsd:string"/>
</message>
<message name="initExpertSystem43Response">
    <part name="return" type="xsd:string"/>
</message>
<portType name="WebServicesPortType">
    <operation name="initExpertSystem">
    <input name="initExpertSystem0Request"
message="tns:initExpertSystem0Request"/>
    <output name="initExpertSystem0Response"
message="tns:initExpertSystem0Response"/>
    </operation>
    <operation name="initExpertSystem2">
    <input name="initExpertSystem21Request"
message="tns:initExpertSystem21Request"/>
    <output name="initExpertSystem21Response"
message="tns:initExpertSystem21Response"/>
    </operation>
    <operation name="initExpertSystem3">
    <input name="initExpertSystem32Request"
message="tns:initExpertSystem32Request"/>
    <output name="initExpertSystem32Response"
message="tns:initExpertSystem32Response"/>
    </operation>
    <operation name="initExpertSystem4">

```



```

    <input name="initExpertSystem43Request"
message="tns:initExpertSystem43Request"/>
    <output name="initExpertSystem43Response"
message="tns:initExpertSystem43Response"/>
  </operation>
</portType>
<binding name="WebServicesBinding" type="tns:WebServicesPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="initExpertSystem">

    <soap:operation soapAction="" style="rpc"/>
    <input name="initExpertSystem0Request">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output name="initExpertSystem0Response">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="initExpertSystem2">
    <soap:operation soapAction="" style="rpc"/>
    <input name="initExpertSystem21Request">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output name="initExpertSystem21Response">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="initExpertSystem3">
    <soap:operation soapAction="" style="rpc"/>
    <input name="initExpertSystem32Request">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output name="initExpertSystem32Response">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="initExpertSystem4">
    <soap:operation soapAction="" style="rpc"/>
    <input name="initExpertSystem43Request">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output name="initExpertSystem43Response">
      <soap:body use="encoded" namespace="MESWS"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>

```

```
    </output>
  </operation>
</binding>
<service name="MESWS">
  <port name="WebServicesPort" binding="tns:WebServicesBinding">
    <soap:address location="http://localhost:8888/Projects-MES-context-
root/MESWS"/>

  </port>
</service>
</definitions>
```